



IntechOpen

New Developments in Robotics Automation and Control

Edited by Aleksandar Lazinica



**NEW DEVELOPMENTS IN
ROBOTICS, AUTOMATION AND CONTROL**

EDITED BY
ALEKSANDAR LAZINICA

New Developments in Robotics Automation and Control

<http://dx.doi.org/10.5772/84>

Edited by Aleksandar Lazinica

© The Editor(s) and the Author(s) 2008

The moral rights of the and the author(s) have been asserted.

All rights to the book as a whole are reserved by INTECH. The book as a whole (compilation) cannot be reproduced, distributed or used for commercial or non-commercial purposes without INTECH's written permission.

Enquiries concerning the use of the book should be directed to INTECH rights and permissions department (permissions@intechopen.com).

Violations are liable to prosecution under the governing Copyright Law.



Individual chapters of this publication are distributed under the terms of the Creative Commons Attribution 3.0 Unported License which permits commercial use, distribution and reproduction of the individual chapters, provided the original author(s) and source publication are appropriately acknowledged. If so indicated, certain images may not be included under the Creative Commons license. In such cases users will need to obtain permission from the license holder to reproduce the material. More details and guidelines concerning content reuse and adaptation can be found at <http://www.intechopen.com/copyright-policy.html>.

Notice

Statements and opinions expressed in the chapters are those of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published chapters. The publisher assumes no responsibility for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained in the book.

First published in Croatia, 2008 by INTECH d.o.o.

eBook (PDF) Published by IN TECH d.o.o.

Place and year of publication of eBook (PDF): Rijeka, 2019.

IntechOpen is the global imprint of IN TECH d.o.o.

Printed in Croatia

Legal deposit, Croatia: National and University Library in Zagreb

Additional hard and PDF copies can be obtained from orders@intechopen.com

New Developments in Robotics Automation and Control

Edited by Aleksandar Lazinica

p. cm.

ISBN 978-953-7619-20-6

eBook (PDF) ISBN 978-953-51-5740-3

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,200+

Open access books available

116,000+

International authors and editors

125M+

Downloads

151

Countries delivered to

Our authors are among the
Top 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Meet the editor



Alex Lazinica is the founder and CEO of IntechOpen. After obtaining a Master's degree in Mechanical Engineering, he continued his PhD studies in Robotics at the Vienna University of Technology. Here he worked as a robotic researcher with the university's Intelligent Manufacturing Systems Group as well as a guest researcher at various European universities, including the Swiss Federal Institute of Technology Lausanne (EPFL). During this time he published more than 20 scientific papers, gave presentations, served as a reviewer for major robotic journals and conferences and most importantly he co-founded and built the International Journal of Advanced Robotic Systems- world's first Open Access journal in the field of robotics. Starting this journal was a pivotal point in his career, since it was a pathway to founding IntechOpen - Open Access publisher focused on addressing academic researchers needs. Alex is a personification of IntechOpen key values being trusted, open and entrepreneurial. Today his focus is on defining the growth and development strategy for the company.

Preface

This book represents the contributions of the top researchers in the field of robotics, automation and control and will serve as a valuable tool for professionals in these interdisciplinary fields.

The book consists of 25 chapters introducing both basic research and advanced developments. Topics covered include kinematics, dynamic analysis, accuracy, optimization design, modelling, simulation and control.

This book is certainly a small sample of the research activity going on around the globe as you read it, but it surely covers a good deal of what has been done in the field recently, and as such it works as a valuable source for researchers interested in the involved subjects.

Special thanks to all authors, which have invested a great deal of time to write such interesting and high quality chapters.

Editor
Aleksandar Lazinica

Contents

Preface	IX
1. The Area Coverage Problem for Dynamic Sensor Networks <i>Simone Gabriele and Paolo Di Giamberardino</i>	001
2. Multichannel Speech Enhancement <i>Lino García and Soledad Torres-Guijarro</i>	027
3. Multiple Regressive Model Adaptive Control <i>Emil Garipov, Teodor Stolkov and Ivan Kalaykov</i>	059
4. Block-synchronous harmonic control for scalable trajectory planning <i>Bernard Girau, Amine Boumaza, Bruno Scherrer and Cesar Torres-Huitzil</i>	085
5. Velocity Observer for Mechanical Systems <i>Ricardo Guerra, Claudiu Iurian and Leonardo Acho</i>	111
6. Evolution of Neuro-Controllers for Trajectory Planning Applied to a Bipedal Walking Robot with a Tail <i>Álvaro Gutiérrez, Fernando J. Berenguer and Félix Monasterio-Huelin</i>	121
7. Robotic Proximity Queries Library for Online Motion Planning Applications <i>Xavier Giralt, Albert Hernansanz, Alberto Rodriguez and Josep Amat</i>	143
8. Takagi-Sugeno Fuzzy Observer for a Switching Bioprocess: Sector Nonlinearity Approach <i>Enrique J. Herrera-López, Bernardino Castillo-Toledo, Jesús Ramírez-Córdova and Eugénio C. Ferreira</i>	155
9. An Intelligent Marshalling Plan Using a New Reinforcement Learning System for Container Yard Terminals <i>Yoichi Hirashima</i>	181
10. Chaotic Neural Network with Time Delay Term for Sequential Patterns <i>Kazuki Hirozawa and Yuko Osana</i>	195
11. PDE based approach for segmentation of oriented patterns <i>Aymeric Histace, Michel Ménard and Christine Cavaro-Ménard</i>	207
12. The robot voice-control system with interactive learning <i>Miroslav Holada and Martin Pelc</i>	219

13.	Intelligent Detection of Bad Credit Card Accounts <i>Yo-Ping Huang, Frode Eika Sandnes, Tsun-Wei Chang and Chun-Chieh Lu</i>	229
14.	Improved Chaotic Associative Memory for Successive Learning <i>Takahiro Ikeya and Yuko Osana</i>	247
15.	Kohonen Feature Map Associative Memory with Refractoriness based on Area Representation <i>Tomohisa Imabayashi and Yuko Osana</i>	259
16.	Incremental Motion Planning With Las Vegas Algorithms <i>Jouandeau Nicolas, Touati Youcef and Ali Cherif Arab</i>	273
17.	Hierarchical Fuzzy Rule-Base System for MultiAgent Route Choice <i>Habib M. Kammoun, Ilhem Kallel and Adel M. Alimi</i>	285
18.	The Artificial Neural Networks applied to servo control systems <i>Yuan Kang , Yi-Wei Chen, Ming-Huei Chu and Der-Ming Chry</i>	303
19.	Linear Programming in Database <i>Akira Kawaguchi and Andrew Nagel</i>	339
20.	Searching Model Structures Based on Marginal Model Structures <i>Sung-Ho Kim and Sangjin Lee</i>	355
21.	Active Vibration Control of a Smart Beam by Using a Spatial Approach <i>Ömer Faruk Kircali, Yavuz Yaman, Volkan Nalbantoğlu and Melin Şahin</i>	377
22.	Time-scaling in the control of mechatronic systems <i>Bálint Kiss and Emese Szádeczky-Kard</i>	411
23.	Heap Models, Composition and Control <i>Jan Komenda, Sébastien Lahaye* & Jean-Louis Boimond</i>	427
24.	Batch Deterministic and Stochastic Petri Nets and Transformation Analysis Methods <i>Labadi Karim, Amodeo Lionel and Haoxun Chen</i>	449
25.	Automatic Estimation of Parameters of Complex Fuzzy Control Systems <i>Yulia Ledeneva, René García Hernández and Alexander Gelbukh</i>	475

The Area Coverage Problem for Dynamic Sensor Networks

Simone Gabriele, Paolo Di Giamberardino
Università degli Studi di Roma "La Sapienza"
Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
Italy

1. Introduction

In this section a brief description of area coverage and connectivity maintenance for sensor networks is given together with their collocation in the scientific literature. Particular attention is given to dynamic sensor networks, such as sensor networks in which sensing nodes move continuously, under the assumption, reasonable in many applications, that synchronous or asynchronous discrete time measures are acceptable instead of continuous ones.

1.1 Area Coverage

Environmental monitoring of lands, seas or cities, cleaning of parks, squares or lakes, mine clearance and critical structures surveillance are only a few of the many applications that are connected with the concept of *area coverage*.

Area coverage is always referred to a set, named *set of interest*, and to an action: then, covering means acting on all the physical locations of the set of interest.

Within the several actions that can be considered, such as manipulating, cleaning, watering and so on, sensing is certainly one of the most considered in literature. Recent technological advances in wireless networking and miniaturizing of electronic computers, have suggested to face the problem of taking measures on large, hazardous and dynamic environments using a large number of smart sensors, able to do simple elaborations and perform data exchange over a communication network. This kind of distributed sensors systems have been named, by the scientific and engineering community, *sensor networks*.

Coverage represents a significant measure of the quality of service provided by a sensor network. Considering static sensors, the coverage problem has been addressed in terms of optimal usage of a given set of sensors, randomly deployed, in order to assure full coverage and minimizing energy consumption (Cardei and Wu, 2006, Zhang and Hou, 2005, Stojmenovic, 2005), or in terms of optimal sensors deployment on a given area, such as optimizing sensors locations, as in (Li et al., 2003, Meguerdichian et al., 2001, Chakrabarty et al., 2002, Isler et al., 2004, Zhou et al., 2004).

The introduction of mobile sensors allows to develop networks in which sensors, starting from an initial random deployment condition, evaluate and move through optimal locations.

In (Li and Cassandras, 2005) coverage maximization using sensors with limited range, while minimizing communications cost, is formulated as an optimization problem. A gradient algorithm is used to drive sensors from initial positions to suboptimal locations.

In (Howard, 2002) an incremental deployment algorithm is presented. Nodes are deployed one-at-a-time into an unknown complex environment, with each node making use of information gathered by previously deployed nodes. The algorithm is designed to maximize network coverage while ensuring line-of-sight between nodes.

A stable feedback control law, in both continuous and discrete time, to drive sensors to so-called centroidal Voronoy configurations, that are critical points of the sensors locations optimization problem, is presented in (Cortes et al., 2004).

Other interesting works on self deploying or self configuring sensor networks are (Cheng and Tsai, 2003, Sameera and Gaurav S., 2004, Tsai et al., 2004)

The natural evolution of these kind of approaches moves in the direction of giving a greater motion capabilities to the network. And once the sensors can move autonomously in the environment, the measurements can be performed also during the motion (*dynamic coverage*). Then, under the assumption, reasonable in many applications, that synchronous or asynchronous discrete time measures are acceptable instead of continuous ones, the number of sensors can be strongly reduced. Moreover, faults or critical situations can be faced and solved more efficiently, simply changing the paths of the working moving sensors. Clearly, coordinated motion of such *dynamic sensor network*, imposes additional requirements, such as avoiding collisions or preserving communication links between sensors. In order to better motivate why and when a mobile sensor network can be a more successful choice than a static one, some considerations are reported. So, given an area A_{tot} to be measured by a sensor network, and ρ_S the measure range of each sensor (sensors are here supposed homogeneous, otherwise the same considerations should be repeated for all the homogeneous subnets), the number N_{stat} of sensors needed for a static network must satisfy

$$N_{stat} \geq \frac{A_{tot}}{\pi \rho_S^2} \quad (1)$$

When a dynamic network is considered, the area covered by sensors is a time function and, clearly, it not decreases as time passes. A simplified discrete time model of the evolution of the area still uncovered, at (discrete) time $t = k + 1$, by a dynamic sensor network moving with the strategy proposed in this chapter, can be given by the following differences equation

$$A_u(k + 1) = \left(1 - \frac{\dot{A}_N}{A_{tot}} \right) A_u(k) \quad (2)$$

where

$$\dot{A}_N = \frac{v_{max}}{2\rho_S} A_{tot} \left(1 - \left(1 - \frac{\pi \rho_S^2}{A_{tot}} \right)^N \right)$$

represents the area covered in the time unit by a number N of mobile sensors subject to the maximum motion velocity v_{max} . Measurements are then modelled as obtained deploying randomly N static sensors on the workspace every $\frac{2\rho_S}{v_{max}}$ seconds. Denoting by

$$A_u(0) = A_{tot} \left(1 - \frac{\pi\rho_S^2}{A_{tot}}\right)^N$$

the initial condition for area to be covered, at each discrete time $t = k$ the fraction of area covered is given by

$$A_{\%}(k) = 1 - \frac{A_u(k)}{A_{tot}} = 1 - \left[\frac{A_u(0)}{A_{tot}} \left(1 - \frac{\dot{A}_N}{A_{tot}}\right)^k \right] \quad (3)$$

The evolution computed using (3) with $N = 5$, $N = 10$ and $N = 15$ has been compared with the results of simulations where the approach described in the chapter is applied. In Fig. 1 this comparison is reported, showing that (3) is a good model for describing the relationship between the area covered and the time using a dynamic solution.

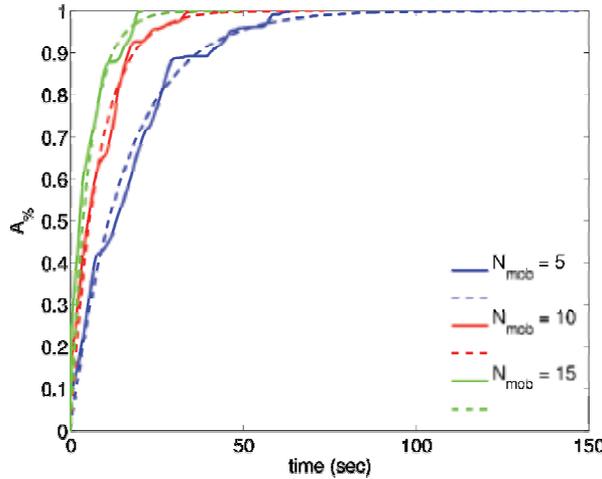


Fig. 1. Comparison between coverage evolution obtained by the model (2) (*dashed*) and simulations of the proposed coverage strategy (*solid*) for different numbers of moving sensors

Then, referring to surveillance tasks, (3) can be used to evaluate the minimum number of sensors (with given ρ_S and v_{max}) required to cover a given fraction $\tilde{A}_{\%}$ of the area of interest according to a given measurement rate. In fact, it is possible to write the relation between the maximum rate at which the network can cover the $\tilde{A}_{\%}$ fraction of A_{tot} and the number of moving sensors as

$$f = \frac{\log\left(1 - \frac{\dot{A}_N}{A_{tot}}\right)}{\log\left(1 - \tilde{A}_{\%}\right) - N \log\left(1 - \frac{\pi \rho_S^2}{A_{tot}}\right)} \quad (4)$$

Such a relationship between N and f is depicted in Fig. 2, showing, as intuitively expected, almost a proportionality between number of sensors and frequency of measurement at each point of the area A_{tot} .

The motivation and the support of the dynamic solution is evidenced by Fig. (1): lower is the refresh frequency of the measurements at each point (that is higher are the time intervals between measurements) and lower is the number of sensors required, once sensors motion is introduced.

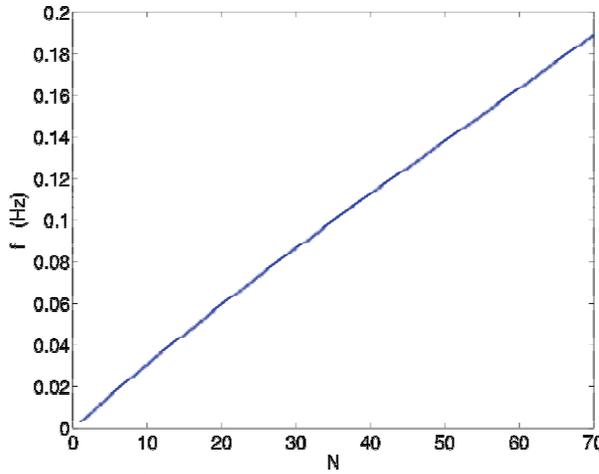


Fig. 2. Maximum measure rate f in function of number of moving sensors. ($A_{tot} = 400 \text{ m}^2$, $\rho_S = 1.5 \text{ m}$, $v_{max} = 1.5 \frac{\text{m}}{\text{sec}}$, $\tilde{A}_{\%} = 0.98$)

Under the assumption of dynamic network, the area coverage problem is posed in terms of looking for optimal trajectories for the N moving sensors in presence of some constraints like communication connection preservation, motion limitations, energetic considerations and so on. In (Tsai et al., 2004, Cecil and Marthler, 2004) the dynamic coverage problem for multiple sensors is studied, with a variational approach, in the level set framework, obstacles occlusions are considered, suboptimal solutions are proposed also in three dimensional environments ((Cecil and Marthler, 2006)). A survey of coverage path planning algorithms for mobile robots moving on the plane is presented in (Choset, 2001). In (Acar et al., 2006) the dynamic coverage problem for one mobile robot with finite range detectors is studied and an approach based on space decomposition and Voronoy graphs is proposed. In (Hussein and Stipanovic, 2007), a distributed control law is developed that guarantees to meet the coverage goal with multiple mobile sensors under the hypothesis of communication network connection. Collisions avoidance is considered.

Various problems associated with optimal path planning for mobile observers such as mobile robots equipped with cameras to obtain maximum visual coverage in the three-dimensional Euclidean space are considered in (Wang, 2003). Numerical algorithms for solving the corresponding approximate problems are proposed.

In (Gabriele and Di Giamberardino, 2007c, Gabriele and Di Giamberardino, 2007a, Gabriele and Di Giamberardino, 2007b) a general formulation of dynamic coverage is given by the authors, a sensor network model is proposed and an optimal control formulation is given. Suboptimal solutions are computed by discretization. Sensors and actuators limits, geometric constraints, collisions avoidance, and communication network connectivity maintenance are considered.

The approaches introduced up to now, also by the authors ((Gabriele and Di Giamberardino, 2007c, Gabriele and Di Giamberardino, 2007a, Gabriele and Di Giamberardino, 2007b)), are referred to homogeneous sensor networks, that is each node in the network is equivalent to any other one in terms of sensing capabilities (same sensor or same set of sensors over each node). Sensor network nodes were called *heterogeneous* with respect to different aspects. In (Ling Lam and Hui Liu, 2007), the problem of deploying a set of mobile sensor nodes, with heterogeneous sensing ranges, to give coverage is addressed.

In (Lazos and Poovendran, 2006), evaluating coverage of a set of sensors, with arbitrary different shapes, deployed according to an arbitrary stochastic distribution is formulated as a set intersection problem.

In (Hussein et al., 2007) the use of two classes of vehicles are used to dynamically cover a given domain of interest. The first class is composed of vehicles, whose main responsibility is to dynamically cover the domain of interest. The second class is composed of coordination vehicles, whose main responsibility is to effectively communicate coverage information across the network.

The problem of deploying nodes, equipped with different sets of sensors, is studied in (Shih et al., 2007) in order to cover a sensing field in which multiple attributes are required to be sensed.

In this chapter the case of different magnitudes to be measured on a given set of interest is considered. Network nodes are then heterogeneous, like in (Shih et al., 2007), with respect to the set of sensors with which they are equipped. Moreover different sensors can have different sensing ranges.

1.2 Connectivity Maintenance

Communication aspects are crucial in the design of a multi sensor systems ((Holger Karl, 2005, Stojmenovic, 2005, Santi, 2005)).

Connectivity is obviously necessary for data exchanging and aggregation but also for localization and coordination, in fact, it is often assumed in formation stabilization (Olfati-Saber and Murray, 2002) or consensus problems (Olfati-Saber et al., 2007).

In classical wireless sensor network (Holger Karl, 2005, Stojmenovic, 2005, Akyildiz et al., 2002, Santi, 2005), composed by densely deployed static sensors, a single node has many neighbours with which direct communication would be possible when using sufficiently large transmission power. However high transmission power requires lots of energy, then, it could be useful to deliberately restrict the set of neighbours controlling transmission power, and then communication range, or by simply turning off some nodes for a certain time. For such networks connectivity can then be achieved opportunely deploying nodes or controlling communication power.

For a dynamic sensor network the problem is more challenging, because network topology is, indeed, *dynamic*. Connectivity maintenance became, then, a motion coordination problem. Each sensor is assumed to have a fixed range over which communication is not reliable. Communication network can be modelled as a state dependent dynamic graph; topology, depending from sensors positions, changes while sensors moves.

Then, connectivity maintenance impose to introduce constraints on the instantaneous positions of sensors. The simplest ways to achieve connectivity is to maintain the starting communication graph topology that's assumed to be connected. This can be obtained imposing fixed topology Maintenance as proposed in (Gabriele and Di Giamberardino, 2007a) or flocking (Olfati-Saber, 2006). However, this approaches impose strong constraints to sensors movement and that can affect other aspects as shown in (Gabriele and Di Giamberardino, 2007b) for coverage. Is then more desirable to allow topology to change over time, even though that introduce challenging dynamic graph control problems.

In (Mesbahi, 2004), starting from a class of problems associated with control of distributed dynamic systems, a controllability framework for state-dependent dynamic graphs is considered.

In (Kim and Mesbahi, 2005) the position of a dynamic state-dependent graph vertices are controlled in order to maximize the second smallest eigenvalue of the Laplacian matrix, also named *algebraic connectivity* and that has emerged as a critical parameter that influences the stability and robustness properties of dynamic systems that operate over an information network.

In (Spanos and Murray, 2004) a measure of robustness of local connectedness of a network is introduced that can be computed by local communication only.

K-hop connectivity preservation is considered, in (Zavlanos, 2005), for a network with dynamic nodes. A centralized control framework that guarantees maintenance of this property is developed. Connectivity is modelled as an invariance problem and transformed into a set of constraints on the control variable.

In (Gabriele and Di Giamberardino, 2007b) a centralized approach to connectivity Maintenance, based on preservation of the edges of one *Minimum Spanning Tree* of the communication graph, is proposed by the authors. Connection Maintenance is introduced as a constraint of an optimization problem.

2. General Formulation

In this section definitions are given in order to introduce useful notations. A general model of a dynamic sensor network is given considering heterogeneous sensors. The coverage problem is formulated with respect to multiple magnitudes, connectivity maintenance constraints are considered. In the following sections additional hypothesis are introduced in order to simplify the general problem and to evaluate suboptimal solutions.

2.1 Dynamic Sensor networks

Let be W a specified spatial domain, a compact subset of the real Euclidean space \mathbb{R}^n ($n=2,3$) called the *set of interest*. The representation of a point $p \in W$ with respect to a given orthonormal basis for \mathbb{R}^n is denoted by $x_p = (x_{p1}, \dots, x_{pn})$.

Let be $\Xi = \{\xi_1, \xi_2, \dots\}$ the set of magnitudes of interest defined on W .

A dynamic sensor network can be view a set $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$ of m mobile sensors. Each *mobile sensor* can be represented by: $\sigma_j = \langle \mathcal{C}^{(j)}, f^{(j)}, \Xi^{(j)}, \{M_\xi^{(j)} \mid \xi \in \Xi^{(j)}\}, \kappa^{(j)} \rangle$ where:

- $\mathcal{C}^{(j)}$ is the sensor configuration space.
- $f^{(j)}$ is the sensor dynamic function, that describe the evolution of sensor configuration according to a control input $u^{(j)}$:

$$\dot{q}^{(j)} = f^{(j)}(q^{(j)}, u^{(j)})$$
- $\Xi^{(j)} \subseteq \Xi$ is the set of magnitudes that sensor σ_j can measure.
- $M_\xi^{(j)} = M_\xi^{(j)}(q^{(j)}) \subseteq W$ is the subset of W within sensor σ_j , in configuration q can measure magnitude $\xi \in \Xi^{(j)}$. Let say that sensor σ_j in configuration $q^{(j)}$ ξ – *cover* the set $M_\xi^{(j)}(q^{(j)})$
- $\kappa^{(j)} : \mathcal{C} \times \mathcal{C} \rightarrow \mathbb{R}$ is the sensor communication function, such as, σ_j in configuration $q^{(j)}$ can communicate with a sensor σ_h in configuration $q^{(h)}$ if and only if $\kappa^{(j)}(q^{(j)}, q^{(h)}) > 0$

Looking at the whole network is possible to define generalized configuration and generalized input as:

$$q = \begin{bmatrix} q^{(1)} \\ q^{(2)} \\ \vdots \\ q^{(m)} \end{bmatrix} \quad u = \begin{bmatrix} u^{(1)} \\ u^{(2)} \\ \vdots \\ u^{(m)} \end{bmatrix}$$

At the same manner the generalized dynamic of the whole network can be written as:

$$\dot{q} = f(q, u) = \begin{bmatrix} f^{(1)}(q^{(1)}, u^{(1)}) \\ f^{(2)}(q^{(2)}, u^{(2)}) \\ \vdots \\ f^{(m)}(q^{(m)}, u^{(m)}) \end{bmatrix}$$

2.2 Coverage

Let indicate with $q^{(j)}(\Theta)$ the evolution of sensor σ_j configuration during a given a time interval $\Theta = [0, t_f]$. It is possible to define the subset of W ξ – *covered* by σ_j during Θ as:

$$M_\xi^{(j)}(q^{(j)}(\Theta)) = \bigcup_{t \in \Theta} M_\xi^{(j)}(q^{(j)}(t)) \quad (5)$$

Considering generalized configuration $q^{(\Theta)}$ is possible to define the network field of measure, respect to magnitude ξ during Θ , as:

$$M_{\xi}(q(\Theta)) = \bigcup_{\sigma_j | \xi \in \Xi^{(j)}} M_{\xi}^{(j)}(q^{(j)}(\Theta)) = \bigcup_{\sigma_j | \xi \in \Xi^{(j)}} \bigcup_{t \in \Theta} M_{\xi}^{(j)}(q^{(j)}(t)) \quad (6)$$

Looking at the whole magnitudes set, the subset of W “ Ξ – covered” by the network can be defined as:

$$M_{\Xi}(q(\Theta)) = \bigcap_{\xi \in \Xi} M_{\xi}(q(\Theta)) \quad (7)$$

The area Ξ – covered by the sensor network during Θ is then the measure of $M(q(\Theta))$:

$$A_{\Xi}(\Theta) = \mu(M_{\Xi}(q(\Theta))) \quad (8)$$

2.3 Communication

According with their communication capabilities sensors can be view as nodes of a dynamic communication network. This network can be represented by a dynamic graph

$$\mathcal{G}(t) = \langle N_{\mathcal{G}}, E_{\mathcal{G}}(t) \rangle$$

where

$$N_{\mathcal{G}} = \sigma$$

indicate the nodes set and

$$E_{\mathcal{G}}(t) = \{(\sigma_j, \sigma_h) \in N_{\mathcal{G}} \times N_{\mathcal{G}} | \kappa^{(j)}(q^{(j)}(t), q^{(h)}(t)) > 0\}$$

indicate the edges set. As seen the edge set is time varying because it depends from the network generalized configuration q .

An alternative representation of the communication graph can be given using the adjacency matrix $A_{\mathcal{G}}(t)$:

$$A_{\mathcal{G}}^{(j,h)}(t) = \kappa^{(j)}(q^{(j)}(t), q^{(h)}(t))$$

2.4 Area Coverage Problem

Making the sensor network to cover the set of interest means evaluating controls that drive the network to measure the value of every magnitude on all the points of W , according with some constrains.

Constrains can be due, for example, from:

- Limitation of sensors motion or measure rate
- Avoiding collisions between sensors

- Maintaining some communication networks features (topology, connectivity,...)

3. Dynamic Sensor Network Model

In this section the general model defined in 2 is specified adding the hypothesis of **L**inear sensors dynamic, **P**roximity based measure model, **P**roximity based communication. Let refer to this particular model as (**LPP**) Model.

3.1 Sensors Dynamics

Each sensor σ_j is modelled, from the dynamic point of view, as a material point of mass m_j moving on \mathbb{R}^2 . The motion is assumed to satisfy the classical simple equations

$$\ddot{x}^{(j)}(t) = \frac{u^{(j)}(t)}{m_j} \quad (9)$$

where $x^{(j)}$ is the sensor position on \mathbb{R}^2 . Sensor configuration is represented by:

$$q^{(j)}(t) = \left(\dot{x}_1^{(j)}(t), x_1^{(j)}(t), \dot{x}_2^{(j)}(t), x_2^{(j)}(t) \right)^T$$

The configuration space is then:

$$\mathcal{C}^{(j)} \subseteq \mathbb{R}^4 \quad \forall j$$

The linearity of 9 allows one to write the dynamics in the form

$$\begin{aligned} \dot{q}^{(j)}(t) &= f^{(j)}(q^{(j)}(t), u^{(j)}(t)) = A_j q^{(j)}(t) + B_j u^{(j)}(t) \\ x^{(j)}(t) &= C_j q^{(j)}(t) \end{aligned} \quad (10)$$

where

$$\begin{aligned} A_j &= \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} & B_j &= \begin{pmatrix} \frac{1}{m_j} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m_j} \\ 0 & 0 \end{pmatrix} \\ C_j &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

The evolutions of configuration (state) and position (output) are related with the input's one according with the well known equations:

$$q^{(j)}(t) = \phi_j(q^{(j)}(0), u^{(j)}(t)) = e^{A_j t} q^{(j)}(0) + \int_0^t e^{A_j(t-\tau)} B_j u^{(j)}(\tau) d\tau \quad (11)$$

and

$$x^{(j)}(t) = \psi_j(\phi_j(q^{(j)}(0), u^{(j)}(t))) = C_j \phi_j(q^{(j)}(0), u^{(j)}(t)) \quad (12)$$

In the rest of the chapter *sensor trajectory* will refer to sensor position evolution. Considering the whole network

$$q(t) = (q^{(1)}(t) \quad q^{(2)}(t) \quad \dots \quad q^{(m)}(t))^T$$

can be defined to denote the generalized configuration, and the vector

$$x(t) = (x^{(1)}(t) \quad x^{(2)}(t) \quad \dots \quad x^{(m)}(t))^T$$

to denote the generalized position that is represented, for each t , by m points in the Euclidean space. Evolution of generalized position will be named *generalized network trajectory*.

At the same manner the generalized input is defined as:

$$u(t) = (u^{(1)}(t) \quad u^{(2)}(t) \quad \dots \quad u^{(m)}(t))^T$$

Generalized dynamics for the whole network can be written as:

$$\begin{aligned} \dot{q}(t) &= Aq(t) + Bu(t) \\ x(t) &= Cq(t) \end{aligned}$$

where:

$$A = \begin{pmatrix} A_1 & 0 & \dots & 0 \\ 0 & A_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_m \end{pmatrix} \quad B = \begin{pmatrix} B_1 & 0 & \dots & 0 \\ 0 & B_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B_m \end{pmatrix}$$

$$C = \begin{pmatrix} C_1 & 0 & \dots & 0 \\ 0 & C_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C_m \end{pmatrix}$$

According with 11 and 12, generalized configuration evolution and network generalized trajectory are related with generalized input by:

$$q(t) = \Phi(q(0), q(t)) = \begin{pmatrix} \phi_1(q^{(1)}(0), u^{(1)}(t)) \\ \phi_2(q^{(2)}(0), u^{(2)}(t)) \\ \vdots \\ \phi_m(q^{(m)}(0), u^{(m)}(t)) \end{pmatrix} \quad (13)$$

and

$$x(t) = \Psi(\Phi(q(0), u(t))) = \begin{pmatrix} \psi_1(\phi_1(q^{(1)}(0), u^{(1)}(t))) \\ \psi_2(\phi_2(q^{(2)}(0), u^{(2)}(t))) \\ \vdots \\ \psi_m(\phi_m(q^{(m)}(0), u^{(m)}(t))) \end{pmatrix} \quad (14)$$

3.2 Coverage Model

It is assumed that at every time t sensor σ_j can take measures on magnitude $\xi \in \Xi^{(j)}$ in a circular area of radius ρ_ξ around its current position $x^{(j)}(t)$. The sensor field of measure respect to ξ is then a disk of centre $x^{(j)}(t)$ and radius ρ_ξ :

$$M_\xi^{(j)}(q^{(j)}) = M_\xi(q^{(j)}) = \{p \in W : \|x^{(j)} - x_p\| \leq \rho_\xi \quad \xi \in \Xi^{(j)}\} \quad (15)$$

As seen in 2.2, starting from $M_\xi^{(j)}(q^{(j)})$ is possible to define the area ξ - covered and the area Ξ - covered by the sensor network during a given time interval .

Homogeneous Sensors

A particular case is the one in witch sensors are homogeneous with respect to sensing capabilities:

$$\Xi^{(j)} = \Xi \quad \forall j$$

Assuming without loss of generality that there is only one magnitude of interest on W , the set covered by sensor σ_j at every time t can be described by:

$$M(q^{(j)}) = \{p \in W : \|q^{(j)} - x_p\| \leq \rho\} \quad (16)$$

It is possible to define the subset of W covered by the sensor network in a time interval Θ as:

$$M(q(\Theta)) = \bigcup_{\sigma_j \in \Sigma} \left[\bigcup_{t \in \Theta} M(q^{(j)}(t)) \right] \quad (17)$$

3.3 Communication Model

The communication network is modelled as an Euclidean graph. Two mobile sensors at time t are assumed to communicate each other if the distance between them is smaller than a given communication radius ρ_C .

For every sensor σ_j , the communication function is given by:

$$\kappa^{(j)}(q^{(j)}, q^{(h)}) = \kappa(q^{(j)}, q^{(h)}) = \rho_C - \|x^{(j)} - x^{(h)}\| \quad (18)$$

Is easy to see that this communication function makes the network graph \mathcal{G} undirected, in fact:

$$\kappa^{(j)}(q^{(j)}, q^{(h)}) = \kappa^{(h)}(q^{(h)}, q^{(j)}) \quad \forall j, h$$

3.4 Coverage Problem Formulation

According with the introduced model is possible to formulate the coverage problem as an optimal control problem. The idea is to maximize the area covered by sensors in a fixed time interval according with some constrains.

3.4.1 Objective Functional

In 2.2 the area ξ - covered by a set of m moving sensors is defined as the union of the measure sets of the sensors, respect to magnitude ξ , at every time t . This quantity is very hard to compute, also for the simple measure set model introduced in 3.2, then an alternative performance measure has to be used.

Defining the distance between a point p of the workspace and a generalized trajectory $x(\Theta)$, within a time interval $\Theta = [0, t_f]$, as

$$d_\xi(x(\Theta), p) = \min_{t \in \Theta, j \in \{1, 2, \dots, m\}} \|x_p - x^{(j)}(\Theta)\| \quad \xi \in \Xi^{(j)} \quad (19)$$

and making use of the function

$$\text{pos}(\chi) = \begin{cases} \chi & \text{if } \chi > 0 \\ 0 & \text{if } \chi \leq 0 \end{cases} \quad (20)$$

that fixes to zero any non positive value, the function

$$\hat{d}_\xi(x(\Theta), p, \rho_\xi) = \text{pos}(d_\xi(x(\Theta), p) - \rho_\xi) \geq 0$$

can be defined. Then, a measure of how the generalized trajectory $q(\Theta)$ produces a good ξ - coverage of the workspace can be given by

$$J_{\xi}(x(\Theta)) = \int_{p \in W} \hat{d}_{\xi}(x(\Theta), p, \rho_{\xi}) \quad (21)$$

Looking at the whole magnitudes of interest set Ξ is possible to introduce a functional that evaluate how a given generalized trajectory $x(\Theta) \Xi - cover$ the set of interest W

$$J_{\Xi}(x(\Theta)) = \sum_{\xi \in \Xi} J_{\xi}(x(\Theta)) \quad (22)$$

Smaller is $J_{\Xi}(x(\Theta))$, better is the $\Xi - coverage$ of W . If $J_{\Xi}(x(\Theta)) = 0$ then $x(\Theta) \Xi - covers$ completely the workspace.

From 3.1 is possible to see how J_{Ξ} can be also written as:

$$J_{\Xi}(q(0), u(\Theta)) = \sum_{\xi \in \Xi} J_{\xi}(\Psi(\Phi(q(0), u(\Theta)))) \quad (23)$$

Homogeneous Sensors

For the homogeneous sensors case the objective functional is:

$$J(z(0), u(\Theta)) = \int_{p \in W} \hat{d}(\Psi(\Phi(z(0), u(\Theta))), p, \rho) \quad (24)$$

3.4.2 Geometric Constraints

It is possible to constrain sensors to move inside a box subset of \mathbb{R}^2

$$x_{min} \leq x^{(j)}(t) \leq x_{max}$$

If needed is possible to set the starting and/or the final state (positions and/or speeds):

$$\begin{aligned} q(0) &= q_{start} \\ q(t_f) &= q_{end} \end{aligned}$$

A particular case is the periodic trajectories constrain, useful in tasks in which measures have to be repeated continuously:

$$q^{(j)}(0) = q^{(j)}(t_f)$$

Is also necessary to avoid collisions between sensors at every time t

$$\|x^{(j)}(t) - x^{(h)}(t)\| \geq \rho_B$$

for $j \neq h$

3.4.3 Dynamic Constraints

Physical limits on the actuators (for the motion) and/or on the sensors (in terms of velocity in the measure acquisition) suggest the introduction of the following additional constraints

$$\begin{aligned} |\dot{x}(t)| &\leq v_{max} \\ |u(t)| &\leq u_{max} \end{aligned}$$

3.4.4 Communication Constraints

As said communication network connectivity is very important for data exchange and transmission, but also for sensor localization, coordination and commands communication. Under the hypothesis that before sensors start moving the communication network is connected, it is possible to maintain connectivity introducing constraints on the instantaneous position of sensors. More strongly, it is possible to impose a fixed network topology, this can be useful, for example, to fix the level of redundancy on the communication link and then to reach node fault tolerance.

Fixed Network Topology

To maintain a fixed network topology every sensor must maintain direct communication with a subset of its starting neighbors that is fixed in time. Indicating with $\mathcal{G}_d = \langle V_G, E_{\mathcal{G}_d}(t) \rangle$ the graph that represents desired topology, where $E_{\mathcal{G}_d}(t) \subseteq E_G(t) \forall t \in \Theta$.

According with 3.3, for every edge of \mathcal{G}_d a distance constrain between a couple of sensors must be introduced, so maintaining a desired topology \mathcal{G}_d means to satisfy the following constrains set $\forall t \in \Theta$:

$$\|x^{(j)}(t) - x^{(h)}(t)\| \leq \rho_C \quad \forall (\sigma_j, \sigma_h) \in E_{\mathcal{G}_d}(t) \quad (25)$$

Network Connectivity Maintenance

Fixed topology maintenance is, obviously a particular case of connectivity maintenance if the desired topology is connected. Anyway, this approach introduces strong constrains on sensors motion. This constrains can be relaxed in only connectivity is needed, allowing network topology to change over time. That increase coverage performances as shown in (Gabriele and Di Giamberardino, 2007b).

As said before, the communication model introduced in 3.3 makes the communication graph $\mathcal{G}(t)$ to be undirected. A undirected graph is connected if and only if it contain a spanning tree. So it is possible to maintain network connectivity constraining every sensor just to maintain direct communication links that corresponds to the edges of a spanning tree of the communication tree.

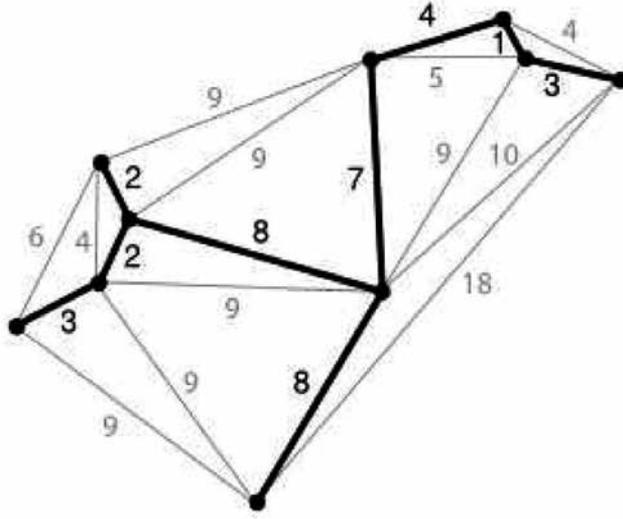


Fig. 3. Minimum Spanning Tree for a planar weighted undirected graph.

Assigning a weight at every edge of E_G is possible to define the *Minimum Spanning Tree* of G as the spanning tree with minimum weight (Figure 3). In particular being G an Euclidean graph it come natural to define the edges weights as:

$$w(x^{(j)}, x^{(h)}) = \|x^{(j)} - x^{(h)}\|$$

in this case the minimum spanning tree is said *Euclidean* (EMST). The EMST can be easily and efficiently computed by standard algorithms (such as Prim's algorithm or Kruskal's algorithm). Indicating the EMST with $\mathcal{T}(t) = \langle V_G, E_{\mathcal{T}}(t) \rangle$, where $E_{\mathcal{T}}(t) \subseteq E_G(t)$, maintaining the communication network connection means to satisfy the following constrains $\forall t \in \Theta$:

$$\|x^{(j)}(t) - x^{(h)}(t)\| \leq \rho_C \quad \forall (\sigma_j, \sigma_h) \in E_{\mathcal{T}}(t) \quad (26)$$

The minimum spanning tree of the communication network graph changes while sensors moves, so the neighbours set of every node change over time making the network topology dynamic.

3.4.5 Optimal Control Problem

The coverage problem can now be formulated as an optimal control problem:

$$\min_{q(0), u(\Theta)} J_{\Xi}(q(0), u(\Theta)) = \sum_{\xi \in \Xi} J_{\xi}(\Psi(\Phi(q(0), u(\Theta))))$$

$$\begin{aligned} x_{min} &\leq \Psi(\Phi(q(0), u(\Theta))) \leq x_{max} \quad \forall t \in \Theta \\ \|\psi(\phi(q^{(j)}(0), u^{(j)}(t))) - \psi(\phi(q^{(h)}(0), u^{(h)}(t)))\| &\geq \rho_B \quad i \neq j \quad \forall t \in \Theta \\ (q(0) &= q_{start}) \\ (q(t_f) &= q_{end}) \end{aligned}$$

$$\begin{aligned} |B_j^T \phi(q^{(j)}(0), u^{(j)}(t))| &\leq v_{max} \quad \forall j \quad \forall t \in \Theta \\ |u(t)| &\leq u_{max} \quad \forall t \in \Theta \end{aligned}$$

$$\|\psi(\phi(q^{(j)}(0), u^{(j)}(t))) - \psi(\phi(q^{(h)}(0), u^{(h)}(t)))\| \leq \rho_C \quad \forall (\sigma_j, \sigma_h) \in E_{\mathcal{T}}(t)$$

$$(\|\psi(\phi(q^{(j)}(0), u^{(j)}(t))) - \psi(\phi(q^{(h)}(0), u^{(h)}(t)))\| \leq \rho_C \quad \forall (\sigma_j, \sigma_h) \in E_{\mathcal{G}_d}(t))$$

This problem is, in general, very hard to solve analytically. In the next section a simpler discretized model is introduced to evaluate suboptimal solutions.

4. Discretized Model

In order to overcome the difficulty of the problem defined in 3.4, a discretization is performed, both with respect to space, and with respect to time in all the time dependent expressions. The workspace W is then divided into square cells, with resolution (size) l_{res} , obtaining a grid in which each point c_{rs} is the centre of a cell, and the trajectories are discretized with sample time T_s . This allow to represent the coverage problem as a solvable *Nonlinear Programming Problem*.

4.1 Sensors Discretized Dynamics

The discrete time sensors dynamic is well described by the following equations:

$$\begin{aligned} q^{(j)}((k+1)T_s) &= A_{dj}q^{(j)}(kT_s) + B_{dj}u^{(j)}(kT_s) \\ x^{(j)}(kT_s) &= C_jq^{(j)}(kT_s) \end{aligned} \tag{27}$$

where

$$A_{dj} = e^{A_j T_s} \quad B_{dj} = \int_0^{T_s} e^{A_j \tau} B_j d\tau$$

Representing the j -th sensor input sequence from time $t = 0$ to time $t = NT_s$ as:

$$u_N^{(j)} = \begin{pmatrix} u^{(j)}(0) \\ u^{(j)}(T_s) \\ \vdots \\ u^{(j)}((N-1)T_s) \end{pmatrix}$$

and defining the following vectors

$$v_N^{(j)} = \begin{pmatrix} q^{(j)}(0) \\ u_N^{(j)} \end{pmatrix} \quad H_n^{(j)} = \begin{pmatrix} A_{dj}^n \\ A_{dj}^{n-1} B_{dj} \\ \vdots \\ B_{dj} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

is possible to write state and output values at time $nT_s \leq NT_s$ as:

$$q^{(j)}(nT_s) = H_n^{(j)T} v_N^{(j)} = A_{dj}^n q^{(j)}(0) + \sum_{k=0}^{n-1} A_{dj}^k B_{dj} u^{(j)}((n-1)T_s - kT_s) \quad (28)$$

and

$$x^{(j)}(nT_s) = C_j q^{(j)}(nT_s) = C_j H_n^{(j)T} v_N^{(j)} \quad (29)$$

State and output sequences, from time $t = 0$ to time $t = NT_s$, can be represented by the following vectors:

$$q_N^{(j)} = \begin{bmatrix} q^{(j)}(0) \\ q^{(j)}(T_s) \\ \vdots \\ q^{(j)}(NT_s) \end{bmatrix} \quad x_N^{(j)} = \begin{bmatrix} x^{(j)}(0) \\ x^{(j)}(T_s) \\ \vdots \\ x^{(j)}(NT_s) \end{bmatrix}$$

According with 28 and 29 the relations between these sequences and the input one are described by:

$$q_N^{(j)} = H_N^{(j)} v_N^{(j)} = \begin{pmatrix} H_0^{(j)T} \\ H_1^{(j)T} \\ \vdots \\ H_N^{(j)T} \end{pmatrix} v_N^{(j)} \quad (30)$$

and

$$x_N^{(j)} = C_{Nj} H_N^{(j)} v_N^{(j)} = \begin{pmatrix} C_j & 0 & \dots & 0 \\ 0 & C_j & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C_j \end{pmatrix} \begin{pmatrix} H_0^{(j)T} \\ H_1^{(j)T} \\ \vdots \\ H_N^{(j)T} \end{pmatrix} v_N^{(j)} \quad (31)$$

As done in 3.1, is possible to define generalized input, state and output sequences of the whole system:

$$v_N = \begin{bmatrix} v_N^{(1)} \\ v_N^{(2)} \\ \vdots \\ v_N^{(m)} \end{bmatrix}$$

$$q_N^{(i)} = \begin{bmatrix} q_N^{(1)} \\ q_N^{(2)} \\ \vdots \\ q_N^{(m)} \end{bmatrix} \quad x_N^{(i)} = \begin{bmatrix} x_N^{(1)} \\ x_N^{(2)} \\ \vdots \\ x_N^{(m)} \end{bmatrix}$$

These sequences are related by:

$$q_N = H_N v_N = \begin{pmatrix} H_N^{(1)} & 0 & \dots & 0 \\ 0 & H_N^{(2)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & H_N^{(m)} \end{pmatrix} v_N \quad (32)$$

And

$$x_N = C_N H_N v_N = \begin{pmatrix} C_{N1} H_N^{(1)} & 0 & \dots & 0 \\ 0 & C_{N1} H_N^{(2)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & C_{Nm} H_N^{(m)} \end{pmatrix} v_N \quad (33)$$

4.2 Coverage Problem Formulation

Using the coverage model defined in 3.2 and the communication model in 3.3, it is possible to formulate the coverage problem as a nonlinear programming problem.

4.2.1 Objective Function

The objective functional defined in 3.4.1 became, after the discretization, a function of the vector v_N :

$$J_{\Xi}(v_N) = \sum_{\xi} \sum_r \sum_s \hat{d}_{\xi}(C_N H_N v_N, c_{rs}, \rho_{\xi}) \quad (34)$$

4.2.2 Nonlinear Programming Problem

Defining geometric, dynamic and communication constrains as in 3.4 is possible to write the coverage problem for a dynamic sensor network as a tractable constrained optimization problem:

$$\min_{v_N} J_{\Xi}(v_N) = \sum_{\xi} \sum_r \sum_s \hat{d}_{\xi}(C_N H_N v_N, c_{rs}, \rho_{\xi})$$

$$\begin{aligned} x_{min} &\leq C H_n^{(j)T} v_N^{(j)} \leq x_{max} \quad \forall j, \forall n = 0, 1, \dots, N \\ \|C H_n^{(j)T} v_N^{(j)} - C H_n^{(h)T} v_N^{(h)}\| &\geq \rho_B \quad j \neq h, \forall n = 0, 1, \dots, N \\ (H_0^{(j)T} V_N^{(j)} &= q_{start} \quad \forall j) \\ (H_N^{(j)T} V_N^{(j)} &= q_{end} \quad \forall j) \end{aligned}$$

$$\begin{aligned} |B^T H_n^{(j)T} v_N^{(j)}| &\leq v_{max} \quad \forall j, \forall n = 0, 1, \dots, N \\ |u^{(j)}(nT_s)| &\leq u_{max} \quad \forall j, \forall n = 0, 1, \dots, N-1 \end{aligned}$$

$$\|C_j H_n^{(j)T} v_N^{(j)} - C_h H_n^{(h)T} v_N^{(h)}\| \leq \rho_C \quad \text{if } \forall (j, h) \parallel (\sigma_j, \sigma_h) \in E_T(nT_s) \quad \forall n$$

or

$$\|C_j H_n^{(j)T} v_N^{(j)} - C_h H_n^{(h)T} v_N^{(h)}\| \leq \rho_C \quad \text{if } \forall (j, h) \parallel (\sigma_j, \sigma_h) \in E_{G_d}(nT_s) \quad \forall n$$

Suboptimal solutions can be computed using numerical methods. In the simulations performed, the SQP (Sequential Quadratic Programming) method has been applied.

5. Simulation Results

In this section simulation results for different cases are presented to show the effectiveness of the proposed methodology. At first two simulations for the single sensor case are

presented to show the *quality* of the computed trajectories that are, anyway, suboptimal. The first case considered is the one of one sensor asked to measure a magnitude ξ , defined a circular area, within a time interval $\Theta = 15 \text{ sec}$. Sensor dynamic parameters are:

$$u_{max} = 0.5 \quad v_{max} = 1.5$$

Sensor starts from position $[0, 0]$ with zero speeds. Sensor radius of measure is $\rho_\xi = 1.5$. Simulation result are showed in figure 4.

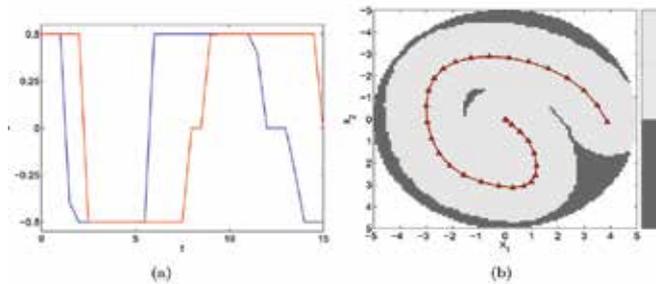


Fig. 4. One sensor covering a circular area. (a) Control components evolution. (b) Speed components evolution. (c) Sensor trajectory and coverage status of the set of interest.

In the second case, showed in figure 5, the constraint of making a cyclic trajectory is added. Cyclic trajectories are very useful for surveillance tasks. Time interval is extended to $\Theta = 25 \text{ sec}$.

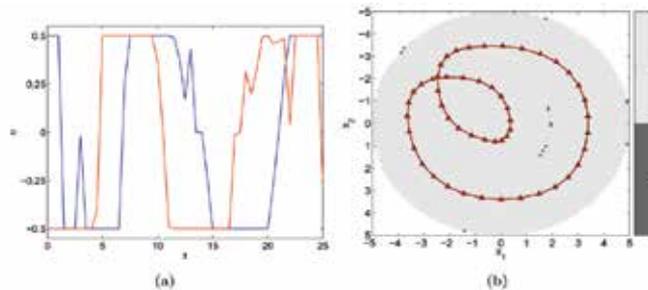


Fig. 5. One sensor covering a circular area making a cyclic trajectory. (a) Control components evolution. (b) Speed components evolution. (c) Sensor trajectory and coverage status of the set of interest

The third case considered (figure 6) is the one of an homogeneous sensor network, with three nodes, covering a box shaped workspace within a time interval $\Theta = 15 \text{ sec}$. Communication between two nodes is assumed to be reliable within a maximum range of

$$\rho_c = 5.5$$

Sensors dynamic parameters are:

$$u_{max} = 1.5 \quad v_{max} = 1.5$$

Collisions avoidance and connectivity maintenance constraints are considered.

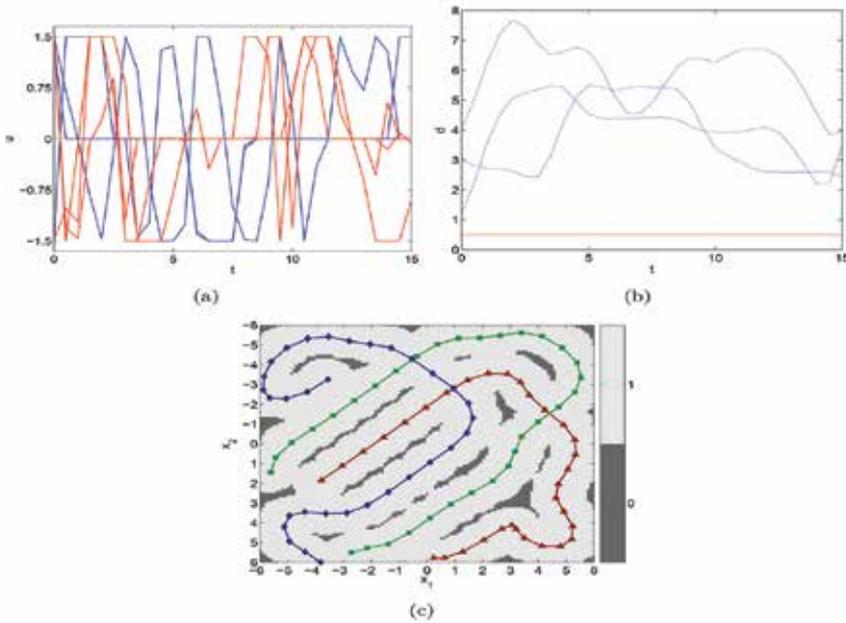


Fig. 6. Coverage of a box shaped workspace with a dynamic sensor network with three homogeneous nodes. (a) Control components evolution. (b) Relative distances between all vehicles, the red line represents minimum distance for collisions avoidance (ρ_{OB}). (c) Sensors trajectories and coverage status of the set of interest.

In figure 7 simulations are shown for the case of an heterogeneous sensor network covering a box shaped workspace within a time interval $\Theta = 15 \text{ sec}$. Three magnitudes of interest are defined,

$$\Xi = \{\xi_1, \xi_2, \xi_3\}$$

The radii within the three magnitudes can be measured are

$$\rho_{\xi_1} = 2 \quad \rho_{\xi_2} = 1 \quad \rho_{\xi_3} = 3$$

Nodes dynamic parameters are:

$$u_{max} = 1.5 \quad v_{max} = 1.5$$

Communication between two nodes is assumed to be reliable within a maximum range of

$$\rho_c = 5.5$$

The sensor network is composed by 4 nodes, with different sensing capabilities.

$$\begin{aligned} \Xi_1 &= \{\xi_1, \xi_2\} & \Xi_2 &= \{\xi_2, \xi_3\} \\ \Xi_3 &= \{\xi_1, \xi_2\} & \Xi_4 &= \{\xi_2, \xi_3\} \end{aligned}$$

Collisions avoidance and connectivity maintenance constraints are considered.

In figure 8 scenario similar to the one considered in the previous case is shown for a generic shaped workspace.

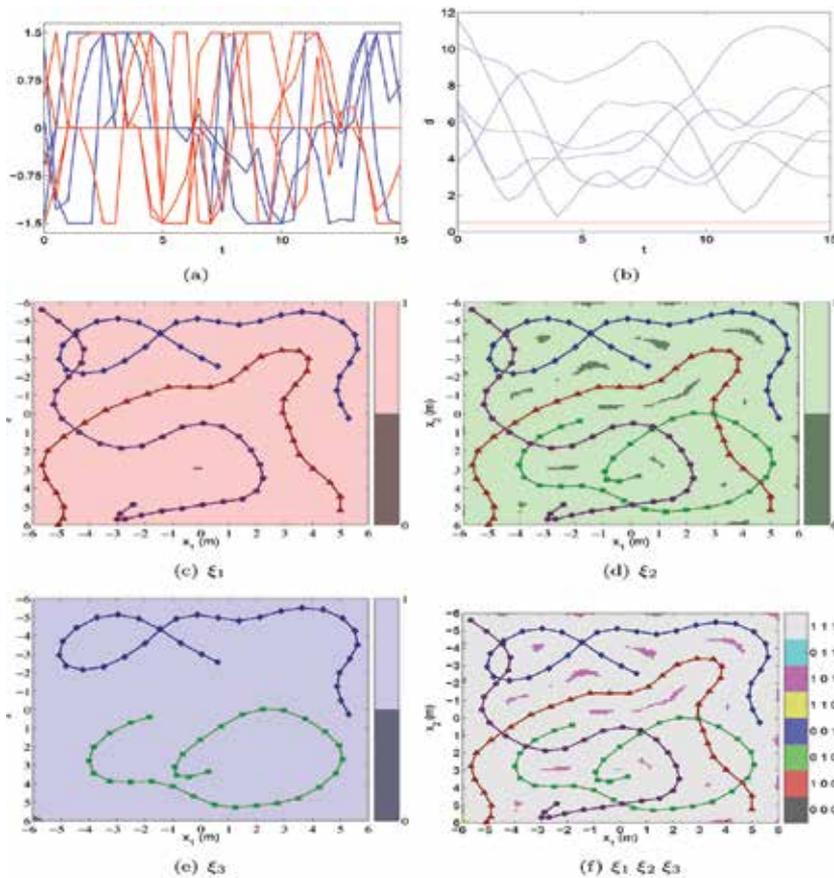


Fig. 7. Coverage of a box shaped workspace with an heterogeneous dynamic sensor network. (a) Control components evolutions. (b) Relative distances between all vehicles, the red line represents minimum distance for collisions avoidance (ρ_B). (c) ξ_1 - sensors trajectories and area ξ_1 - covered. (d) ξ_2 - sensors trajectories and area ξ_2 - covered. (e) ξ_3 - sensors trajectories and area ξ_3 - covered status. (f) All nodes trajectories and coverage status of the workspace with respect to the whole magnitudes set Ξ

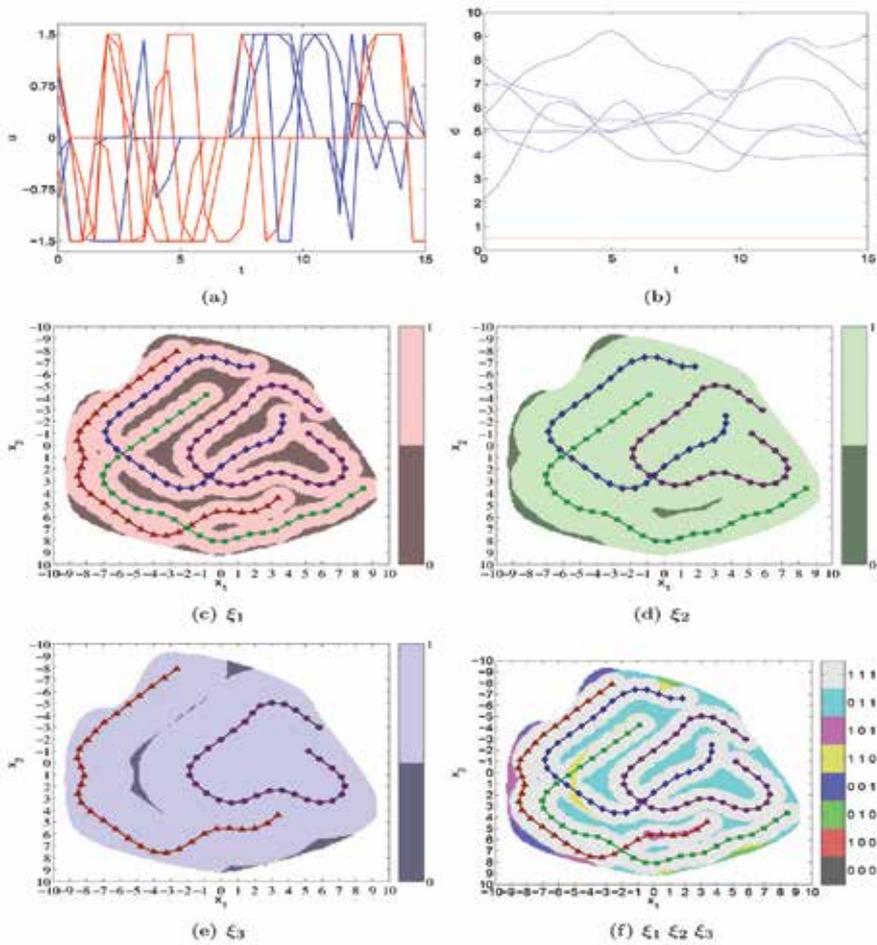


Fig. 8. Coverage of a generic shaped workspace with an heterogeneous dynamic sensor network. (a) Control components evolutions. (b) Relative distances between all vehicles, the red line represents minimum distance for collisions avoidance (ρ_B). (c) ξ_1 - sensors trajectories and area ξ_1 - covered. (d) ξ_2 - sensors trajectories and area ξ_2 - covered. (e) ξ_3 - sensors trajectories and area ξ_3 - covered status. (f) All nodes trajectories and coverage status of the workspace with respect to the whole magnitudes set Ξ

6. Conclusions

In this chapter the case of heterogeneous mobile sensor networks has been considered. The mobility of the sensors is introduced in order to allow a reduced number of sensors to measure the same field, under the assumption that the temporal resolution of the measures, i.e. the maximum time between two consecutive measures at the same coordinates, is not too small. In addition, each mobile platform representing the nodes of the net has been considered equipped with different sets of sensors, so introducing a non homogeneity in the sensor network. A general formulation of the field coverage problem as been introduced in

terms of optimal control techniques. All the constraints introduced by kinematics and dynamic limits on mobility of the moving elements as well as by communications limits (network connectivity) have been considered. A global approach has been followed making use of time and space discretization, so getting a suboptimal solution. Some simulation results show the behaviour and the effectiveness of the proposed solution.

8. References

- Acar, Choset, and Lee, J. Y. (2006). Sensor-based coverage with extended range detectors. *IEEE Transactions on Robotics and Automation*, 22(1):189–198.
- Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114.
- Cardei, M. and Wu, J. (2006). Energy-efficient coverage problems in wireless ad hoc sensor networks. *Computer communications*, 29(4):413–420.
- Cecil and Marthler (2004). A variational approach to search and path planning using level set methods. Technical report, UCLA CAM.
- Cecil and Marthler (2006). A variational approach to path planning in three dimensions using level set methods. *Journal of Computational Physics*, 221:179–197.
- Chakrabarty, K., Iyengar, S., Qi, H., and Cho, E. (2002). Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers*, 51:1448–1453.
- Cheng, L.-T. and Tsai, R. (2003). A level set framework for visibility related variational problems. Technical report, UCLA CAM.
- Choset (2001). Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126.
- Cortes, J., Martinez, S., Karatas, T., and Bullo, F. (2004). Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20:243–255.
- Gabriele, S. and Di Giamberardino, P. (2007a). Communication constraints for mobile sensor networks. In *Proceedings of the 11th WSEAS International Conference on Systems*.
- Gabriele, S. and Di Giamberardino, P. (2007b). Dynamic sensor networks. *Sensors & Transducers Journal (ISSN 1726- 5479)*, 81(7):1302–1314.
- Gabriele, S. and Di Giamberardino, P. (2007c). Dynamic sensor networks. an approach to optimal dynamic field coverage. In *ICINCO 2007, Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics, Intelligent Control Systems and Optimization*.
- Gabriele, S. and Di Giamberardino, P. (2008) Mobile sensors networks under communication constraints. *WSEAS Transactions on Systems*, 7(3): 165–174
- Holger Karl, A. W. (2005). *Protocols and Architectures for Wireless Sensor Networks*. Wiley.

- Howard, Mataric, S. (2002). An incremental self-deployment for mobile sensor networks. *Autonomous Robots*.
- Hussein, I. I. and Stipanovic, D. M. (2007). Effective coverage control using dynamic sensor networks with flocking and guaranteed collision avoidance. In *American Control Conference, 2007. ACC '07*, pages 3420–3425.
- Hussein, I. I., Stipanovic, D. M., and Wang, Y. (2007). Reliable coverage control using heterogeneous vehicles. In *Decision and Control, 2007 46th IEEE Conference on*, pages 6142–6147.
- Isler, V., Kannan, S., and Daniilidis, K. (2004). Sampling based sensor-network deployment. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*.
- Kim, Y. and Mesbahi, M. (2005). On maximizing the second smallest eigenvalue of a state-dependent graph laplacian. In *Proceedings of American Control Conference*.
- Lazos, L. and Poovendran, R. (2006). Stochastic coverage in heterogeneous sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2:325 – 358.
- Li, W. and Cassandras, C. (2005). Distributed cooperative coverage control of sensor networks. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 2542–2547.
- Li, X.-Y., Wan, P.-J., and Frieder, O. (2003). Coverage in wireless ad hoc sensor networks. *IEEE Transactions on Computers*, 52:753–763.
- ling Lam, M. and hui Liu, Y. (2007). Heterogeneous sensor network deployment using circle packings. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4442–4447.
- Meguerdichian, S., Koushanfar, F., Potkonjak, M., and Srivastava, M. (2001). Coverage problems in wireless ad-hoc sensor networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1380–1387vol.3.
- Mesbahi, M. (2004). On state-dependent dynamic graphs and their controllability properties. In *Proceedings of 43rd IEEE Conference on Decision and Control*.
- Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: algorithms and theory. *Automatic Control, IEEE Transactions on*, 51:401–420.
- Olfati-Saber, R., Fax, J. A., and Murray, R. M. (2007). Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95:215–233.
- Olfati-Saber, R. and Murray, R. (2002). Distributed structural stabilization and tracking for formations of dynamic multi-agents. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 1, pages 209–215vol.1.

- Sameera, P. and Gaurav S., S. (2004). Constrained coverage for mobile sensor networks. In *IEEE International Conference on Robotics and Automation*, pages 165-172.
- Santi, P. (2005). *Topology Control in Wireless Ad Hoc and Sensor Networks*. Wiley.
- Shih, K.-P., Chen, H.-C., and Liu, B.-J. (2007). Integrating target coverage and connectivity for wireless heterogeneous sensor networks with multiple sensing units. In *Networks, 2007. ICON 2007. 15th IEEE International Conference on*, pages 419-424.
- Spanos, D. and Murray, R. (2004). Robust connectivity of networked vehicles. In *43rd IEEE Conference on Decision and Control*.
- Stojmenovic, I. (2005). *Handbook of Sensor Networks Algorithms and Architecture*. Wiley.
- Tsai, Cheng, Osher, Burchard, and Sapiro (2004). Visibility and its dynamics in a pde based implicit framework. *Journal of Computational Physics*, 199:260-290.
- Wang, P. K. C. (2003). Optimal path planning based on visibility. *Journal of Optimization Theory and Applications*, 117:157-181.
- Zavlanos, M.M. Pappas, G. (2005). Controlling connectivity of dynamic graphs. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*.
- Zhang, H. and Hou, J. C. (2005). Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc and Sensor Wireless Networks, an International Journal*, 1:89-124.
- Zhou, Z., Das, S., and Gupta, H. (2004). Connected k-coverage problem in sensor networks. In *Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on*, pages 373-378.

Multichannel Speech Enhancement

Lino García and Soledad Torres-Guijarro
*Universidad Europea de Madrid, Universidad de Vigo
 Spain*

1. Introduction

1.1 Adaptive Filtering Review

There are a number of possible degradations that can be found in a speech recording and that can affect its quality. On one hand, the signal arriving the microphone usually incorporates multiple sources: the desired signal plus other unwanted signals generally termed as noise. On the other hand, there are different sources of distortion that can reduce the clarity of the desired signal: amplitude distortion caused by the electronics; frequency distortion caused by either the electronics or the acoustic environment; and time-domain distortion due to reflection and reverberation in the acoustic environment.

Adaptive filters have traditionally found a field of application in noise and reverberation reduction, thanks to their ability to cope with changes in the signals or the sound propagation conditions in the room where the recording takes place. This chapter is an advanced tutorial about multichannel adaptive filtering techniques suitable for speech enhancement in multiple input multiple output (MIMO) very long impulse responses. Single channel adaptive filtering can be seen as a particular case of the more complex and general multichannel adaptive filtering. The different adaptive filtering techniques are presented in a common foundation. Figure 1 shows an example of the most general MIMO acoustical scenario.

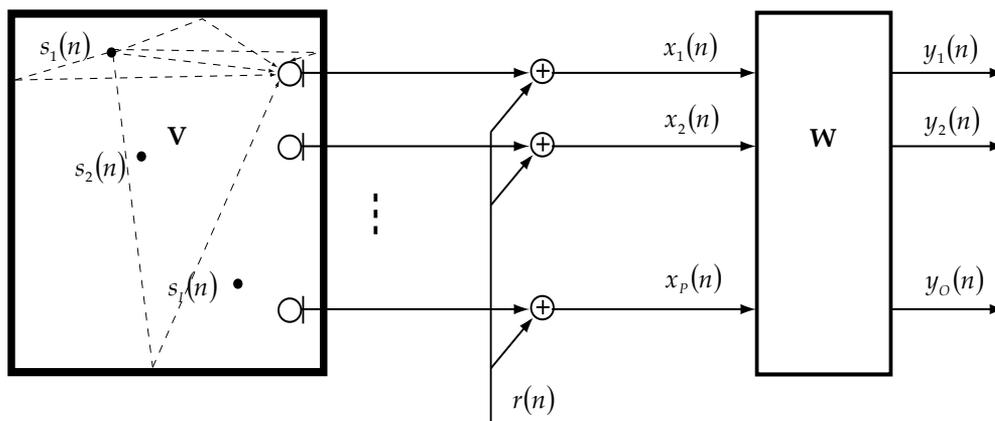


Fig. 1. Audio application scenario.

The box, on the left, represents a reverberant room. \mathbf{V} is a $P \times LI$ matrix that contains the acoustic impulse responses (AIR) between the I sources and P microphones (channels); L is a filters length. Sources can be interesting or desired signals (to enhance) or noise and interference (to attenuate). The discontinuous lines represent only the direct path and some first reflections between the $s_1(n)$ source and the microphone with output signal $x_1(n)$. Each $\mathbf{v}_{pi}(n)$ vector represents the AIR between $i=1\dots I$ and $p=1\dots P$ positions and is constantly changing depending on the position of both: source or microphone, angle between them, radiation pattern, etc.

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_{11} & \mathbf{V}_{12} & \cdots & \mathbf{V}_{1I} \\ \mathbf{V}_{21} & \mathbf{V}_{22} & \cdots & \mathbf{V}_{2I} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{V}_{P1} & \mathbf{V}_{P2} & \cdots & \mathbf{V}_{PI} \end{bmatrix},$$

$$\mathbf{v}_{pi} = [\mathcal{V}_{pi1} \ \mathcal{V}_{pi2} \ \cdots \ \mathcal{V}_{piL}]. \quad (1)$$

$r(n)$ is an additive noise or interference signal. $x_p(n)$, $p=1\dots P$ is a corrupted or poor quality signal that wants to be improved. The filtering goal is to obtain a \mathbf{W} matrix so that $y_o(n) \approx \hat{s}_i(n)$ corresponds to the identified signal. The signals in the Fig. 1 are related by

$$\mathbf{x}(n) = \mathbf{V}\mathbf{s}(n) + r(n), \quad (2)$$

$$\mathbf{y}(n) = \mathbf{W}\mathbf{x}(n). \quad (3)$$

$\mathbf{s}(n)$ is a $LI \times 1$ vector that collects the source signals,

$$\mathbf{s}(n) = [\mathbf{s}_1^T(n) \ \mathbf{s}_2^T(n) \ \cdots \ \mathbf{s}_I^T(n)]^T, \quad (4)$$

$$\mathbf{s}_i(n) = [s_i(n) \ s_i(n-1) \ \cdots \ s_i(n-L+1)]^T.$$

$\mathbf{x}(n)$ is a $P \times 1$ vector that corresponds to the convolutive system output excited by $\mathbf{s}(n)$ and the adaptive filter input of order $O \times LP$. $x_p(n)$ is an input corresponding to the channel p containing the last L samples of the input signal x ,

$$\mathbf{x}(n) = [\mathbf{x}_1^T(n) \ \mathbf{x}_2^T(n) \ \cdots \ \mathbf{x}_P^T(n)]^T, \quad (5)$$

$$\mathbf{x}_p(n) = [x_p(n) \ x_p(n-1) \ \cdots \ x_p(n-L+1)]^T.$$

\mathbf{W} is an $O \times LP$ adaptive matrix that contains an AIRs between the P inputs and O outputs

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_{11} & \mathbf{w}_{12} & \cdots & \mathbf{w}_{1P} \\ \mathbf{w}_{21} & \mathbf{w}_{22} & \cdots & \mathbf{w}_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}_{O1} & \mathbf{w}_{O2} & \cdots & \mathbf{w}_{OP} \end{bmatrix},$$

$$\mathbf{w}_{op} = [w_{op1} \ w_{op2} \ \cdots \ w_{opL}]. \quad (6)$$

For a particular output $o = 1 \dots O$, normally matrix \mathbf{W} is rearranged as column vector

$$\mathbf{w} = [\mathbf{w}_1 \ \mathbf{w}_2 \ \cdots \ \mathbf{w}_p]^T. \quad (7)$$

Finally, $y(n)$ is an $O \times 1$ target vector, $\mathbf{y}(n) = [y_1(n) \ y_2(n) \ \cdots \ y_O(n)]^T$.

The used notation is the following: a or α is a scalar, \mathbf{a} is a vector and \mathbf{A} is a matrix in time-domain \mathbf{a} is a vector and \mathbf{A} is a matrix in frequency-domain. Equations (2) and (3) are in matricial form and correspond to convolutions in a time-domain. The index n is the discrete time instant linked to the time (in seconds) by means of a sample frequency F_s according to $t = nT_s$, $T_s = 1/F_s$. T_s is the sample period. Superscript T denotes the transpose of a vector or a matrix, $*$ denotes the conjugate of a vector or a matrix and superscript H denotes Hermitian (the conjugated transpose) of a vector or a matrix. Note that, if adaptive filters are $L \times 1$ vectors, L samples have to be accumulated per channel (i.e. delay line) to make the convolutions (2) and (3).

The major assumption in developing linear time-invariant (LTI) systems is that the unwanted noise can be modeled by an additive Gaussian process. However, in some physical and natural systems, noise can not be modelled simply as an additive Gaussian process, and the signal processing solution may also not be readily expressed in terms of mean squared errors (MSE)¹.

From a signal processing point of view, the particular problem of noise reduction generally involves two major steps: *modeling* and *filtering*. The modelling step generally involves determining some approximations of either the noise spectrum or the input signal spectrum. Then, some filtering is applied to emphasize the signal spectrum or attenuate/reject the noise spectrum (Chau, 2001). Adaptive filtering techniques are used largely in audio applications where the ambient noise environment has a complicated spectrum, the statistics are rapidly varying and the filter coefficients must automatically change in order to maintain a good intelligibility of the speech signal. Thus, filtering techniques must be

¹ MSE is the best estimator for random (or stochastic) signals with Gaussian distribution (normal process). The Gaussian process is perhaps the most widely applied of all stochastic models: most error processes, in an estimation situation, can be approximated by a Gaussian process; many non-Gaussian random processes can be approximated with a weighted combination of a number of Gaussian densities of appropriated means and variances; optimal estimation methods based on Gaussian models often result in linear and mathematically tractable solutions and the sum of many independent random process has a Gaussian distribution (central limit theorem) (Vaseghi, 1996).

powerful, precise and adaptive. Most *non-referenced* noise reduction systems have only one single input signal. The task of estimating the noise and/or signal spectra must then make use of the information available only from the single input signal and the noise reduction filter will also have only the input signal for filtering. *Referenced* adaptive noise reduction/cancellation systems work well only in constrained environments where a good reference input is available, and the crosstalk problem is negligible or properly addressed.

2. Multichannel Adaptive Filters

In a multichannel system ($P > 1$) it is possible to remove noise and interference signals by applying sophisticated adaptive filtering techniques that use spatial or redundant information. However there are a number of noise and distortion sources that can not be minimized by increasing the number of microphones. Examples of this are the surveillance, recording, and playback equipment. There are several classes of adaptive filtering (Honig & Messerschmitt, 1984) that can be useful for speech enhancement, as will be shown in Sect. 4. The differences among them are based on the external connections to the filter. In the *estimator* application [see Fig. 2(a)], the internal parameters of the adaptive filter are used as *estimate*. In the *predictor* application [see Fig. 2(b)], the filter is used to filter an input signal, $x(n)$, in order to minimize the output signal, $e(n) = x(n) - y(n)$, within the constraints of the filter structure. A *predictor* structure is a linear weighting of some finite number of past input samples used to *estimate* or *predict* the current input sample. In the *joint-process estimator* application [see Fig. 2(c)] there are two inputs, $x(n)$ and $d(n)$. The objective is usually to minimize the size of the output signal, $e(n) = d(n) - y(n)$, in which case the objective of the adaptive filter itself is to generate an estimate of $d(n)$, based on a filtered version of $x(n)$, $y(n)$ (Honig & Messerschmitt, 1984).

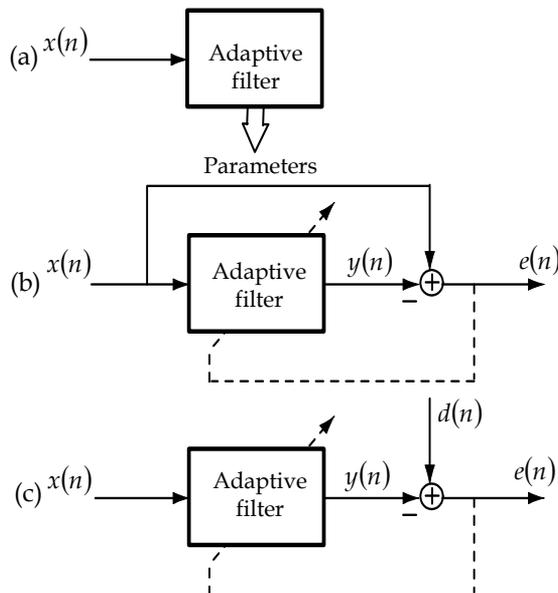


Fig. 2. Classes of adaptive filtering.

2.1 Filter Structures

Adaptive filters, as any type of filter, can be implemented using different structures. There are three types of *linear filters with finite memory*: the *transversal filter*, *lattice predictor* and *systolic array* (Haykin, 2002).

2.1.1 Transversal

The *transversal filter*, *tapped-delay line filter* or *finite-duration impulse response filter (FIR)* is the most suitable and the most commonly employed structure for an adaptive filter. The utility of this structure derives from its simplicity and generality.

The multichannel transversal filter output used to build a joint-process estimator as illustrated in Fig. 2(c) is given by

$$y(n) = \sum_{p=1}^P \sum_{l=1}^L w_{pl} x_p(n-l+1) = \sum_{p=1}^P \langle \mathbf{w}_p, \mathbf{x}_p(n) \rangle = \langle \mathbf{w}, \mathbf{x}(n) \rangle . \tag{8}$$

Where $\mathbf{x}(n)$ is defined in (5) and \mathbf{w} in (7). Equation (8) is called *finite convolution sum*.

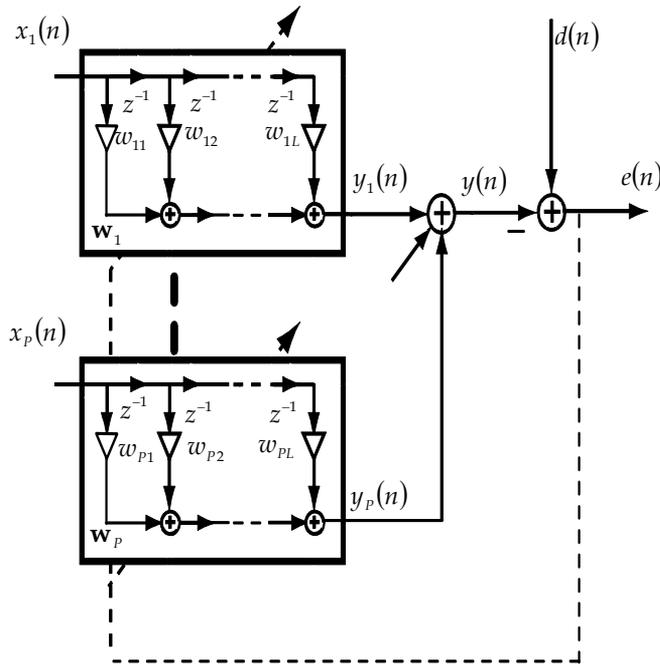


Fig. 3. Multichannel transversal adaptive filtering.

2.1.2 Lattice

The *lattice filter* is an alternative to the *transversal filter* structure for the realization of a *predictor* (Friedlander, 1982).

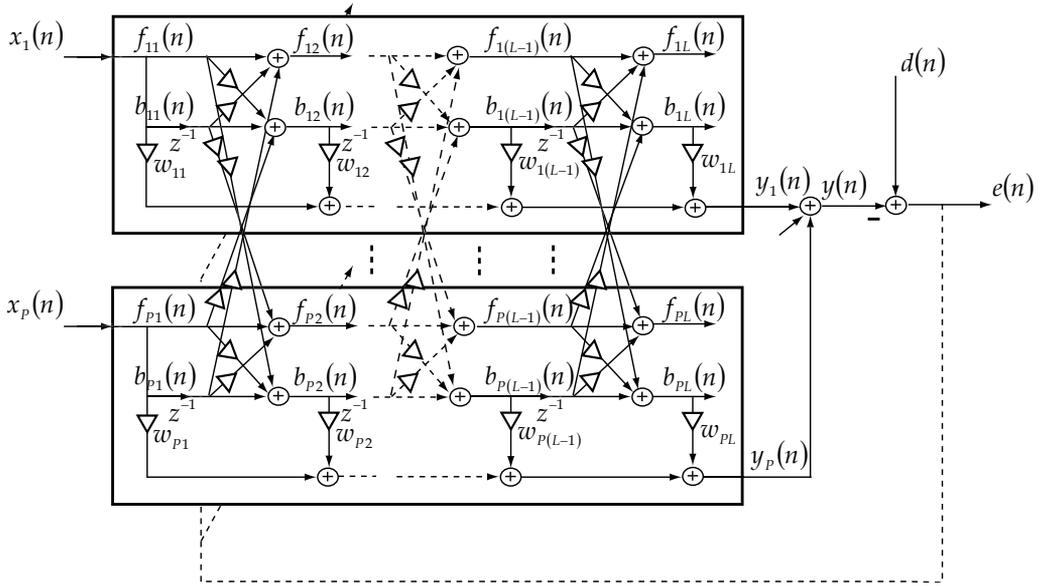


Fig. 4. Multichannel adaptive filtering with lattice-ladder joint-process estimator.

The multichannel version of lattice-ladder structure (Glentis et al., 1999) must consider the interchannel relationship of the reflection coefficients in each stage l .

$$\mathbf{f}_l(n) = \mathbf{f}_{l-1}(n) + \mathbf{K}_l^* \mathbf{b}_{l-1}(n-1), \mathbf{f}_1(n) = \mathbf{x}(n), \quad (9)$$

$$\mathbf{b}_l(n) = \mathbf{b}_{l-1}(n-1) + \mathbf{K}_l \mathbf{f}_{l-1}(n), \mathbf{b}_1(n) = \mathbf{x}(n). \quad (10)$$

Where $\mathbf{f}_l(n) = [f_{1l}(n) \ f_{2l}(n) \ \dots \ f_{pl}(n)]^T$, $\mathbf{b}_l(n) = [b_{1l}(n) \ b_{2l}(n) \ \dots \ b_{pl}(n)]^T$,

$\mathbf{x}(n) = [x_1(n) \ x_2(n) \ \dots \ x_p(n)]^T$, and

$$\mathbf{K}_l = \begin{bmatrix} k_{11l} & k_{12l} & \dots & k_{1pl} \\ k_{21l} & k_{22l} & \dots & k_{2pl} \\ \vdots & \vdots & \ddots & \vdots \\ k_{p1l} & k_{p2l} & \dots & k_{ppl} \end{bmatrix}^T.$$

The *joint-process estimation* of the *lattice-ladder* structure is especially useful for the adaptive filtering because its predictor *diagonalizes* completely the autocorrelation matrix. The transfer function of a lattice filter structure is more complex than a transversal filter because the reflexion coefficients are involved,

$$\mathbf{b}(n) = \mathbf{A}\mathbf{b}(n-1) + \mathbf{K}\mathbf{f}_1(n), \quad (11)$$

$$y(n) = \mathbf{w}\mathbf{A}\mathbf{b}(n-1) + \mathbf{w}\mathbf{K}\mathbf{f}_1(n). \quad (12)$$

Where $\mathbf{w} = [\mathbf{w}_1^T \quad \mathbf{w}_2^T \quad \cdots \quad \mathbf{w}_L^T]^T$ is a $LP \times 1$ vector of the joint-process estimator coefficients, $\mathbf{w}_l = [w_{1l} \quad w_{2l} \quad \cdots \quad w_{pl}]^T$. $\mathbf{b}(n) = [\mathbf{b}_1^T(n) \quad \mathbf{b}_2^T(n) \quad \cdots \quad \mathbf{b}_L^T(n)]^T$ is a $LP \times 1$ backward predictor coefficients vector. \mathbf{A} is a $LP \times LP$ matrix obtained with a recursive development of (9) and (10),

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} & \cdots & \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} \\ \mathbf{I}_{P \times P} & \mathbf{0}_{P \times P} & \cdots & \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} \\ \mathbf{K}_1^* \mathbf{K}_2 & \mathbf{I}_{P \times P} & \cdots & \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} \\ \mathbf{K}_1^* \mathbf{K}_3 & \mathbf{K}_2^* \mathbf{K}_3 & \cdots & \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{K}_1^* \mathbf{K}_{L-3} & \mathbf{K}_2^* \mathbf{K}_{L-3} & \cdots & \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} \\ \mathbf{K}_1^* \mathbf{K}_{L-2} & \mathbf{K}_2^* \mathbf{K}_{L-2} & \cdots & \mathbf{I}_{P \times P} & \mathbf{0}_{P \times P} & \mathbf{0}_{P \times P} \\ \mathbf{K}_1^* \mathbf{K}_{L-1} & \mathbf{K}_2^* \mathbf{K}_{L-1} & \cdots & \mathbf{K}_{L-2}^* \mathbf{K}_{L-1} & \mathbf{I}_{P \times P} & \mathbf{0}_{P \times P} \end{bmatrix}. \quad (13)$$

$\mathbf{I}_{P \times P}$ is a matrix with only ones in the main diagonal and $\mathbf{0}_{P \times P}$ is a $P \times P$ zero matrix.

$\mathbf{K} = [\mathbf{I}_{P \times P} \quad \mathbf{K}_1 \quad \mathbf{K}_2 \quad \cdots \quad \mathbf{K}_{L-1}]^T$ is a $LP \times P$ reflection coefficients matrix.

2.2 Adaptation Algorithms

Once a filter structure has been selected, an *adaptation algorithm* must also be chosen. From control engineering point of view, the speech enhancement is a system identification problem that can be solved by choosing an optimum criteria or cost function $J(\mathbf{w})$ in a *block* or *recursive* approach. Several alternatives are available, and they generally exchange increased complexity for improved performance (speed of adaptation and accuracy of the transfer function after adaption or misalignment defined by $\varepsilon = \|\mathbf{v} - \mathbf{w}\|^2 / \|\mathbf{v}\|^2$).

2.2.1 Cost Functions

Cost functions are related to the statistics of the involved signals and depend on some error signal

$$J(\mathbf{w}) = f\{e(n)\}. \quad (14)$$

The error signal $e(n)$ depends on the specific structure and the adaptive filtering strategy but it is usually some kind of similarity measure between the target signal $s_i(n)$ and the

estimated signal $y_o(n) \approx \hat{s}_i(n)$, for $I = O$. The most habitual cost functions are listed in Table1.

$J(\mathbf{w})$	Comments
$\ e(n)\ ^2$	Mean squared error (MSE). Statistic mean operator
$\frac{1}{N} \sum_{n=1}^{N-1} e^2(n)$	MSE estimator. MSE is normally unknown
$e^2(n)$	Instantaneous squared error
$ e(n) $	Absolute error. Instantaneous module error
$\sum_{n=1}^n \lambda^{n-m} e^2(m)$	Least squares (Weighted sum of the squared error)
$E\{\ \mathbf{f}_l(n)\ ^2 + \ \mathbf{b}_l(n)\ ^2\}$	Mean squared predictor errors (for a lattice structure)

Table 1. Cost functions for adaptive filtering.

2.2.2 Stochastic Estimation

Non-recursive or *block* methods apply batch processing to a transversal filter structure. The input signal is divided into time blocks, and each block is processed independently or with some overlap. This algorithms have *finite memory*.

The use of memory (vectors or matrice blocks) improves the benefits of the adaptive algorithm because they emphasize the variations in the crosscorrelation between the channels. However, this requires a careful structuring of the data, and they also increase the computational exigencies: memory and processing. For channel p , the input signal vector defined in (5) happens to be a matrix of the form

$$\mathbf{X}_p(n) = \begin{bmatrix} \mathbf{x}_p^T(n-N+1) & \mathbf{x}_p^T(n-(N-1)+1) & \cdots & \mathbf{x}_p^T(n) \end{bmatrix}^T, \quad (15)$$

$$\mathbf{X}_p(n) = \begin{bmatrix} x_p(n-N+1) & x_p(n-(N-1)+1) & \cdots & x_p(n) \\ x_p(n-N) & x_p(n-(N-1)) & \cdots & x_p(n-1) \\ \vdots & \vdots & \ddots & \vdots \\ x_p(n-N-L+2) & x_p(n-(N-1)-L+2) & \cdots & x_p(n-L+1) \end{bmatrix},$$

$$\mathbf{d}(n) = [d(n-N+1) \quad d(n-(N-1)+1) \quad \cdots \quad d(n)]^T, \quad (16)$$

where N represents the *memory size*. The input signal matrix to the multichannel adaptive filtering has the form

$$\mathbf{X}(n) = [\mathbf{X}_1^T(n) \quad \mathbf{X}_2^T(n) \quad \cdots \quad \mathbf{X}_p^T(n)]^T. \quad (17)$$

In the most general case (with order memory N), the input signal $\mathbf{X}(n)$ is a matrix of size $LP \times N$. For $N = 1$ (memoryless) and $P = 1$ (single channel) (17) is reduced to (5).

There are adaptive algorithms that use memory $N > 1$ to modify the coefficients of the filter, not only in the direction of the input signal $x(n)$, but within the hyperplane spanned by the $x(n)$ and its $N - 1$ immediate predecessors $[x(n) \ x(n-1) \ \dots \ x(n-N+1)]$ per channel.

The block adaptation algorithm updates its coefficients once every N samples as

$$\mathbf{w}(m+1) = \mathbf{w}(m) + \Delta\mathbf{w}(m), \quad (18)$$

$$\Delta\mathbf{w}(m) = \arg \min J(\mathbf{w}).$$

The matrix defined by (15) stores $K = L + N - 1$ samples per channel. The time index m makes reference to a single update of the weights from time n to $n + N$, based on the K accumulated samples.

The *stochastic recursive* methods, unlike the different optimization *deterministic iterative* algorithms, allow the system to approach the solution with the partial information of the signals using the general rule

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta\mathbf{w}(n), \quad (19)$$

$$\Delta\mathbf{w}(n) = \arg \min J(\mathbf{w}).$$

The new estimator $\mathbf{w}(n+1)$ is updated from the previous estimation $\mathbf{w}(n)$ plus the *adapting-step* or *gradient* obtained from the cost function minimization $J(\mathbf{w})$. These algorithms have an *infinite memory*. The trade-off between convergence speed and the accuracy is intimately tied to the length of memory of the algorithm. The error of the joint-process estimator using a transversal filter with memory can be rewritten like a vector as

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{y}(n) = \mathbf{d}(n) - \mathbf{X}^T(n)\mathbf{w}(n). \quad (20)$$

The unknown system solution, applying the MSE as the cost function, leads to the *normal* or *Wiener-Hopf equation*. The Wiener filter coefficients are obtained by setting the gradient of the square error function to zero, this yields

$$\mathbf{w} = [\mathbf{X}\mathbf{X}^H]^{-1} \mathbf{X}\mathbf{d}^* = \mathbf{R}^{-1}\mathbf{r}. \quad (21)$$

\mathbf{R} is a correlation matrix and \mathbf{r} is a cross-correlation vector defined by

$$\mathbf{R} = \mathbf{X}\mathbf{X}^H = \begin{bmatrix} \mathbf{X}_1\mathbf{X}_1 & \mathbf{X}_1\mathbf{X}_2 & \cdots & \mathbf{X}_1\mathbf{X}_P \\ \mathbf{X}_2\mathbf{X}_1 & \mathbf{X}_2\mathbf{X}_2 & \cdots & \mathbf{X}_2\mathbf{X}_P \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}_P\mathbf{X}_1 & \mathbf{X}_P\mathbf{X}_2 & \cdots & \mathbf{X}_P\mathbf{X}_P \end{bmatrix}, \quad (22)$$

$$\mathbf{r} = \mathbf{X}\mathbf{d}^* = \begin{bmatrix} \mathbf{X}_1\mathbf{d}^* & \mathbf{X}_2\mathbf{d}^* & \cdots & \mathbf{X}_P\mathbf{d}^* \end{bmatrix}^T. \quad (23)$$

For each $i=1\dots I$ input source, $P(P-1)/2$ relations are obtained: $\mathbf{x}_p^H \mathbf{w}_q = \mathbf{x}_q^H \mathbf{w}_p$ for $p, q=1\dots P$, with $p \neq q$. Given vector $\mathbf{u} = \left[\sum_{p=2}^P \mathbf{w}_p^T \quad -\mathbf{w}_1^T \quad \cdots \quad -\mathbf{w}_1^T \right]^T$, due to the nearness with which microphones are placed in scenario of Fig. 1, it is possible to verify that $\mathbf{R}\mathbf{u} = \mathbf{0}_{PL \times 1}$, thus \mathbf{R} is not invertible and no unique problem solution exists. The adaptive algorithm leads to one of many possible solutions which can be very different from the target \mathbf{v} . This is known as a *non-unicity problem*.

For a prediction application, the cross-correlation vector \mathbf{r} must be slightly modified as $\mathbf{r} = \mathbf{X}\mathbf{x}(n-1)$, $\mathbf{x}(n-1) = [x(n-1) \quad x(n-2) \quad \cdots \quad x(n-N)]^T$ and $P=1$.

The optimal Wiener-Hopf solution $\mathbf{w}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{r}$ requires the knowledge of both magnitudes: the correlation matrix \mathbf{R} of the input matrix \mathbf{X} and the cross-correlation vector \mathbf{r} between the input vector and desired answer \mathbf{d} . That is the reason why it has little practical value. So that the linear system given by (21) has solution, the correlation matrix \mathbf{R} must be nonsingular. It is possible to estimate both magnitudes according to the windowing method of the input vector.

The *sliding window* method uses the sample data within a window of finite length N . Correlation matrix and cross-correlation vector are estimated averaging in time,

$$\mathbf{R}(n) = \mathbf{X}(n)\mathbf{X}^H(n)/N, \quad (24)$$

$$\mathbf{r}(n) = \mathbf{X}(n)\mathbf{d}^*(n)/N.$$

The method that estimates the autocorrelation matrix like in (24) with samples organized as in (15) is known as the *covariance method*. The matrix that results is positive semidefinite but it is not Toeplitz.

The *exponential window* method uses a recursive estimation according to certain forgetfulness factor λ in the rank $0 < \lambda < 1$,

$$\mathbf{R}(n) = \lambda\mathbf{R}(n-1) + \mathbf{X}(n)\mathbf{X}^H(n), \quad (25)$$

$$\mathbf{r}(n) = \lambda\mathbf{r}(n-1) + \mathbf{X}(n)\mathbf{d}^*(n).$$

When the excitation signal to the adaptive system is not stationary and the unknown system is time-varying, the exponential and sliding window methods allow the filter to forget or to eliminate errors happened farther in time. The price of this forgetfulness is deterioration in the fidelity of the filter estimation (Gay & Benesty, 2000).

A recursive estimator has the form defined in (19). In each iteration, the update of the estimator is made in the $\Delta \mathbf{w}(n)$ direction. For all the optimization deterministic iterative schemes, a stochastic algorithm approach exists. All it takes is to replace the terms related to the cost function and calculate the approximate values by each new set of input/output samples. In general, most of the adaptive algorithms turn a stochastic optimization problem into a deterministic one and the obtained solution is an approximation to the one of the original problem.

The gradient $\mathbf{g} = \nabla J(\mathbf{w}) = \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{X}\mathbf{d}^* + 2\mathbf{X}\mathbf{X}^H \mathbf{w}$, can be estimated by means of $\mathbf{g} = -2(\mathbf{r} + \mathbf{R}\mathbf{w})$, or by the equivalent one $\mathbf{g} = -\mathbf{X}\mathbf{e}^*$, considering \mathbf{R} and \mathbf{r} according to (24) or (25). It is possible to define recursive updating strategies, per each l stage, for lattice structures as

$$\mathbf{K}_l(n+1) = \mathbf{K}_l(n) + \Delta \mathbf{K}_l(n), \quad (26)$$

$$\Delta \mathbf{K}_l(n) = \arg \min J(\mathbf{K}_l).$$

2.2.3 Optimization strategies

Several strategies to solve $\Delta \mathbf{w} = \arg \min J(\mathbf{w})$ are proposed (Glentis et al., 1999) (usually of the least square type). It is possible to use a quadratic (second order) approximation of the error-performance surface around the current point denoted $\mathbf{w}(n)$. Recalling the second-order *Taylor series* expansion of the cost function $J(\mathbf{w})$ around $\mathbf{w}(n)$, with $\Delta \mathbf{w} = \mathbf{w} - \mathbf{w}(n)$, you have

$$J(\mathbf{w} + \Delta \mathbf{w}) \cong J(\mathbf{w}) + \Delta \mathbf{w}^H \nabla J(\mathbf{w}) + \frac{1}{2} \Delta \mathbf{w}^H \nabla^2 J(\mathbf{w}) \Delta \mathbf{w} \quad (27)$$

Deterministic iterative optimization schemes require the knowledge of the cost function, the *gradient* (first derivatives) defined in (29) or the *Hessian* matrix (second order partial derivatives) defined in (45,52) while *stochastic recursive* methods replace these functions by impartial estimations.

$$\nabla J(\mathbf{w}) = \left[\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_1} \quad \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_2} \quad \dots \quad \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_L} \right]^T, \quad (28)$$

$$\nabla^2 J(\mathbf{w}) = \begin{bmatrix} \frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}_1 \partial \mathbf{w}_1} & \frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}_1 \partial \mathbf{w}_2} & \dots & \frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}_1 \partial \mathbf{w}_L} \\ \frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}_2 \partial \mathbf{w}_1} & \frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}_2 \partial \mathbf{w}_2} & \dots & \frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}_2 \partial \mathbf{w}_L} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}_L \partial \mathbf{w}_1} & \frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}_L \partial \mathbf{w}_2} & \dots & \frac{\partial^2 J(\mathbf{w})}{\partial \mathbf{w}_L \partial \mathbf{w}_L} \end{bmatrix}^T. \quad (29)$$

The vector $\mathbf{g}(n) = \nabla J(\mathbf{w})$ is the *gradient* evaluated at $\mathbf{w}(n)$, and the matrix $\mathbf{H}(n) = \nabla^2 J(\mathbf{w})$ is the *Hessian* of the cost function evaluated at $\mathbf{w}(n)$.

Several first order adaptation strategies are: to choose a starting initial point $\mathbf{w}(0)$, to increment election $\Delta \mathbf{w}(n) = \mu(n)\mathbf{g}(n)$; two decisions are due to take: movement direction $\mathbf{g}(n)$ in which the cost function decreases fastest and the step-size in that direction $\mu(n)$. The iteration stops when a certain level of error is reached $\Delta \mathbf{w}(n) < \xi$,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n)\mathbf{g}(n). \quad (30)$$

Both parameters $\mu(n)$, $\mathbf{g}(n)$ are determined by a cost function. The second order methods generate values close to the solution in a minimum number of steps but, unlike the first order methods, the second order derivatives are very expensive computationally. The adaptive filters and its performance are characterized by a selection criteria of $\mu(n)$ and $\mathbf{g}(n)$ parameters.

Method	Definition	Comments
SD	$\mu(n) = -\frac{\ \mathbf{g}\ ^2}{\mathbf{g}^H \mathbf{R} \mathbf{g}}$	Steepest-Descent
CG	(See below)	Conjugate Gradient
NR	$\mu(n) = \alpha \mathbf{Q}$	Newton-Raphson

Table 2. Optimization methods.

The *optimization methods* are useful to find the minimum or maximum of a quadratic function. Table 2 summarizes the optimization methods. SD is an iterative optimization procedure of easy implementation and computationally very cheap. It is recommended with cost functions that have only one minimum and whose gradients are isotropic in magnitude respect to any direction far from this minimum. NR method increases SD performance using a carefully selected weighting matrix. The simplest form of NR uses $\mathbf{Q} = \mathbf{R}^{-1}$. *Quasy-Newton*

methods (QN) are a special case of NR with \mathbf{Q} simplified to a constant matrix. The solution to $J(\mathbf{w})$ is also the solution to the *normal equation* (21). The *conjugate gradient* (CG) (Boray & Srinath, 1992) was designed originally for the minimization of convex quadratic functions but, with some variations, it has been extended to the general case. The first CG iteration is the same that the SD algorithm and the new successive directions are selected in such a way that they form a set of vectors mutually conjugated to the Hessian matrix (corresponding to the autocorrelation matrix, \mathbf{R}), $\mathbf{q}_i^H \mathbf{R} \mathbf{q}_j = 0, \forall i \neq j$. In general, CG methods have the form

$$\mathbf{q}_l = \begin{cases} -\mathbf{g}_l, & l = 1 \\ -\mathbf{g}_l + \beta_l \mathbf{q}_{l-1}, & l > 1 \end{cases} \quad (31)$$

$$\mu_l = \frac{\langle \mathbf{g}_l, \mathbf{q}_l \rangle}{\langle \mathbf{q}_l, \mathbf{g}_l - \mathbf{p}_l \rangle}, \quad (32)$$

$$\beta_l = \frac{\|\mathbf{g}_l\|^2}{\|\mathbf{g}_{l-1}\|^2}, \quad (33)$$

$$\mathbf{w}_{l+1}(n) = \mathbf{w}_l(n) + \mu_l(n) \mathbf{q}_l. \quad (34)$$

CG spans the search directions from the gradient in course, \mathbf{g} , and a combination of previous \mathbf{R} -conjugated search directions. β guarantees the \mathbf{R} -conjugation. Several methods can be used to obtain β . This method (33) is known as Fletcher-Reeves. The gradients can be obtained as $\mathbf{g} = \nabla J(\mathbf{w})$ and $\mathbf{p} = \nabla J(\mathbf{w} - \mathbf{g})$.

The *memoryless* LS methods in Table 3 use the instantaneous squared error cost function $J(\mathbf{w}) = e^2(n)$. The descent direction for all is a gradient $\mathbf{g}(n) = \mathbf{x}(n)e^*(n)$. The LMS algorithm is a stochastic version of the SD optimization method. NLMS frees the convergence speed of the algorithm with the power signal. FNLMS filters the signal power estimation; $0 < \beta < 1$ is a weighting factor. PNLMS adaptively controls the size of each weight.

Method	Definition	Comments
LMS	$\mu(n) = \alpha$	Least Means Squares
NLMS	$\mu(n) = \frac{\alpha}{\ \mathbf{x}(n)\ ^2 + \delta}$	Normalized LMS
FNLMS	$\mu(n) = \frac{\alpha}{\mathbf{p}(n)}$	Filtered NLMS
PNLMS	$\mu(n) = \frac{\alpha \mathbf{Q}}{\mathbf{x}^H(n) \mathbf{Q} \mathbf{x}(n) + \delta}$	Proportionate NLMS

Table 3. Memoryless Least-Squares (LS) methods.

Method	Definition	Comments
RLS	$\mu(n) = \mathbf{R}^{-1}(n)$ $\mathbf{g}(n) = \mathbf{x}(n)e^*(n)$	Recursive Least-Squares
LMS-SW	$\mu(n) = \frac{\ \mathbf{g}(n)\ ^2}{\mathbf{g}^H(n)\mathbf{X}(n)\mathbf{X}^H(n)\mathbf{g}(n) + \delta}$ $\mathbf{g}(n) = \mathbf{X}(n)e^*(n)$	Sliding-Window LMS
APA	$\mu(n) = \frac{\alpha}{\mathbf{X}(n)\mathbf{X}^H(n) + \delta\mathbf{I}}$ $\mathbf{g}(n) = \mathbf{X}(n)e^*(n)$	Affine Projection Algorithm
PRA	$\mathbf{w}(n+1) = \mathbf{w}(n-N+1) + \mu(n)\mathbf{g}(n)$ $\mu(n) = \frac{\alpha}{\mathbf{X}(n)\mathbf{X}^H(n) + \delta\mathbf{I}}$ $\mathbf{g}(n) = \mathbf{X}(n)e^*(n)$	Partial Rank Algorithm
DLMS	$\mu(n) = \frac{1}{\langle \mathbf{x}(n), \mathbf{z}(n) \rangle}$ $\mathbf{g}(n) = \mathbf{z}(n)e^*(n)$ $\mathbf{z}(n) = \mathbf{x}(n) + \frac{\langle \mathbf{x}(n), \mathbf{x}(n-1) \rangle}{\ \mathbf{x}(n-1)\ ^2} \mathbf{x}(n-1)$	Decorrelated LMS
TDLMS	$\mu(n) = \frac{\alpha\mathbf{Q}}{\ \mathbf{x}(n)\ ^2}, \mathbf{Q}\mathbf{Q}^H = \mathbf{I}$ $\mathbf{g}(n) = \mathbf{x}(n)e^*(n)$	Transform-Domain DLMS

Table 4. Least-Squares with memory methods.

\mathbf{Q} is a diagonal matrix that weights the individual coefficients of the filters, α is a *relaxation constant* and δ guarantees that the denominator never becomes zero. These algorithms are very cheap computationally but their convergence speed depends strongly on the *spectral condition number* of the autocorrelation matrix \mathbf{R} (that relate the extreme eigenvalues) and can get to be unacceptable as the correlation between the P channels increases.

The *projection algorithms* in Table 4 modify the filters coefficients in the input vector direction and on the subspace spanned by the $N-1$ redecessors. RLS is a recursive solution to the normal equation that uses MSE as cost function. There is an alternative fast version FRLS. LMS-SW is a variant of SD that considers a data window. The step can be obtained by a linear search. APA is a generalization of RLS and NLMS. APA is obtained by projecting the adaptive coefficients vector \mathbf{w} in the *affine subspace*. The affine subspace is obtained by means of a translation from the orthogonal origin to the subspace where the vector \mathbf{w} is projected. PRA is a strategy to reduce the computational complexity of APA by updating the coefficients every N samples. DLMS replaces the system input by an orthogonal component to the last input (order 2). These changes the updating vector direction of the correlated input signals so that these ones correspond to uncorrelated input signals. TDLMS decorrelates into transform domain by means of a \mathbf{Q} matrix.

The adaptation of the transversal section of the joint-process estimator in the lattice-ladder structure depends on the gradient $\mathbf{g}(n)$ and, indirectly, on the reflection coefficients, through the backward predictor, $\mathbf{g}(n) = \mathbf{b}(n)$. However, the reflection coefficient adaptation depends on the gradient of $y(n)$ with respect to them

$$\nabla J(\mathbf{K}) = \left[\begin{array}{cccc} \frac{\partial J(\mathbf{K})}{\partial \mathbf{K}_1} & \frac{\partial J(\mathbf{K})}{\partial \mathbf{K}_2} & \dots & \frac{\partial J(\mathbf{K})}{\partial \mathbf{K}_L} \end{array} \right]^T, \quad (35)$$

$$\nabla^2 J(\mathbf{K}) = \left[\begin{array}{cccc} \frac{\partial^2 J(\mathbf{K})}{\partial \mathbf{K}_1 \partial \mathbf{K}_1} & \frac{\partial^2 J(\mathbf{K})}{\partial \mathbf{K}_1 \partial \mathbf{K}_2} & \dots & \frac{\partial^2 J(\mathbf{K})}{\partial \mathbf{K}_1 \partial \mathbf{K}_L} \\ \frac{\partial^2 J(\mathbf{K})}{\partial \mathbf{K}_2 \partial \mathbf{K}_1} & \frac{\partial^2 J(\mathbf{K})}{\partial \mathbf{K}_2 \partial \mathbf{K}_2} & \dots & \frac{\partial^2 J(\mathbf{K})}{\partial \mathbf{K}_2 \partial \mathbf{K}_L} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J(\mathbf{K})}{\partial \mathbf{K}_L \partial \mathbf{K}_1} & \frac{\partial^2 J(\mathbf{K})}{\partial \mathbf{K}_L \partial \mathbf{K}_2} & \dots & \frac{\partial^2 J(\mathbf{K})}{\partial \mathbf{K}_L \partial \mathbf{K}_L} \end{array} \right]. \quad (36)$$

In a more general case, concerning to a multichannel case, the gradient matrix can be obtained as $\mathbf{G} = \nabla J(\mathbf{K})$. Two recursive updatings are necessary

$$\mathbf{w}_l(n+1) = \mathbf{w}_l(n) + \mu_l(n) \mathbf{g}_l(n), \quad (37)$$

$$\mathbf{K}_l(n+1) = \mathbf{K}_l(n) + \lambda_l(n) \mathbf{G}_l(n) \quad (38)$$

Table 5 resumes the least-squares for lattice.

GAL is a NLMS extension for a lattice structure that uses two cost functions: instantaneous squared error for the transversal part and prediction MSE for the lattice-ladder part,

$$\mathbf{B}_l(n) = \beta \mathbf{B}_l(n-1) + (1-\beta) \left(|\mathbf{f}_l(n)|^2 + |\mathbf{b}_l(n-1)|^2 \right), \text{ where } \alpha \text{ and } \sigma \text{ are relaxation factors.}$$

Method	Definition	Comments
GAL	$\mu_l(n) = \frac{\alpha}{\ \mathbf{b}_l(n)\ ^2}$ $\mathbf{g}_l(n) = \mathbf{b}_l(n)e^*(n)$ $\lambda_l(n) = \frac{\sigma}{\mathbf{B}_{l-1}(n)}$ $\mathbf{G}_l(n) = \mathbf{b}_{l-1}(n-1)\mathbf{f}_l^H(n) + \mathbf{b}_l(n)\mathbf{f}_{l-1}^H(n-1)$	Gradient Adaptive Lattice
CGAL	(See below)	CG Adaptive Lattice

Table 5. Least-Squares for lattice.

For CGAL, the same algorithm described in (31-34) is used but it is necessary to rearrange the gradient matrices of the lattice system in a column vector. It is possible to arrange the gradients of all lattice structures in matrices. $\mathbf{U}(n) = [\mathbf{g}_1^T(n) \ \mathbf{g}_2^T(n) \ \cdots \ \mathbf{g}_P^T(n)]^T$ is the $P \times L$ gradient matrix with respect to the transversal coefficients, $\mathbf{g}_p(n) = [g_{p1} \ g_{p2} \ \cdots \ g_{pL}]^T$, $p = 1 \dots P$. $\mathbf{V}(n) = [\mathbf{G}_1(n) \ \mathbf{G}_2(n) \ \cdots \ \mathbf{G}_P(n)]^T$ is a $P \times (L-1)P$ gradient matrix with respect to the reflection coefficients; and rearranging these matrices in one single column vector, $[\mathbf{u}^T \ \mathbf{v}^T]^T$ is obtained with

$$\mathbf{u} = [g_{11} \ \cdots \ g_{1L} \ g_{21} \ \cdots \ g_{2L} \ \cdots \ g_{P1} \ \cdots \ g_{PL}]^T,$$

$$\mathbf{v} = [G_{111} \ \cdots \ G_{1P1} \ \cdots \ G_{P11} \ \cdots \ G_{PP1} \ G_{112} \ \cdots \ G_{PP(L-1)}]^T.$$

$$\mathbf{q}_l = \begin{cases} -\mathbf{g}_l, & l = 1 \\ -\mathbf{g}_l + \beta_l \mathbf{q}_{l-1}, & l > 1 \end{cases} \quad (39)$$

$$\mathbf{g}_l = \begin{cases} [\mathbf{u}^T \ \mathbf{v}^T]^T, & l = 1 \\ \alpha \mathbf{g}_{l-1} + (1-\alpha)[\mathbf{u}^T \ \mathbf{v}^T]^T, & l > 1 \end{cases} \quad (40)$$

$$\beta_l = \frac{\|\mathbf{g}_l\|^2}{\|\mathbf{g}_{l-1}\|^2}, \quad (41)$$

$$\mathbf{w}_{l+1} = \mathbf{w}_l + \mu \mathbf{u}_l, \quad (42)$$

$$\mathbf{K}_{l+1} = \mathbf{K}_l + \lambda_l \mathbf{V}_l. \quad (43)$$

The time index n has been removed by simplicity. $0 < \alpha < 1$ is a forgetfulness factor which weights the innovation importance specified in a low-pass filtering in (40). The gradient selection is very important. A mean value that uses more recent coefficients is needed for gradient estimation and to generate a vector with more than one conjugate direction (40).

3. Multirate Adaptive Filtering

The adaptive filters used for speech enhancement are probably very large (due to the AIRs). Multirate adaptive filtering works at a lower sampling rate that allows reducing the complexity (Shynk, 1992). Depending on how the data and filters are organized, these approaches may upgrade in performance and avoid end-to-end delay. Multirate schemes adapt the filters in smaller sections at lower computational cost. This is only necessary for real-time implementations. Two approaches are considered. The *subband adaptive filtering* approach splits the spectra of the signal in a number of subbands that can be adapted independently and afterwards the filtering can be carried out in a fullband. The *frequency-domain adaptive filtering* partitions the signal in time-domain and projects it into a transformed domain (i.e. frequency) using better properties for adaptive processing. In both cases the input signals are transformed into a more desirable form before adaptive processing and the adaptive algorithms operate in transformed domains, whose basis functions orthogonalize the input signal, speeding up the convergence. The *partitioned convolution* is necessary for fullband delayless convolution and can be seen as an efficient frequency-domain convolution.

3.1 Subband Adaptive Filtering

The fundamental structure for subband adaptive filtering is obtained using band-pass filters as basis functions and replacing the fixed gains for adaptive filters. Several implementations are possible. A typical configuration uses an *analysis filter bank*, a processing stage and a *synthesis filter bank*. Unfortunately, this approach introduces an end-to-end delay due to the synthesis filter bank. Figure 5 shows an alternative structure which adapts in subbands and filters in full-band to remove this delay (Reilly et al., 2002).

K is the *decimation ratio*, M is the number of bands and N is the prototype filter length. k is the low rate time index. The sample rate in subbands is reduced to F_s/K . The input signal per channel is represented by a vector $\mathbf{x}_p(n) = [x(n) \ x(n-1) \ \dots \ x(n-L+1)]^T$, $p=1\dots P$. The adaptive filter in full-band per channel $\mathbf{w}_p = [w_{p1} \ w_{p2} \ \dots \ w_{pL}]^T$ is obtained by means of the \mathbf{T} operator as

$$\mathbf{w}_p = \Re \left\{ \sum_{m=1}^{M/2} \left(\mathbf{h}_{m \downarrow K} * \mathbf{w}_{pm} \right) \uparrow_K * \mathbf{g}_m \right\}, \quad (44)$$

from the subband adaptive filters per each channel \mathbf{w}_{pm} , $p=1\dots P$, $m=1\dots M/2$ (Reilly et al., 2002). The subband filters are very short, of length $C = \left\lceil \frac{L+N-1}{K} \right\rceil - \left\lfloor \frac{N}{K} \right\rfloor + 1$, which

allows to use much more complex algorithms. Although the input signal vector per channel $\mathbf{x}_p(n)$ has size $L \times 1$, it acts as a delay line which, for each iteration k , updates K samples.

$\downarrow K$ is an operator that means downsampling for a K factor and $\uparrow K$ upsampling for a K factor. \mathbf{g}_m is a synthesis filter in subband m obtained by modulating a prototype filter. \mathbf{H} is a polyphase matrix of a generalized discrete Fourier transform (GDFT) of an oversampled ($K < M$) analysis filter bank (Crochiere & Rabiner, 1983). This is an efficient implementation of a uniform complex modulated analysis filter bank. This way, only a prototype filter \mathbf{p} is necessary; the prototype filter is a low-pass filter.

The band-pass filters are obtained modulating a prototype filter. It is possible to select different adaptive algorithms or parameter sets for each subband. For delayless implementation, the full-band convolution may be made by a *partitioned convolution*.

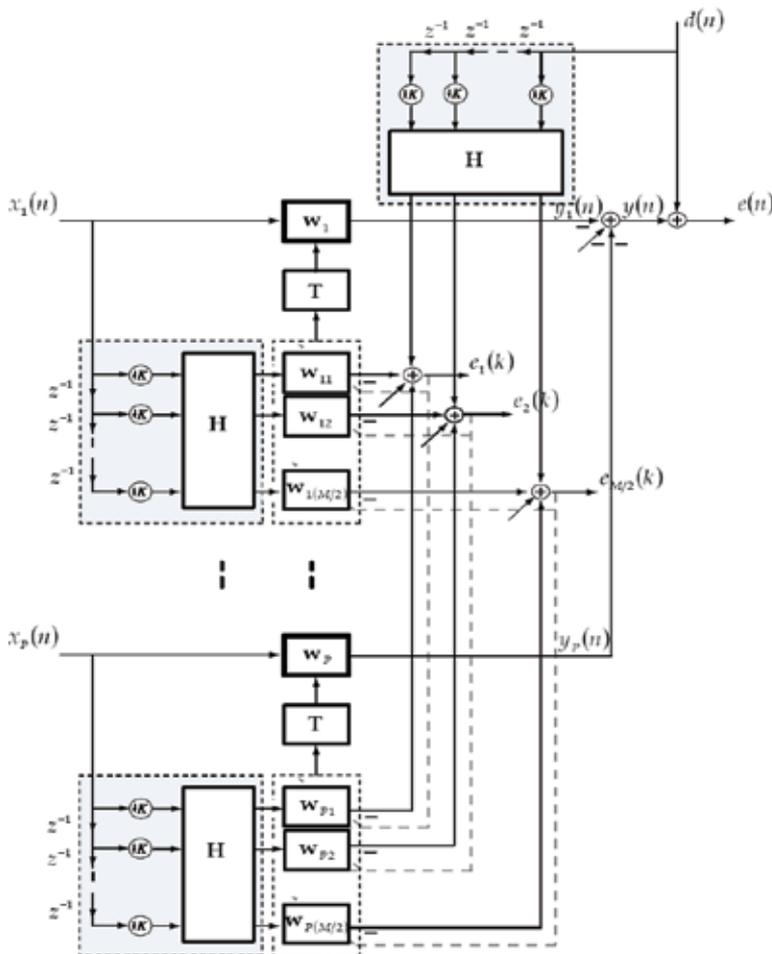


Fig. 5. Subband adaptive filtering. This configuration is known as *open-loop* because the error is in the time-domain. An alternative *closed-loop* can be used where the error is in the subband-domain. Gray boxes correspond to efficient polyphase implementations. See details in (Reilly et al., 2002).

3.2 Frequency Domain Adaptive Filtering

The basic operation in *frequency-domain adaptive filtering* (FDAF) is to transform the input signal in a “more desirable” form before the adaptation process starts (Shynk, 1992) in order to work with matrix multiplications instead of dealing with slow convolutions.

The frequency-domain transform employs one or more *discrete Fourier transforms* (DFT), T operator in Fig. 6, and can be seen as a pre-processing block that generates decorrelated output signals. In the more general FDAF case, the output of the filter in the time-domain (3) can be seen as the direct frequency-domain translation of the block LMS (BLMS) algorithm. That efficiency is obtained taking advantage of the equivalence between the linear convolution and the circular convolution (multiplication in the frequency-domain).

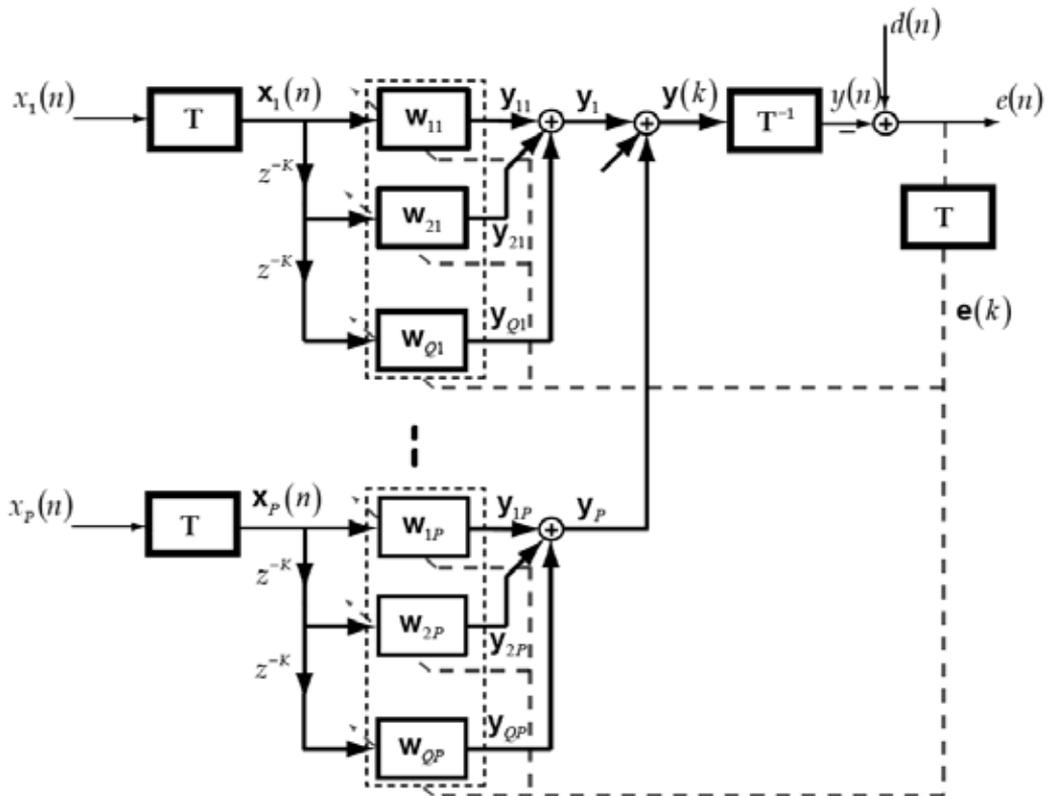


Fig. 6. Partitioned block frequency-domain adaptive filtering.

It is possible to obtain the linear convolution between a finite length sequence (filter) and an infinite length sequence (input signal) with the overlapping of certain elements of the data sequence and the retention of only a subgroup of the DFT.

The *partitioned block frequency-domain adaptive filtering* (PBFDAF) was developed to deal efficiently with such situations (Paez & Otero, 1992). The PBFDAF is a more efficient implementation of the LMS algorithm in the frequency-domain. It reduces the computational burden and bounds the user-delay. In general, the PBFDAF is widely used due to its good trade-off between speed, computational complexity and overall latency.

However, when working with long AIRs, the convergence properties provided by the algorithm may not be enough. This technique makes a sequential partition of the impulse response in the time-domain prior to a frequency-domain implementation of the filtering operation.

This time segmentation allows setting up individual coefficient updating strategies concerning different sections of the adaptive canceller, thus avoiding the need to disable the adaptation in the complete filter. In the PBFDAF case, the filter is partitioned transversally in an equivalent structure. Partitioning \mathbf{w}_p in Q segments (K length) we obtain

$$y(n) = \sum_{p=1}^P \sum_{q=1}^Q \sum_{m=0}^{K-1} x_p(n - qK - m) w_{p(qK+m)} , \quad (45)$$

Where the total filter length L , for each channel, is a multiple of the length of each segment $L = QK$, $K \leq L$. Thus, using the appropriate data sectioning procedure, the Q linear convolutions (per channel) of the filter can be independently carried out in the frequency-domain with a total delay of K samples instead of the QK samples needed by standard FDAF implementations. Figure 6 shows the block diagram of the algorithm using the overlap-save method. In the frequency-domain with matricial notation, (45) can be expressed as

$$\mathbf{Y} = \mathbf{X} \otimes \mathbf{W} , \quad (46)$$

where $\mathbf{X} = \mathbf{F}\mathbf{x}$ represents a matrix of dimensions $M \times Q \times P$ which contains the Fourier transform of the Q partitions and P channels of the input signal matrix \mathbf{x} . \mathbf{F} represents the DFT matrix defined as $\mathbf{F} = W_M^{-mn}$ of size $M \times M$ and \mathbf{F}^{-1} as its inverse. Of course, in the final implementation, the DFT matrix should be substituted by much more efficient *fast Fourier transform* (FFT). Being \mathbf{X} , $2K \times P$ -dimensional (supposing 50% overlapping between the new block and the previous one). It should be taken into account that the algorithm adapts every K samples. \mathbf{W} represents the filter coefficient matrix adapted in the frequency-domain (also $M \times Q \times P$ -dimensional) while the \otimes operator multiplies each of the elements one by one; which, in (46), represents a *circular convolution*. The output vector \mathbf{y} can be obtained as the double sum (rows) of the \mathbf{Y} matrix. First we obtain a $M \times P$ matrix which contains the output of each channel in the frequency-domain \mathbf{y}_p , $p = 1 \dots P$, and secondly, adding all the outputs we obtain the whole system output, \mathbf{y} . Finally, the output in the time-domain is obtained by using $\mathbf{y} = \text{last } K \text{ components of } \mathbf{F}^{-1}\mathbf{y}$. Notice that the sums are performed prior to the time-domain translation. This way we reduce $(P-1)(Q-1)$ FFTs in the complete filtering process. As in any adaptive system the error can be obtained as

$$\mathbf{e} = \mathbf{d} - \mathbf{y} \quad (47)$$

with $\mathbf{d} = [d(mK) \ d(mK+1) \ \dots \ d((m+1)K-1)]^T$. The error in the frequency-domain (for the actualization of the filter coefficients) can be obtained as

$$\mathbf{e} = \mathbf{F} \begin{bmatrix} \mathbf{0}_{K \times 1} \\ \mathbf{e} \end{bmatrix}. \quad (48)$$

As we can see, a block of K zeros is added to ensure a correct linear convolution implementation. In the same way, for the block gradient estimation, it is necessary to employ the same error vector in the frequency-domain for each partition q and channel p . This can be achieved by generating an error matrix \mathbf{E} with dimensions $M \times Q \times P$ which contains replicas of the error vector, defined in (48), of dimensions P and Q ($\mathbf{E} \leftarrow \mathbf{e}$ in the notation). The actualization of the weights is performed as

$$\mathbf{W}(m+1) = \mathbf{W}(m) + \mu(m) \mathbf{G}(m). \quad (49)$$

The instantaneous gradient is estimated as

$$\mathbf{G} = -\mathbf{X}' \otimes \mathbf{E}. \quad (50)$$

This is the unconstrained version of the algorithm which saves two FFTs from the computational burden at the cost of decreasing the convergence speed. The constrained version basically makes a gradient projection. The gradient matrix is transformed into the time-domain and is transformed back into the frequency-domain using only the first K elements of \mathbf{G} as

$$\mathbf{G} = \mathbf{F} \begin{bmatrix} \mathbf{G} \\ \mathbf{0}_{K \times Q \times P} \end{bmatrix}. \quad (51)$$

A conjugate gradient version of PBFDAF is possible by transforming the gradient matrix to vectors and reverse (García, 2006). The vectors \mathbf{g} and \mathbf{p} in (31,32) should be changed by $\mathbf{g}_l \leftarrow \bar{\mathbf{G}}_l$, $\bar{\mathbf{G}}_l = \nabla J(\mathbf{W}_l)$ and $\mathbf{p}_l \leftarrow \bar{\mathbf{P}}_l$, $\bar{\mathbf{P}}_l = \nabla J(\mathbf{W}_l - \bar{\mathbf{G}}_l)$, with gradient estimation obtained by averaging the instantaneous gradient estimates over N past values

$$\bar{\mathbf{G}}_l = \nabla J(\mathbf{W}_l) = \frac{2}{N} \sum_{k=1}^N \mathbf{G}_{l-k} \Big|_{\mathbf{W}_l, \mathbf{X}_{l-k}, \mathbf{d}_{l-k}}.$$

3.3 Partitioned Convolution

For each input i , the AIR matrix, \mathbf{V} , is reorganized in a column vector $\mathbf{v} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_p]^T$ of size $N = LP \times 1$ and initially partitioned in a reasonable number Q of equally-sized blocks \mathbf{v}_q , $q = 1 \dots Q$, of length K . Each of these blocks is treated as a

separate impulse response, and convolved by a standard overlap-and-save process, using T operator (FFT windows of length L). All input data are processed in overlapped blocks of L samples (each block at $L - K$ samples to the last). Each block is zero-padded to length L (typically equal to $2K$), and transformed with FFT so that a collection of Q frequency-domain filters v_q is obtained. The results of the multiplications of these Q filters v_q with the FFTs of the Q input blocks are summed, producing the same result as the unpartitioned convolution, by means of proper delays applied to the blocks of convolved data. Finally an T^{-1} operator (IFFT) of the first accumulator is made to submit an output data block (obviously only the last $L - K$ block samples). Each block of input data needs to be FFT transformed just once, and thus the number of forward FFTs is minimized (Armelloni et al., 2003). The main advantage compared to unpartitioned convolution is that the latency of the whole filtering processing is just M points instead of $2N$, and thus the I/O delay is kept to a low value, provided that the impulse response is partitioned in a sensible number of chunks (8-32). Figure 7 outlines the whole process.

Suppose that $A = (A_1, A_2, \dots, A_Q)$ is a set of multiplications of the first data block and $B = (B_1, B_2, \dots, B_Q)$ the second, then for time-index 1 it is only necessary to consider A_1 . At the next index-time, corresponding to $K+1$ samples, the sum is formed with $(B_Q, B_1 + A_2, B_2 + A_3, \dots, B_{Q-1} + A_Q)$. If $C = (C_1, C_2, \dots, C_Q)$ corresponds to the third block the sum is formed with $(C_Q, C_1 + B_2 + A_3, C_2 + B_3 + A_4, \dots, C_{Q-1} + B_Q)$. An efficient implementation of this sum can be implemented using a double buffering technique (Armelloni et al., 2003).

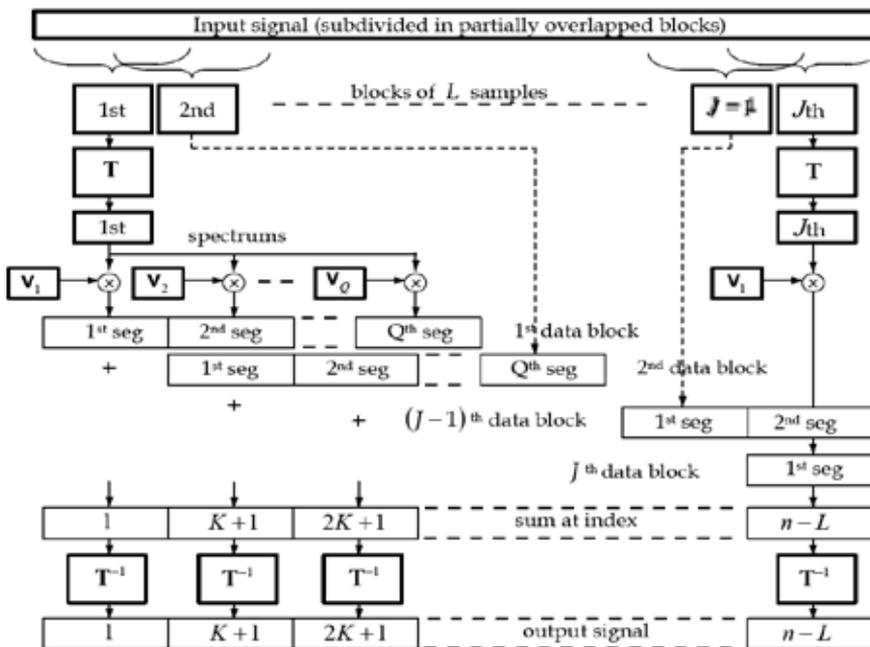


Fig. 7. Partitioned convolution. Each output signal block is produced taking only the $L - K$ last samples of the block.

3.4 Delayless Approach for Real-Time Applications

The filtering operation can be made *delayless* by operating the first block in the time-domain (direct convolution) while the rest of the blocks continue to operate in the frequency domain (Morgan & Thi, 1995). The fast convolution starts after the samples have been processed for direct convolution. The direct convolution allows giving samples to the output while data is incoming. This approach is applicable to the multirate frameworks described.

4. Applications

Once the theoretical foundations of the adaptive filtering have been reviewed, the most important techniques that can be applied to speech enhancement are introduced.

4.1 Spectral Equalization

The adaptive spectral equalization is widely used for noise suppression and corresponds to the single-input and single-output (SISO) *estimator* application (class a, Fig. 2); a single microphone, $P = 1$, is employed. This approach estimates a *noiseprint spectra* and subtracts it from the whole signal in the frequency-domain. The Wiener filter estimator is the result of

estimating $y(n)$ from $s(n)$ that minimizes the MSE $\|y(n) - s(n)\|^2$ given by $\mathbf{y} = \mathbf{Q}\mathbf{x}$, $\mathbf{x} = \mathbf{s} + \mathbf{r}$, and that results.

$$\mathbf{q} \cong \frac{|\mathbf{x}|^2 - |\mathbf{d}|^2}{|\mathbf{x}|^2}, \quad (52)$$

$\mathbf{Q} = \text{diag}\{[\mathbf{q}_1 \ \mathbf{q}_2 \ \dots \ \mathbf{q}_M]\}$ is a diagonal matrix which contains the spectral gain in the frequency-domain; normally \mathbf{T} is a short-time Fourier transform (STFT), suitable for not stationary signals, and \mathbf{T}^{-1} its inverse. In this case this algorithm is known as short-time spectral attenuation (STSA). The $M \times 1$ vector \mathbf{q} contains the main diagonal components of \mathbf{Q} . \mathbf{d} is the noise spectrum (normally unknown). In this case an estimation of the noiseprint spectra $\mathbf{d} = \hat{\mathbf{r}}$ from the mixture \mathbf{x} (noisy signal) is necessary (in intervals when the speech is absent and only the noise is present). The STFT is defined as

$\mathbf{x}_k = \sum_{n=1}^N h(n)x(m-n)W_M^{-mk}$, $m=0..M-1$, where k is the time index about which the short-time spectrum is computed, m is the discrete frequency index, $h(n)$ is an analysis window, N dictates the duration over which the transform is computed, and M is the number of frequency bins at which the STFT is computed.

For stationary signals the squared-magnitude of the STFT provides a sample estimate of the power spectrum of the underlying random process. This form (53) is basic to nearly all the noise reduction methods investigated over last forty years (Gay & Benesty, 2000). The specific form to obtain \mathbf{Q} is known as the *suppression rule*.

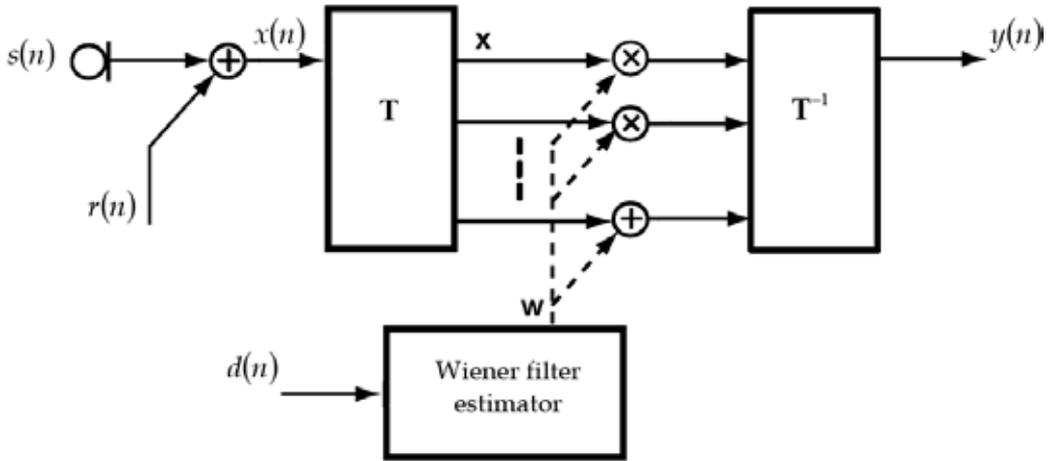


Fig. 8. Spectral equalization.

Power Subtraction

An alternative estimate from Wiener's theory is achieved assuming that \mathbf{s} can be estimated if its magnitude is estimated as

$$\hat{\mathbf{s}} = \sqrt[\alpha]{|\mathbf{x}|^\alpha - \beta |\mathbf{d}|^\alpha}, \quad (53)$$

and the phase of the noisy signal \mathbf{x} can be used, if its SNR is reasonably high, in place of the phase of \mathbf{s} . α is an exponent and β is a parameter introduced to control the amount of noise to be subtracted ($\beta=1$ for full subtraction and $\beta>1$ for over subtraction). A paramount issue in spectral subtraction is to obtain a good noise estimate; its accuracy greatly affects the noise reduction performance (Benesty & Huang, 2003).

4.2 Linear Prediction

The *adaptive linear prediction* (ALP) is employed in an attempt to separate the deterministic $y(n) \approx \hat{s}(n)$ and stochastic part $e(n) \approx \hat{r}(n)$ assuming that the noise and interference signal has a broadband spectra. ALP corresponds to single-input and single-output (SISO) *predictor* application (class b, Fig. 2) with a single microphone, $P=1$.

Most signals, such as speech and music, are partially predictable and partially random. The random input models the unpredictable part of the signal, whereas the filter models the predictable structure of the signal. The aim of linear prediction is to model the mechanism that introduces the correlation in a signal (Vaseghi, 1996). The solution to this system corresponds to a Wiener solution (21) with the cross-correlation vector, \mathbf{r} , slightly modified. The delay z^{-D} in the ALP filter should be selected in such a way that $d(n) = x(n) + r(n)$ and $d(n-D)$ are still correlated. If D is too long, the correlation in $d(n)$ and $d(n-D)$ is weak and unpredictable for the ALP filter; for that reason it cannot be canceled suitably. If D is

too short, the deterministic part of signal in $d(n)$ and $d(n-D)$ remains correlated after D ; for that reason it can be predicted and cancelled by the ALP filter. $D=1$ causes that the voice in $d(n)$ and $d(n-D)$ is strongly correlated. A cascade of ALP filters of lower order independently adapted improves the modeling of the general ALP filter. In this case, the prediction is performed in successive refinements, the adaptation steps μ can be greater, and thus each stage is less affected by the disparity of *eigenvalues* which results in a faster convergence.

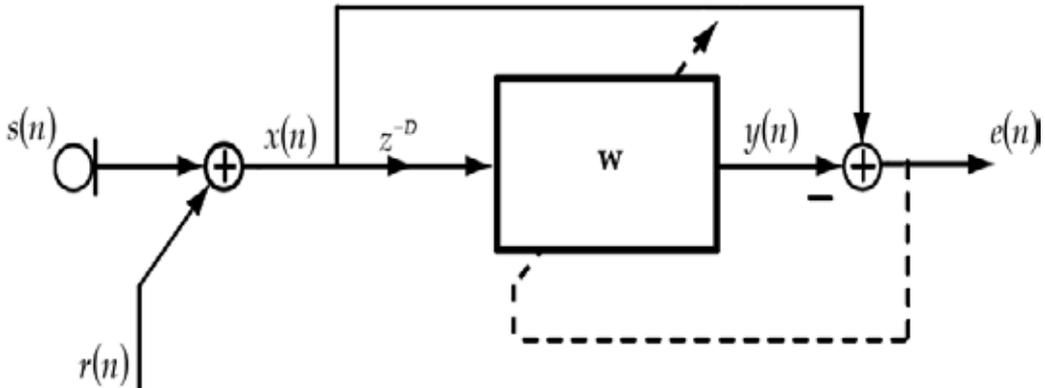


Fig. 9. Adaptive linear predictor.

4.3 Noise Cancellation

The *adaptive noise cancellation* (ANC) cancels the primary unwanted noise $r(n)$ by introducing a canceling antinoise of equal amplitude but opposite phase using a reference signal. This reference signal is derived from one or more sensors located at points near the noise and interference sources where the interest signal is weak or undetectable.

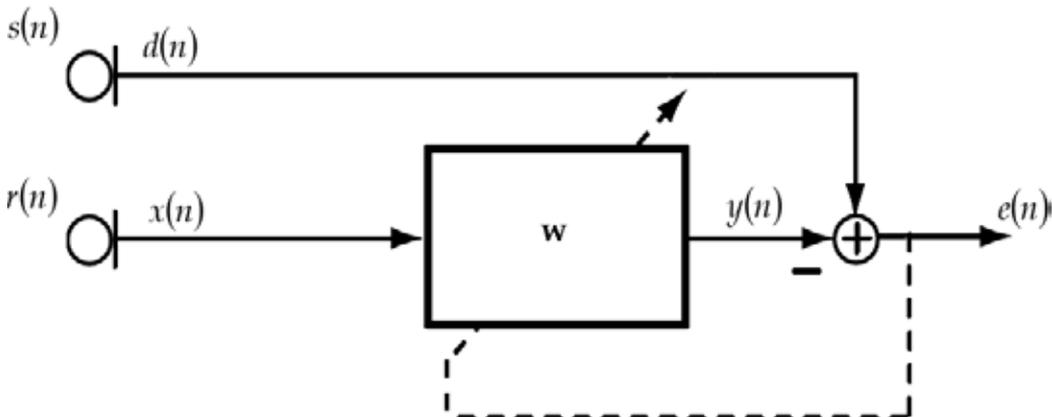


Fig. 10. Adaptive noise cancellation.

A typical ANC configuration is depicted in Fig. 10. Two microphones are used, $P = 2$. The

primary input $d(n) = s(n) + r(n)$ collects the sum of unwanted noise $r(n)$ and speech signal $s(n)$, and the auxiliary or reference input measures the noise signal $x(n) = r(n)$. ANC corresponds to multiple-input and single-output (MISO) *joint-process estimator* application (class c, Fig. 2) with at least two microphones, $P = 2$.

4.4 Beamforming

Beamforming is a multiple-input and single-output (MISO) application and consists of multichannel advanced multidimensional (space-time domain) filtering techniques that enhance the desired signal as well as suppress the noise signal.

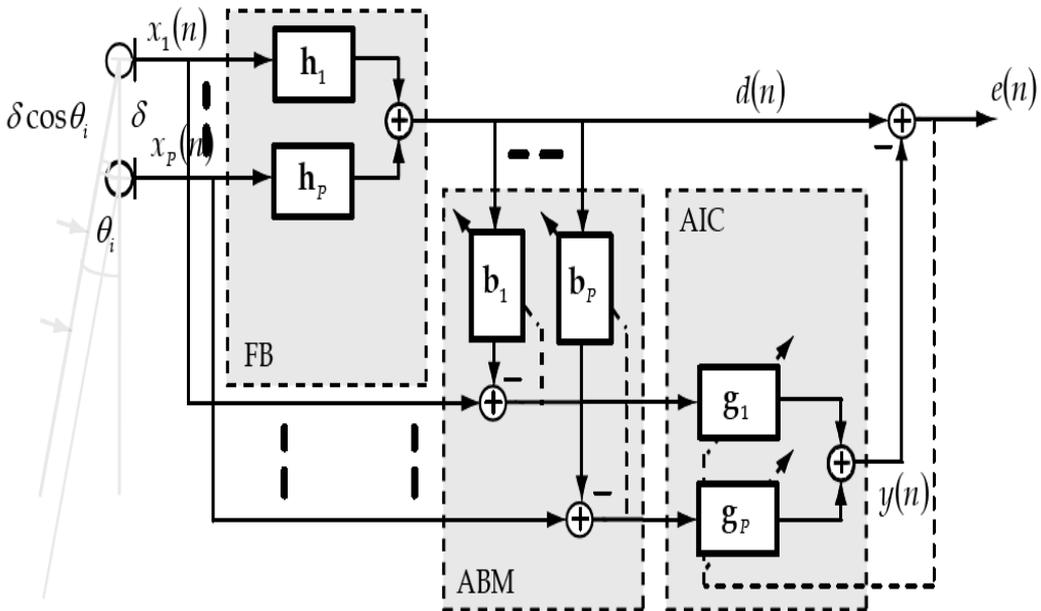


Fig. 11. Adaptive beamforming. Robust generalized sidelobe canceller (RGSC). Fixed beamforming (FB) allow conforming determined directivity pattern. The adaptive block matrix (ABM) or *blocking matrix*, with coefficient-constrained adaptive filters, prevents the target signal from leaking into the adaptive interference canceller (AIC). The AIC uses norm-constrained adaptive filters that can further improve the robustness against target signal cancellation.

In beamforming, two or more microphones are arranged in an array of some geometric shape. A *beamformer* is then used to filter the sensor outputs and amplifies or attenuates the signals depending on their *direction of arrival* (DOA), θ . The spatial response, or *beampattern*, of a beamformer generally features a combination of mainlobes that may be aimed at the target sources, and smaller sidelobes and null points aimed at the interference sources. Beampatterns are generally frequency-dependent, unless the beamformer is specifically designed to be frequency independent.

The sound sources are assumed to be in the *far-field* of the microphone array, i.e. the distance of the source from the array is much greater than the distance between the microphones (the spherical wavefronts emanating from the sources can be approximated as plane wavefronts). Each source $s_i(n)$ arrives to microphone 1 with delay $\kappa_i = \delta \cos \theta_i / v$ relative to its arrival to 2 because it has to travel an extra distance $\delta \cos \theta_i$; θ_i is the DOA of $s_i(n)$ and $v \approx 355 \text{ms}^{-1}$ is a velocity of sound. $\delta \leq v/F_s$ represents the spatial sampling interval of the wavefield; it has to fulfill this inequality to avoid spatial aliasing. The generalized sidelobe canceller (GSC) is an adaptive beamformer that keeps track of the characteristics of the interfering signal, leading to a high interference rejection performance. Initially, the P microphone inputs $x_p(n)$, $p = 1 \dots P$, go through the FB that directs the beam towards the expected DOA. The beamformer output $y(n) = \langle \mathbf{h}, \mathbf{x}(n) \rangle$ contains the enhanced signal originating from the pointed direction, which is used as a reference by the ABM. The coefficient vector \mathbf{h} has to fulfill both spatial and temporal constrains $\mathbf{C}\mathbf{h} = \mathbf{c}$, $\mathbf{h} = \mathbf{C}[\mathbf{C}^H\mathbf{C}]^{-1}\mathbf{c}$. The ABM adaptively subtracts the signal of interest, represented by the reference signal $y(n)$, from each channel input $x_p(n)$, and provides the interference signals. The columns of \mathbf{C} must be pairwise orthogonal to the columns of the blocking matrix \mathbf{B} , $\mathbf{C}^H\mathbf{B} = \mathbf{0}$. The quiescent vector \mathbf{h} is a component independent of data and $\mathbf{w} = \mathbf{h} - \mathbf{B}\mathbf{g}$ is a filter that satisfies the linear constrains $\mathbf{C}^H\mathbf{w} = \mathbf{C}^H(\mathbf{h} - \mathbf{B}\mathbf{g}) = \mathbf{C}^H\mathbf{h} = \mathbf{c}$. The upper signal path in Fig. 11 has to be orthogonal to the lower signal path. In order to suppress only those signals that originate from a specific tracking region, the adaptive filter coefficients are constrained within predefined boundaries (Benesty & Huang, 2003). These boundaries are specified based on the maximum allowed deviation between the expected DOA and the actual DOA. The interference signals, obtained from the ABM, are passed to the AIC, which adaptively removes the signal components that are correlated to the interference signals from the beamformer output $y(n)$. The norm of the filter coefficients in the AIC is constrained to prevent them from growing excessively large. This minimizes undesirable target signal cancellation, when the target signal leaks into the AIC, further improving the robustness of the system (Yoon et al., 2007).

In noise reduction systems, the beamformer can be used to either reject the noise (interference) by attenuating signals from certain DOAs, or focus on the desired signal (target) by amplifying signals from the target DOA and attenuating all signals that are not from the target DOAs. For non real-time speech enhancement applications it is possible to select a set of DOAs to be tested. Therefore adaptive algorithms with directional constrains, like a RGSC, can be exploited to achieve better noise reduction performance.

4.5 Deconvolution

Both *blind signal separation* (BSS), also known as *blind source separation*, and *multichannel blind deconvolution* (MBD) problems are a type of inverse problems with similarities and subtle differences between them: in the MBD only one source is considered, and thus the system is single-input single-output (SISO), while in BSS there are always multiple independent sources and the mixing system is MIMO; the interest of MBD is to deconvolve the source from the AIRs, while the task of BSS is double: on the one hand the sources must be separated, on the other hand the sources must be deconvolved from the multiple AIRs since

each sensor collects a combination of every original source convolved by different filters (AIRs) according to (2) (Gkalelis, 2004).

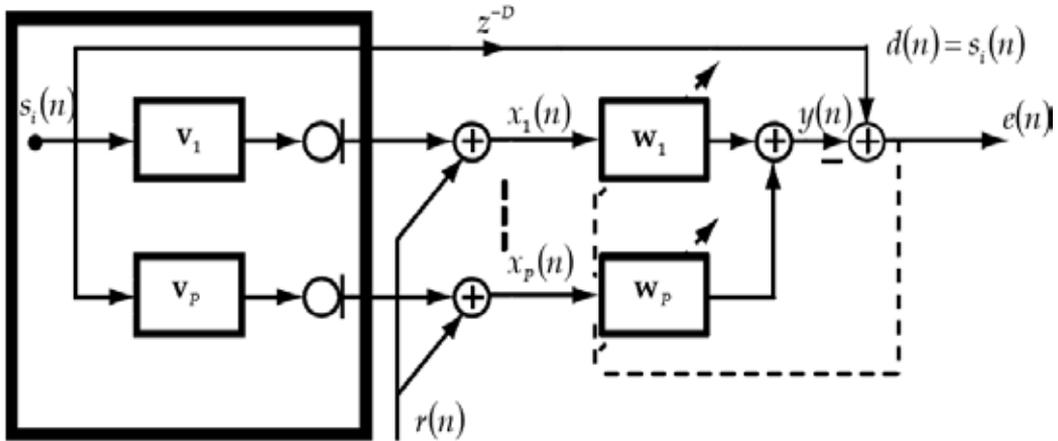


Fig. 12. Multichannel blind deconvolution.

In both cases, the *blind deconvolution* or *equalization* approach as well as the *blind separation* one, must estimate adaptively the inverse of the convolutive system that allows recovering the input signals and suppressing the noise. The goal is to adjust \mathbf{W} so that $\mathbf{WV} = \mathbf{PD}$, where \mathbf{P} is a permutation matrix and \mathbf{D} is a diagonal matrix whose (p,p) th is $\alpha_p z^{-\kappa_p}$; α_p is a nonzero scalar weighing, and κ_p is an integer delay.

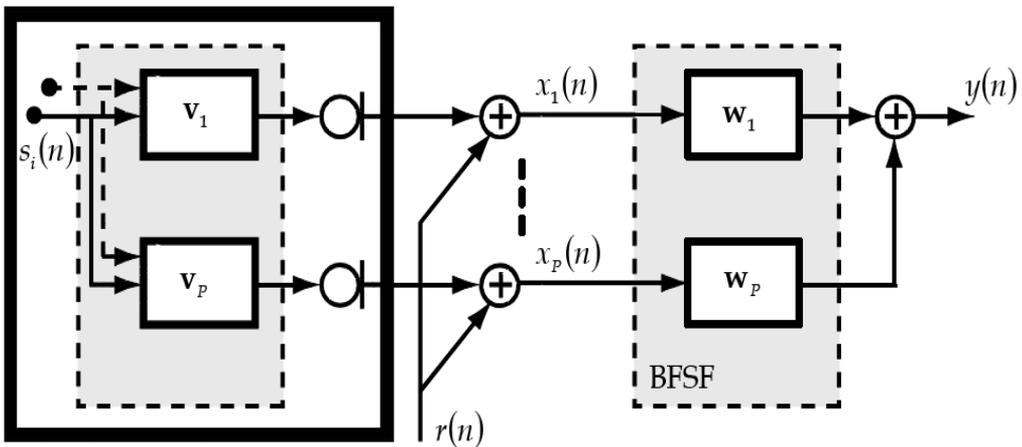


Fig. 13. Blind source factor separation.

BSS deals with the problem of separating I unknown sources by observing P microphone signals. In the *underdetermined* case ($P < I$) there are infinitely possible vectors $\mathbf{s}(n)$ that satisfy (3). There are mainly two ways to achieve the minimum norm solution. In the first,

the right generalized inverse of \mathbf{V} is estimated and then applied to the set of microphone signals $\mathbf{x}(n)$. Another class of algorithms employ the sparseness of speech signal to design better inversion strategies and identify the minimum norm solution. Many techniques of convolutive BSS have been developed by extending methods originally designed for blind deconvolution of just one channel. A usual practice is to use *blind source factor separation* (BSFS) technique, where one source (factor) is separated from the mixtures, and combine it with a *deflationary* approach, where the sources are extracted one by one after deflating, i.e. removing, them from the mixed signals. The MIMO FIR filter \mathbf{W} used for BSS becomes a multiple-input single-output (MISO) depicted in Fig. 13. The output $y(n)$ corresponds to (8) and the tap-stacked column vector containing all demixing filter weights defined in (7) is obtained as

$$\mathbf{u} = \mathbf{R}\mathbf{p} \quad (54)$$

$$\mathbf{w} = \frac{\mathbf{u}}{\sqrt{\mathbf{u}^H \mathbf{R} \mathbf{u}}}$$

where \mathbf{R} is a block matrix where its blocks are the correlation matrices \mathbf{R}_{pq} between the p -th channel and q -th channel defined in (22) and \mathbf{p} is a block vector where its blocks are the cross-cumulant vector $\mathbf{p} = \text{cum}\{\mathbf{x}(n), y(n) \dots y(n)\}$ (Gkalelis, 2004). The second step in (54) is just the normalization of the output signal $y(n)$. This is apparent left multiplying by $\mathbf{x}(n)$. The deflationary BSS algorithm for $i = 1 \dots I$ sources can be summarized as following: one source is extracted with the BSFS iterative scheme till convergence (54) and the filtering of the microphone signals with the estimated filters from the BSFS method (8) is performed; the contribution of the extracted source into the mixtures x_p , $p = 1 \dots P$, is estimated (with the LS criterion) and the contribution of the o -th source into i -th mixture is computed by using the estimated filter \mathbf{b} , $c(n) = \langle \mathbf{b}, \mathbf{y}(n) \rangle$ with $\mathbf{y}(n) = [y(n) \ y(n-1) \ \dots \ y(n-B+1)]$, $B \ll L$; deflate the contribution $c(n)$ from the p -th mixture, $x_p(n) = x_p(n) - c(n)$, $p = 1 \dots P$. This method is very suitable for speech enhancement application where only one source should be extracted, i.e. speech.

It is possible to consider the deflationary BSFS (DBSFS) structure as a GSC. ABM exactly corresponds to the deflating filters of the deflationary approach. By comparing the different parts, i.e. the BSFS block and the fixed beamformer, it is concluded that it may be possible to construct similar algorithms to those of GSC.

5. Conclusion

This chapter is an advanced tutorial about multichannel adaptive filtering for speech enhancement. Different techniques have been examined in a common foundation. Several approaches of filtering techniques were presented as the number of channels increases.

The spectral equalization (power subtraction), in general, can achieve more noise reduction than an ANC and a beamformer method. However, it is based on the noise spectrum

estimator instead of the unknown noise spectra at each time, producing a distortion known as “musical noise” (because of the way it sounds). The performance of ANC depends on the coherence between the input noisy signal and the reference noise signal. Only if the coherence is very high the results are spectacular, therefore, this fact limits its application to particular cases. The amount of noise that can be canceled by a beamformer relies on the number of microphones in the array and on the SNR of the input signal. More microphones can lead to more noise reduction. However, the effectiveness of a beamformer in suppressing directional noise depends on the angular separation between signal and the noise source (Benesty & Huang, 2003). The ALP method is very simple because only second order statistics are required, but the estimation is only optimal if the residue is i.i.d. Gaussian (Solé-Casals et al., 2000).

All these techniques are narrowly connected. The linear prediction of $x(n)$ is nothing but the deconvolution of $x(n)$ (Solé-Casals et al., 2000). In (Taleb et al., 1999), the problem of Wiener system blind inversion using source separation methods is addressed. This approach can also be used for blind linear deconvolution. In (Gkalelis, 2004) the link between the deflationary approach (the extension of the single channel blind deconvolution algorithm) and the traditional GSC structure is showed. Several strategies between different approaches are also possible, i.e. in (Yi & Philipos, 2007), a Wiener filter, that uses linear prediction to estimate the signal spectrum, is presented.

The best filter to enhance a particular recording will be chosen based on experience and experimentation (Koenig et al., 2007). Nevertheless, the algorithm developer would find it useful to have a quality measure that helps to compare, in general terms, the performance of different implementations of a certain algorithm (Yi & Philipos, 2007). One substantial ingredient of this performance is the intelligibility attained after processing the recording, or even better the increase of intelligibility compared to the unprocessed sample. Therefore, one possible way to measure the performance of an enhancement algorithm, and probably the best, would be to use a panel of listeners and subjective tests. To attain significant results, different speech recordings with different types and degrees of noise and distortion should be used as inputs to the algorithm, and therefore the task would probably become unapproachable in terms of time and effort, setting aside the fact that the experiment would hardly be repeatable.

In order to properly monitor the performance of the algorithms, different types and degrees of degradations should be imposed to the test signal. The model used to deal with degradations can be as simple as an additive noise, for a mono version of the test signal corrupted by random noise or a second talker speech, or as complex as a virtual room simulator for early reflexions and a stocastic reverberation generator, for a detailed acoustic model of the recording room, where several noise sources can be placed in different places. Measured impulse responses of a real chamber is another option to obtain very realistic mono or multi-channel virtual recordings.

6. References

- Armelloni, E.; Giottoli, C. & Farina, A. (2003) Implementation of Real-time Partitioned Convolution on a DSP Board, 2003 *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, pp. 71-74.

- Benesty, J. & Huang, Y. (2003) *Adaptive Signal Processing (Applications to Real-World Problems)*. Springer, Berlin Heidelberg New York Hong Kong London Milan Paris Tokyo.
- Boray, G.K. & Srinath, M.D. (1992) Conjugate Gradient Techniques for Adaptive Filtering. *IEEE Transactions on Circuits and Systems*, 39(1), 1-10.
- Chau, E.Y-H. (2001) *Adaptive Noise Reduction Using A Cascaded Hybrid Neural Network*. MS Thesis, University of Guelph, Ontario.
- Crochiere, R.E. & Rabiner, L.R. (1983) *Multirate Digital Signal Processing*. Prentice-Hall, London Sidney Toronto Mexico New Delhi Tokyo Singapore Rio de Janeiro.
- Friedlander, B. (1982) Lattice Filters for Adaptive Processing. *Proceedings of the IEEE*, 70(8), 829-867.
- García, L. (2006) *Cancelación de Ecos Multicanal*. PhD Thesis, Universidad Politécnica de Madrid, Spain.
- Gay, S.L. & Benesty, J. (2000) *Acoustic Signal Processing for Telecommunication*. Kluwer Academic Publishers, Boston Dordrecht London.
- Gkalelis, N. (2004) *Undetermined Blind Source Separation for Speech Signals*. MS Thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany.
- Glentis, G.-O.; Berberidis, K. & Theodoridis, S. (1999) Efficient least square adaptive algorithms for FIR transversal filtering: A unified view, *IEEE Signal Processing Magazine*, 16(4), 13-41.
- Haykin, S. (2002) *Adaptive Filter Theory*. Prentice-Hall, Inc., New Jersey.
- Honig, M.L. & Messerschmitt, D.G. (1984) *Adaptive Filters: Structures, Algorithms and Applications*. Kluwer Academic Publishers, Boston The Hague London Lancaster.
- Koenig, B.E.; Lacey, D.S. & Killion, S.A. (2007) Forensic enhancement of digital audio recordings, *Journal of the Audio Engineering Society*, 55(5), 352-371.
- Morgan, D.R. & Thi, J.C. (1995) A Delayless Subband Adaptive Filter Architecture, *IEEE Transactions on Signal Processing*, 43(8), 1819-1830.
- Páez Borrillo, J.M. & Otero, M.G. (1992) On The Implementation of a Partitioned Block Frequency Domain Adaptive Filter (PBFDAF) For Long Acoustic Echo Cancellation, *Signal Processing*, 27(3), 301-315.
- Reilly, J.P.; Wilbur, M.; Seibert, M. & Ahmadvand, N. (2002) The Complex Subband Decomposition and its Application to the Decimation of Large Adaptive Filtering Problems, *IEEE Transactions on Signal Processing*, 50(11), 2730-2743.
- Shynk, J.J. (1992) Frequency-domain and Multirate Adaptive Filtering, *IEEE Signal Processing Magazine*, 9(1), 14-37.
- Solé-Casals, J.; Jutten, C. & Taleb, A. (2000) Source Separation Techniques Applied to Linear Prediction, *2th International Workshop on Independent Component Analysis and Blind Source Separation Proceedings ICA2000*, pp. 193-198.
- Taleb, A.; Solé-Casals, J. & Jutten, C. (1999) Blind Inversion of Wiener Systems, *IWANN 99*, pp. 655-664.
- Vaseghi, S.V. (1996) *Advanced Signal Processing and Digital Noise Reduction*. John Willey & Sons Ltd. and B.G. Teubner, Chichester New York Brisbane Toronto Singapore Stuttgart Leipzig
- Yi, H. & Philipos, C.L. (2007) A comparative intelligibility study of single microphone noise reduction algorithms, *The Journal of the Acoustical Society of America*, 122(3), 1777-1786.

Yoon, B-Y.; Tashev, I. & Acero, A. (2007) Robust Adaptive Beamforming Algorithm Using Instantaneous Direction Of Arrival With Enhanced Noise Suppression Capability. *IEEE International Conference on Acoustics, Speech and Signal Processing* 1:I-133–I-136.

Multiple Regressive Model Adaptive Control

Emil Garipov*, Teodor Stoilkov* & Ivan Kalaykov**

**Technical University of Sofia, Bulgaria*

***Örebro University, Örebro, Sweden*

1. Introduction

It is common practice to use linear plant models and linear controllers in the control systems design. Such approach has simple explanation applying to plants with insignificant non-linearity or to those, functioning closely to a working point. But linear controllers, indeed with some modification, are used even for plants with significant non-linearities. Because of several reasons the non-linear controllers have not broad application. First, the linear control theory is well developed; while the non-linear control methods are clear for few engineers in practice. Second, there are some technological and economical difficulties to get high quality study of the process to be controlled in order to build detailed (more precise) non-linear plant model. Third, new ideas in the field of the control theory are continuously realized, which expand the span of the linear control systems applications as an alternative to utilizing complicated models at the expense of troubles of theoretical and practical nature.

During the last years a strategy “separate and rule over” is employed more and more by the researchers when trying to solve complex systems tasks using the principle: “Each complex task can be split into a limited number of simple subtasks in order to solve them independently, thereby formulate the solution of the initial complex task by their particular solutions”. Thus an old idea in the classical works [Wenk & Bar-Shalom, 1980; Maybeck & Hentz, 1987] was revived, so a complex (non-linear and/or time-variant) processes with high degree of uncertainty is represented by a family or a bank of (linear and/or time invariant) models with low degree of uncertainty [Li & Bar-Shalom, 1992; Morse, 1996; Narendra & Balakrishnan, 1997; Murray-Smith & Johansen, 1997].

In fact, the multiple-model adaptive control (MMAC) theory is based mainly on the state space representation via Kalman filters as a tool for static and dynamic estimation of the system model states [Blom & Bar-Shalom, 1988; Li & Bar-Shalom, 1996; Li & He, 1999]. The alternative of implementing multiple-model control using a set of input-output models was the next natural step, even to answer the question: “Why publications in the state space dominate and input-output models are not used for traditional linear control of complex plants, neglecting the fact that the standard system identification delivers basically such type of models”. The researchers in the field of switching control theory are among the

supporters of input-output models in the MMAC [Anderson et al., 2001; Hespanha & Morse, 2002; Hespanha et al., 2003].

A MMAC of time-variant plants using a bank of controllers designed on the base of linear sampled-time models is presented in the next sections. Our research on this topic is on dead-beat controllers (DBC), because on one side the design of DBC is relatively simple and on other side it is appropriate to demonstrate the theoretical development of the multiple-model control based on selecting the DBC order independently on the plant model order [Garipov & Kalaykov, 1991] and on selecting sampling period for the DBC independently on the sampling period of the entire control system. In both aspects the advantage of the DBC is the possibility to express and respectively determine the extreme magnitudes of the control signal through the DBC coefficients. Two approaches to implement the closed loop system are discussed, namely by switching and by weighting the control signal to the plant. A novel solution for MMAC is formulated, which guarantees the control signal magnitude to stay always within given constraints, introduced for example by the control valve, for all operating regimes of the system. Two types of multiple-model controllers are proposed: the first operates at fixed sampling period and contains a set of controllers of different orders, and the second contains a set of controllers of the same fixed order but computed for different sampling periods. Examples of the MMAC are demonstrated and results are compared with the behavior of some standard control schemes.

2. Main principles and concepts in the MMAC

2.1. Modeling the uncertainty in control systems

The most methods for controller design require a good knowledge of controlled plant dynamics or the exact plant model. If this information is incomplete the controller design is under the conditions of *a priori uncertainty* regarding the structure and parameters of the plant model and/or disturbances on the plant. On the other hand, the study of the most industrial processes during their operation is impeded due to equipment aging or failures, operating regimes variations or/and noisy factors changes. And if the a priori uncertainty could be justified before the control design, the *a posteriori uncertainty* accompanies the entire control system work. It is obvious that the continuously variation in operating conditions make the controllers function incorrect during the time even in case of exactly known process models.

The historical overview shows various ways of representing the uncertainty in control systems. Limiting the framework to the difference equation as a typical input-output plant description, one can find out that the deterministic time invariant model is substituted in the seventies of 20 century with the stochastic one and the plant dynamics uncertainty is presented by an unmeasured random process on its output, i.e. *the uncertainty is presented as a noise in the output measurements.* When the theory moved the emphasis to time-variant systems in the eighties, this was a sign of recognition that if plant dynamics is changing in time, it can be tracked by estimating the changing model parameters, i.e. *the uncertainty is presented as a noise over the physical model parameters.* Meanwhile there were attempts deterministic interval models to be applied, so the uncertain plant dynamics is presented by a multi-variant model, i.e. *the uncertainty is described as a combination of disturbances to the physical model parameters.* Time-variant and interval models describe with various degrees of

complexity changing plant dynamics. The first type of models can be substituted with the bank of elementary time-invariant models called local models. The second type of models includes a set of time-invariant models for the plant dynamics, every one of which defined within a given range of plant parameters variations. In this case the local models correspond to particular operating regimes or plant states. Nevertheless, for both types of models the following idea is used: a bank of more simple models is used instead of its complicated presentation by a global model. It means that the plant control design of a complex controller can be replaced by a bank of local controllers tuned for every elementary model.

2.2. Multiple-model adaptive control (MMAC)

The core idea to get over the control system uncertainty is to realize a strategy for control of arbitrary in complexity plant by a bank of linear discrete controllers, which parameters depend on the corresponding linear discrete models, presented the plant dynamics at various operating regimes. This strategy is known as *multiple model adaptive control (MMAC)*. The following characteristics are typical for this type of control:

- First, the continuous-time space of the plant dynamics is approximated at limited number of operating regimes. This approach is something other than the indirect adaptive control (well known as self tuning control (STC), where the estimation procedure takes place at each sampling instant, which means that the plant dynamics is examined at practically infinite number of operating points. Hence the MMAC is defined as a new control methodology, which provides new features of the control system by simply using the elements and techniques from the classical control theory and practice.
- Second, MMAC escapes the necessity of on-line plant model estimation. It is true that the bank of local models corresponds to the current plant dynamics at each operating point but these structures are evaluated before the control system starts operating. Hence, MMAC can avoid also all problems of the closed-loop identification compared to the standard indirect adaptive control.
- Third, in case when one exact plant model is not suitable for all operating regimes, the following MMAC approaches can be applied:
 - (a). Multiple-Model Switching Control (MMSC) - Used if the operating regimes are predefined or are quite different. The principle of relay-race control can be observed – each controller of the bank takes independent action in the control system tuned according the best corresponding plant model at the corresponding regime.
 - (b). Multiple-Model Weighting Control (MMWC) - Used if the operating regimes are not known in advance. The plant description is made as combination of the models for other operation regimes or as mixture of limited number of hypothetical models taken from the model bank. The global control is formed by contributions of all local controllers of the bank depending on various weights.

Hence, MMAC is defined as adaptive control, because it uses different combinations of models to describe complex system behavior, thus, even when the plant and controller are time-variant, the controller is designed as being for a time-invariant system.

- Forth, to identify the current operating regime is a specific task to recognize single or a set of performance indices of the control system. A test or number of tests is applied in order to determine some desired conditions (model and plant fit, control system errors,

system performance with respect to a reference model, constraints on signals in the system, etc.), then predefined or prepared in advance solutions for the multiple-model controller behavior is selected. From that viewpoint, the MMAC can be seen as supervisory control as well.

3. Design of MMAC based on a bank of input-output models

3.1. Stages of the design

The multiple model control using bank of controllers tuned under corresponding bank of plant models is a classical control scheme. Usually the model's and controller's sampling periods are the same as the control system sampling period. A block-diagram of the MMACS is given in Fig. 1 for time-variant plant control.

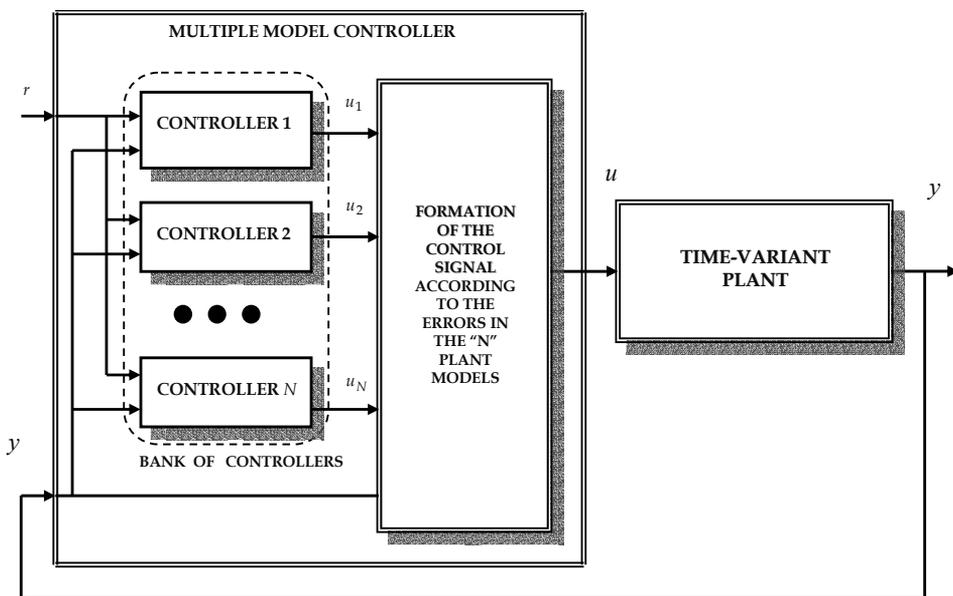


Fig. 1. Structure scheme of the MMACS

The design of MMACS is performed in several stages:

Stage 1. Preliminary choice of a limited set of models, including amount and type of models, estimation of model parameters. Naturally, the MMAC designer aims at a good model, and therefore at a good controller covering a wide range of the system operating conditions. MMACS will act optimally if the model adequately presents the identified plant. When the system is not well studied and it is difficult to obtain a non-linear plant model, MMAC offers the use of a combination of linear models or the choice of the best one among the model set. Such solution is sub-optimal, but acceptable for the prescribed performance criteria. Continuous-time or sampled-time models may be used but the last one is common. The amount of selected models is usually related to the operating condition at which the control system is expected to work.

Stage 2. Preliminary design of sampled multiple-model controller including amount and type of the local controllers and tuning of the controllers' coefficients. The system functional behavior depends mainly on the designed controllers according to the predefined system performance criterion, which is related to the controlled variable $y(\cdot)$. It is accepted that each local controller is tuned according to the corresponding model in the set from Stage 1.

Stage 3. Implementation of the control system, including selection and implementation of the techniques for calculation of the weighting coefficients, and choice of system initial conditions. The weighting technique determines the weights μ_i for the output of each local controller. Usually they depend on values of preferred error signals, for example identification errors, control errors, deviations from a trajectory, etc., which are included into a corresponding criterion such as integral square error ISE, integral absolute error IAE, etc., under given observed time interval. One possible universal type of supervisor to form the control u in a control system is shown in the block diagram on Fig. 1 and is disclosed in details on Fig. 2.

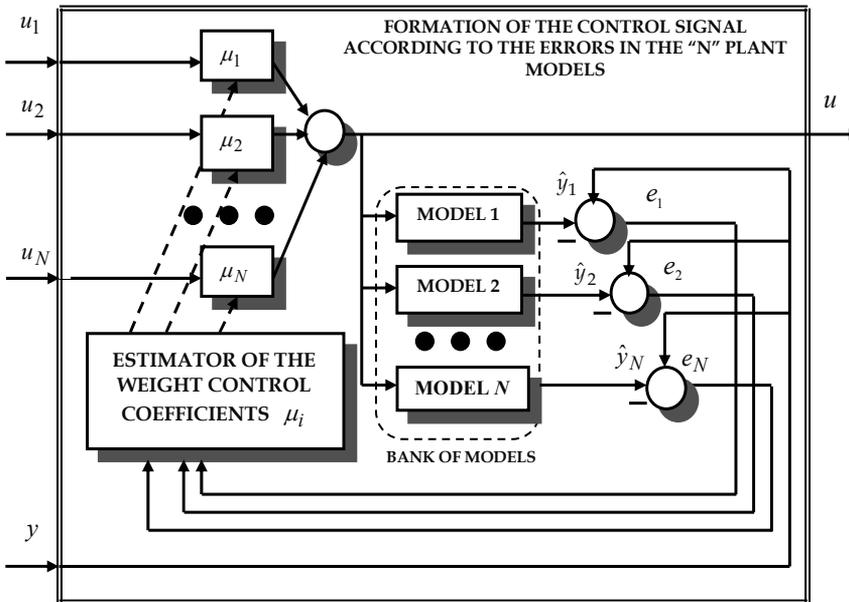


Fig. 2. Detailed block diagram of the MMACS

3.2. Algorithm for output feedback system controller

The general tasks described above can be ordered in the following basic algorithm for discrete MMAC of a continuous-time plant.

Step 1. The number N of the operating plant regimes is specified. The sampling period T_0 for the system is selected, meaning that the signals are measured at time instances kT_0 , $k=0,1,\dots,M$, during the time interval $T = MT_0$. The sampling period is further excluded

from the values index to get shorter notation. The reference signal is defined $r(kT_0) \equiv r(k)$, $k=0,1,\dots,M$, $r(0)=0$. Initial values of the weighting coefficients $\mu_j(0)=1/N$ are determined. The control system is examined taking the zero initial conditions for the plant and the controller, i.e. $y(0)=0$ and $u(0)=0$. Then in the classical system the output signal will be formed answering the following rules:

- ◆ Rule 1. $y(1)=\dots=y(d)=0$ in case of d sampling periods time-delay in the system ($d \geq 1$ for sampled continuous-time plant using zero-order hold),
- ◆ Rule 2. $y(d+1) \neq 0$, when $r(1) \neq 0$ and $u(1) \neq 0$, if the controller doesn't put its own time-delay into the control system.
- ◆ Rule 3. During the first d sampling intervals the system operates practically without a feedback, therefore certain well known problems in the control might appear.

Step 2. A bank of N discrete-time models is identified using the input-output plant measurements collected at the specified N operating regimes. The suggested models are

$$A_j(q^{-1})y(k) = B_j(q^{-1})u(k-d_j) + e_j(k), \quad (j=1,2,\dots,N), \quad (1)$$

where $A_j(q^{-1}) = 1 + a_{j1}q^{-1} + \dots + a_{jn_a}q^{-jn_a}$ and $B_j(q^{-1}) = b_{j1}q^{-1} + \dots + b_{jn_b}q^{-jn_b}$ are polynomials of the unit delay q^{-1} with order, respectively, jn_a and jn_b (more often $jn_a = jn_b = jn$), and d_j presents the delay of the j -th model expresses as integer number of sampling periods. The random signal e_j represents the model estimation error (the mismatch between the physical plant dynamics and its model, the measurement noise or any other disturbances to the plant). When the error is a white Gaussian process, the model parameter estimates are unbiased according the standard least squares method. In case of colored noise the modified least squares methods have to be used in order to get unbiased estimates.

Step 3. A multiple-model controller is designed containing a set of N sampled-time controllers with 2 degrees of freedom and description

$$P_j(q^{-1})u_j(k) = -S_j(q^{-1})y(k) + T_j(q^{-1})r(k), \quad (j=1,2,\dots,N), \quad (2)$$

where polynomials $P_j(q^{-1}) = 1 + p_{j1}q^{-1} + \dots + p_{jn_p}q^{-jn_p}$, $S_j(q^{-1}) = s_{j0} + s_{j1}q^{-1} + \dots + s_{jn_s}q^{-jn_s}$ and $T_j(q^{-1}) = 1 + t_{j1}q^{-1} + \dots + t_{jn_t}q^{-jn_t}$ have sizes and parameters according to the selected design method, as well as the corresponding plant model from the bank of the N -th sampled-time models (2). A special case of (3) presents sampled-time controllers with one degree of freedom $S_j(q^{-1}) = T_j(q^{-1}) = Q_j(q^{-1}) = q_{j0} + q_{j1}q^{-1} + \dots + q_{jn_q}q^{-jn_q}$, $r(k) - y(k) = e(k)$, so that

$$P_j(q^{-1})u_j(k) = Q_j(q^{-1})e(k), \quad (j = 1, 2, \dots, N). \quad (3)$$

A cycle for MMACS elements operating at $k = 1, 2, \dots, M$

Start

Step 1. *Functioning of the set of controllers.*

A current control signal is formed at the output of the j -th local controller described by equation (3)

$$\begin{aligned} u_j(k) = & -p_{j1}u_j(k-1) - \dots - p_{jn_p}u_j(k-jn_p) - s_{j0}y(k) - s_{j1}y(k-1) - \dots - s_{jn_s}y(k-jn_s) \\ & + r(k) + t_{j1}r(k-1) + \dots + t_{jn_r}r(k-jn_r) \end{aligned} \quad (4)$$

or by equation (4)

$$u_j(k) = -p_{j1}u_j(k-1) - \dots - r_{jn_p}u_j(k-jn_p) + q_{j0}e(k) + q_{j1}e(k-1) + \dots + q_{jn_q}e(k-jn_q) \quad (5)$$

Step 2. *Weighting control signals of all local controllers.*

A global control signal is calculated by

$$u(k) = \sum_{j=1}^N \mu_j(k-1) u_j(k), \quad (j = 1, 2, \dots, N) \quad (6)$$

The initial values of the weighting coefficients in MMACS can be selected as $\mu(0) = 1/N$, i.e. the weighting mechanism starts with an equal weight of each controller.

Step 3. *Plant response measured.*

The plant output $y(k) = f\{y(k-1), \dots, u(k-1), \dots\}$ is measured.

Step 4. *Supervisory function to form the weighting coefficients at the next cycle.*

Step 4.1. The output $\hat{y}_j(k)$ of each local model is calculated

$$\hat{y}_j(k) = -a_{j1}\hat{y}_j(k-1) - \dots - a_{jn_a}\hat{y}_j(k-jn_a) + b_{j1}u_j(k-d_j) + \dots + b_{jn_b}u_j(k-d_j-jn_b), \quad (7)$$

Step 4.2. The a posteriori residual error $\hat{e}_j(k)$ at each local model output is estimated

$$\hat{e}_j(k) = \{y(k) - \hat{y}_j(k)\} / r(k) \quad (8)$$

Step 4.3. A performance index of each local model is fixed

$$J_j(k) = \hat{e}_j^2(k) \quad (9)$$

Step 4.4. An exponential smoothing is applied to decrease the influence of random factors to the MMACS

$$\bar{J}_j(k) = \lambda \bar{J}_j(k-1) + (1-\lambda)J_j(k), \quad \bar{J}_j(0) = J_j(0), \quad (10)$$

where $\lambda = e^{-4/L}$, L is the number of old values between which the smoothing is done.

Step 4.5. The weighting coefficients for the next cycle of the procedure are calculated

$$\mu_j(k) = \bar{J}_j^{-1}(k) \left[\sum_{j=1}^N \bar{J}_j^{-1}(k) \right]^{-1}, \quad \sum_{j=1}^N \mu_j(k) = 1. \quad (11)$$

End

Alternative step 4.5. If the weighting control expression (12) is exchanged by switching one, then at each sampled-time instant a local controller with index c operates only, which is equivalent to setting the weight of the c -th local controller to be 1, i.e. $\mu_c(k) = 1$. The corresponding plant model is chosen among the set of models according to the threshold value $\bar{J}_c(k)$ in the inequality [Boling et al, 2003]:

$$\bar{J}_c(k) > (1+h) \min_j \{ \bar{J}_j(k) \}. \quad (12)$$

3.3. Test design example of MMACS

Let the continuous time-variant plant be defined as:

$$W_o(p) = \frac{K^{(t)}}{(T_1^{(t)}p+1)(T_2p+1)(T_3p+1)}$$

with time-invariant constants $T_2 = 7.5 \text{ s}$ and $T_3 = 5 \text{ s}$. It is proposed that the observation interval is $T = kT_0$, $k = 0, 1, \dots, M = 300$, $T_0 = 1 \text{ s}$, and the gain $K^{(t)}$ and the time constant $T_1^{(t)}$ evolve as shown on Fig. 3a and Fig. 3b.

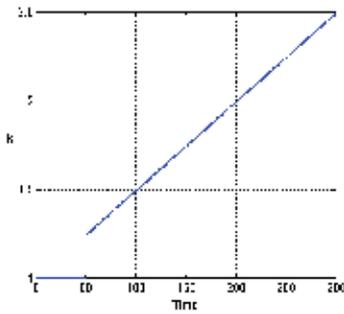


Fig. 3a. Evolution of $K(t)$

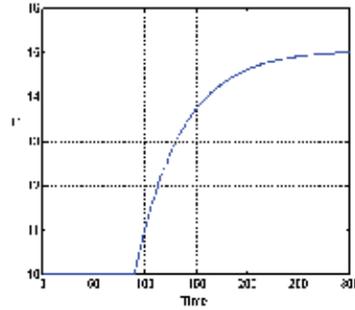
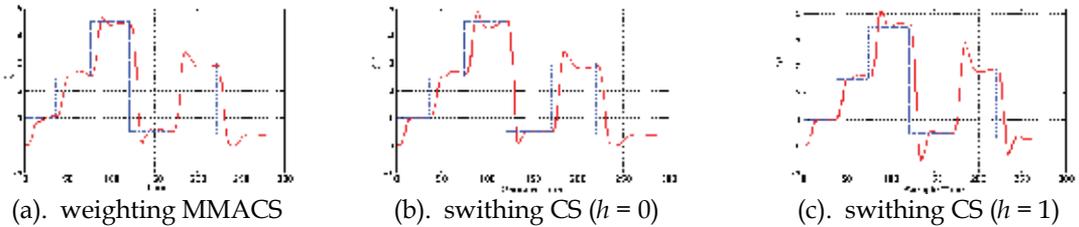


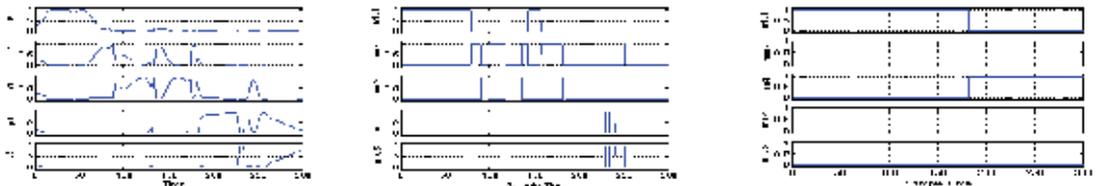
Fig. 3b. Evolution of $T_1(t)$

The MMACS for this plant is tested at 5 operating regimes ($M = 5$). The corresponding primary continuous time-invariant plant models are defined to cover the areas of the parameters' evolution. Five local sampled-time models are calculated to form the bank of models in order to design the local controllers. Then a multiple-model controller is constructed, consisting of five dead-beat controllers each of them tuned for the corresponding model according to the technique described in the next sections. MMACS starts with equal weights $\mu_i(0) \equiv \mu(0) = 1/5$.

Some tests of the designed MMACS are on Fig. 4a (system output and reference), and on Fig. 5a (the behavior of the weighting coefficients for each local controller output). Table 1 demonstrates that MMACS outperforms the other systems. The importance of this is additionally highlighted by the fact MMACS does not use the time-consuming plant identification procedure in real time as a part of self tuning controller.



(a). weighting MMACS (b). switching CS ($h = 0$) (c). switching CS ($h = 1$)
Fig. 4. Output y and reference r



(a). weighting MMACS (b). switching CS ($h = 0$) (c). switching CS ($h = 1$)
Fig. 5. Weighting coefficients

Type of the control system	A quality measure
MMACS with weighted control u	0.8122
MMACS with switching control u ($h = 0$)	0.8137
MMACS with switching control u ($h = 1$)	0.8161
Classical CS	0.9236
Adaptive CS (STC)	0.8892

Table 1. Mean-square error for comparison

4. Multiple-model adaptive control with control signal constraints

4.1. Introduction

Control systems in practice operate under constraints on the control signal, normally introduced by the control valve. When such constraints are not included in the design of the controller, the system performance differs significantly to the theoretically expected behavior. In case control signal formed by the standard controller is beyond these constraints, it will not be propagated with the required (expected) magnitude according to the unconstrained case. The solution of this problem is offered by the DB controller with increased order, as it adheres to the important principle "*an increased order controller provides decreased magnitude of the control signal*" [Isermann, 1981]. Accordingly, the choice of the order increment that can reduce the influence of the constraints on the control signal is a recommended requirement for the control system designer [Garipov & Kalaykov, 1991].

In control systems with existing constraints on the magnitude of the control signal, linear control at any operating point is feasible in the following two cases:

- (a) When the controller is of sufficiently high order, such that it provides control magnitude not beyond the constraints. Such over dimensioned controller, however, is normally inert and sluggish.
- (b) When the controller is of varying order, which adjusts its coefficients according the necessity to keep the control signal magnitude limited [Garipov & Kalaykov, 1991], but preserving the linear nature of the controller at any operating point of the system.

Equivalent of the system of case (b) is the multiple-model control system with multiplexing DB controllers of various orders is shown on Fig. 6.

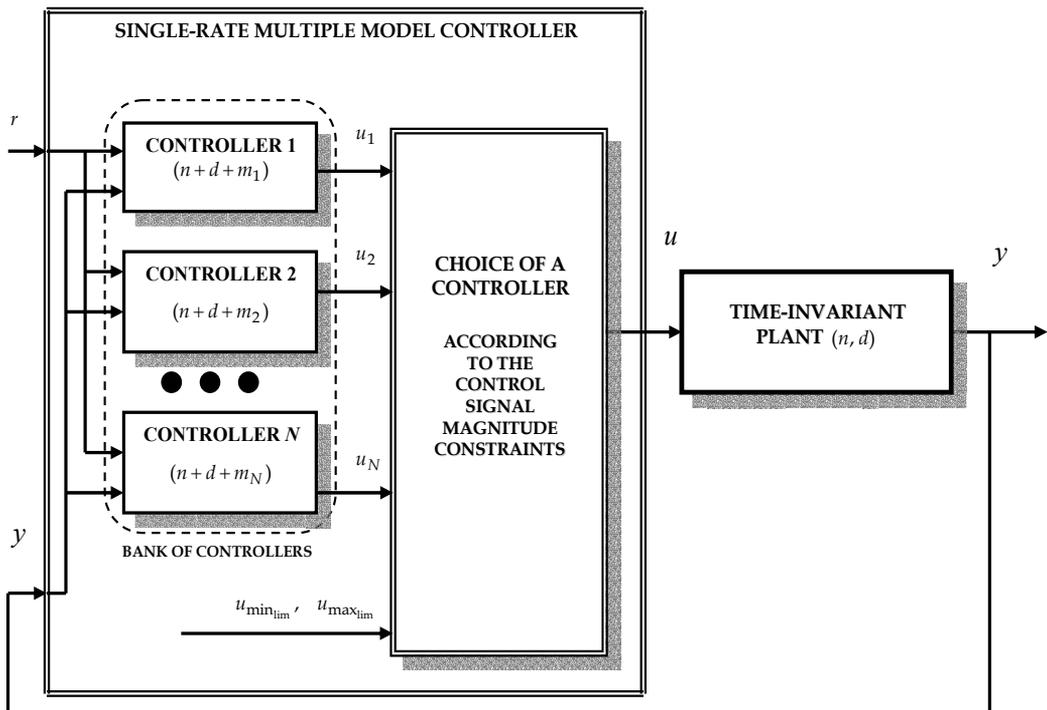


Fig. 6. Single-rate multiple-model system with control signal constraints

The difference to the system described in the previous Section 3, given on Fig. 1, is the selection mechanism, which is based on the requirement the DBC to guarantee a linear control signal within the predefined constraints at any operating point, depending on the desired (possibly stepwise changing only) reference signal.

4.2. Design of DBC of increased order [Garipov & Kalaykov, 1990]

Fundamental property of the DBC is the finite step response time $(n+d)T_0$ of the closed-loop system, where n is the order and d is the time delay of the sampled-data model of the controlled plant. Keeping the sampling period T_0 we can only obtain longer step response by increasing the DBC order. This, however, has the positive effect of decreasing the extreme magnitudes of the control signal, because the energy of the control signal spreads over larger number of sampling intervals (longer time). Thus, the application of DB control can be revived, as presented in the text below.

The design of DBC of increased order is based on the following assumptions:

Assumption 1: The DBC of increased order denoted by DBC $(n+m, d)$ is described by fraction of two polynomials of n_p -th order, where $n_p = n + d + m$ and m means the order increment. When $m = 0$, the DBC is of normal order, when $m = 1$ DBC is of increased by one order [Isermann, 1981], etc.

Assumption 2: At initial conditions the controlled variable $y(0) = 0$, a step change of the reference signal r is applied ($r(0) = r(1) = \dots = 1$) and the step response settles in finite time, i.e.

$$\begin{aligned} y(0) = y(1) = \dots = y(d) = 0, \quad y(d+1) \neq 0, \quad y(k) = r(k) = 1 \quad \text{when } k \geq n+d+m, \\ u(0) \neq 0, \quad u(k) = u(n+m) \quad \text{when } k > n+m. \end{aligned} \quad (13)$$

Following the signal behavior after (14), the z-domain images of the respective quantities are

$$Y(z) = \sum_{i=1+d}^{n+d+m-1} y(i)z^{-i} + 1 * \left[\sum_{s=n+d+m}^{\infty} z^{-s} \right], \quad U(z) = \sum_{i=0}^{n+m-1} u(i)z^{-i} + \text{const} * \left[\sum_{s=n+m}^{\infty} z^{-s} \right], \quad R(z) = \frac{1}{1-z^{-1}}.$$

By constructing the following two fractions

$$\frac{Y(z)}{R(z)} = z^{-d}P^{(m)}(z) = \sum_{i=1}^{n+m} p_i z^{-i} = (1-z^{-1})Y(z), \quad \frac{U(z)}{R(z)} = z^{-d}Q^{(m)}(z) = \sum_{i=0}^{n+m} q_i z^{-i} = (1-z^{-1})U(z),$$

the DBC polynomials $P^{(m)}$ and $Q^{(m)}$ are formed. It is not difficult to establish the following **properties of the coefficients of the DBC($n+m,d$)**:

$$\begin{aligned} \sum_{i=1}^k p_i^{(m)} = y(k), \quad k = 1+d, 2+d, \dots, n+d+m \quad \text{and} \quad \sum_{i=1}^{n+m} p_i^{(m)} = y(n+d+m) = 1 \\ \sum_{i=0}^k q_i^{(m)} = u(k), \quad k = 0, 1, 2, \dots, n+m \quad \text{and} \quad \sum_{i=0}^{n+m} q_i^{(m)} = u(n+m) = \text{const}. \end{aligned} \quad (14)$$

The closed-loop control system has a transfer function

$$W_{CL}(z) = \frac{Y(z)}{U(z)} = z^{-d}P^{(m)}(z) = \frac{\sum_{i=1+d}^{n+d+m} p_i z^{n+d+m-i}}{z^{n+d+m}} \quad (15)$$

with a characteristic equation $z^{n+d+m} = 0$, which means that the system has an infinite degree of stability. The properties of $p_i^{(m)}$ coefficients in (15) means the DBC($n+m,d$) guarantees the system steady error to be asymptotically closed to zero (17), because

$$W_{CL}(z)|_{z=1} \equiv z^{-d}P^{(m)}(z)|_{z=1} = \sum_{i=1}^{n+m} p_i = 1. \quad (16)$$

It is not difficult to prove the inherent existence of the same property of the DBC($n+m,d$). Rewriting the DBC transfer function

$$W_C(z) = \frac{1}{W(z)} \frac{W_{CL}(z)}{1 - W_{CL}(z)} = \frac{1}{W(z)} \frac{z^{-d}P^{(m)}(z)}{1 - z^{-d}P^{(m)}(z)} \quad (17)$$

and including the controlled plant transfer function

$$\frac{z^{-d}B(z)}{A(z)} = W(z) = \frac{Y(z)}{U(z)} = \frac{\frac{Y(z)}{R(z)}}{\frac{U(z)}{R(z)}} = \frac{z^{-d}P^{(m)}(z)}{Q^{(m)}(z)}, \quad (18)$$

yield

$$W_C(z) = \frac{Q^{(m)}(z)}{1 - z^{-d}P^{(m)}(z)} = \frac{Q_C(z)}{P_C(z)}. \quad (19)$$

The denominator polynomial in (20) is

$$P_C(z) = 1 - z^{-d}(p_1^{(m)}z^{-1} + p_2^{(m)}z^{-2} + \dots + p_n^{(m)}z^{-n} + \dots + p_{n+m}^{(m)}z^{-n-m}),$$

which can be modified such, that the inherent integral part of DBC($n+m,d$) can be demonstrated

$$P_C(z) = (1 - z^{-1})\bar{P}(z),$$

where

$$\begin{aligned} \bar{P}(z) = & [(1 + z^{-1} + \dots + z^{-d})p_1^{(m)} + (1 + z^{-1} + \dots + z^{-d} + z^{-(d+1)})p_2^{(m)} + \dots + \\ & (1 + z^{-1} + \dots + z^{-(d+n+m-1)})p_{n+m}^{(m)}] \end{aligned}$$

Hence, the final description of DBC($n+m,d$) becomes

$$W_C(z) = \frac{Q^{(m)}(z)}{(1 - z^{-1})\bar{P}(z)} = \frac{Q_C(z)}{P_C(z)}, \quad \deg \bar{P}(z) = n + m + d - 1. \quad (20)$$

$$A_2 = \begin{bmatrix} a_n & a_{n-1} & \cdot & \cdot & a_1 \\ 0 & a_n & a_{n-1} & \cdot & a_2 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & a_n \end{bmatrix}, \quad B_2 = \begin{bmatrix} b_n & b_{n-1} & \cdot & \cdot & b_1 \\ 0 & b_n & \cdot & \cdot & b_2 \\ 0 & 0 & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & b_n \end{bmatrix}, \quad E = [111\dots 1],$$

$\dim A_1 = (n+m) \times (n+m)$, $\dim B_1 = (n+m) \times (n+m+1)$, $\dim A_2 = n \times n$, $\dim B_2 = n \times n$, $\dim E = 1 \times (n+m)$, $\dim p = (n+m) \times 1$, $\dim q = (n+m+1) \times 1$. The dimensions of the null-matrices are: $\dim D_e = 1 \times (n+m+1)$, $\dim D_a = n \times m$, $\dim D_b = n \times (m+1)$.

Obviously, the equation (23) represents incomplete system of $(2n+m+1)$ linear equations with $(2n+2m+1)$ unknown parameters. Therefore, to enable the solution m additional equations have to be added, for example m conditions for the elements in the θ vector.

From (15) it is already known that the control signal magnitude at separate time instants depends on the coefficients in the $Q^{(m)}$ polynomial. This can be used to formulate a new concept for design of $DB(n+m,d)$ controller: the unique solution of (23) to be persuaded by appropriate fit of the coefficients q_i with the constraints on the control signal. We complement the already demonstrated by Isermann [Isermann, 1981] principle "*an increased order controller provides decreased magnitude of the control signal*" by a new concept, namely "*flexible tuning of the coefficients q_0, q_1, \dots, q_m can provide linear control at any operating point of the control system*".

Accordingly, in the proposed method for $DB(n+m,d)$ controller design we introduce a matrix Z , $\dim Z = m \times (n+m+1)$, which augments the incomplete rank equation (23) to the full rank equation

$$X\theta = Y, \quad (23)$$

where the matrices

$$X = \begin{bmatrix} & X^* & \\ \dots & \dots & \dots \\ D_z & \vdots & Z \end{bmatrix}, \quad Y = \begin{bmatrix} Y^* \\ \dots \\ D_y \end{bmatrix}$$

have dimensions $\dim X = (2n+2m+1) \times (2n+2m+1)$, $\dim Y = 2n+2m+1$ and D_z, D_y are zero-blocks with dimensions $\dim D_z = m \times (n+m)$, $\dim D_y = m \times 1$.

The following rules for putting together the elements of the Z matrix are established:

Rule 1. The number of elements in a row corresponds to the number of coefficients in the $Q^{(m)}$ polynomial, as $\deg Q^{(m)} = n+m+1$.

- Rule 2.** The elements can take only a binary value: "0" means that the corresponding coefficient of the $Q^{(m)}$ polynomial exists (i.e. it is nonzero), while "1" means the corresponding coefficient of $Q^{(m)}$ does not exist (i.e. we consider this coefficient as set to zero).
- Rule 3.** Only one value "1" is permitted in a row and it cannot be at the first or last position, because this means change of the degree of $Q^{(m)}$.
- Rule 4.** The nonexistence of the j -th coefficient ($j = 2, 3, \dots, n+m$) in $Q^{(m)}$ (i.e. it is set to zero) means holding of the $(j-1)$ -th value of the control signal, which enables the designer to shape the behavior of the control system.

The proposed methodology is demonstrated below for a $DB(n+m,d)$ controller with $m = 2$ for a third order plant with a delay ($n_a = n_b = n = 3$ and $d = 1$):

$$W(z) = \frac{0.06525z^{-1} + 0.04793z^{-2} - 0.00750z^{-3}}{1 - 1.49863z^{-1} + 0.70409z^{-2} - 0.09978z^{-3}} z^{-1}$$

The controller parameters estimates, allocated in the unknown parameters vector,

$$\theta = \left[p_1^{(2)} p_2^{(2)} p_3^{(2)} p_4^{(2)} p_5^{(2)} q_0^{(2)} q_1^{(2)} q_2^{(2)} q_3^{(2)} q_4^{(2)} q_5^{(2)} \right]^T, \\ \dim \theta = (2 * 3 + 2 * 2 + 1) \times 1 = 11 \times 1,$$

can be obtained by solving equation (24), in which

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_0 & 0 & 0 & 0 & 0 & -b_1 & 0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 & -b_2 & -b_1 & 0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & 0 & 0 & -b_3 & -b_2 & -b_1 & 0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 & 0 & 0 & -b_3 & -b_2 & -b_1 & 0 & 0 \\ 0 & a_3 & a_2 & a_1 & a_0 & 0 & 0 & -b_3 & -b_2 & -b_1 & 0 \\ 0 & 0 & a_3 & a_2 & a_1 & 0 & 0 & 0 & -b_3 & -b_2 & -b_1 \\ 0 & 0 & 0 & a_3 & a_2 & 0 & 0 & 0 & 0 & -b_3 & -b_2 \\ 0 & 0 & 0 & 0 & a_3 & 0 & 0 & 0 & 0 & 0 & -b_3 \\ 0 & 0 & 0 & 0 & 0 & z_{11} & z_{12} & z_{13} & z_{14} & z_{15} & z_{16} \\ 0 & 0 & 0 & 0 & 0 & z_{21} & z_{22} & z_{23} & z_{24} & z_{25} & z_{26} \end{bmatrix}, \quad Y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\dim X = 11 \times 11, \quad \dim Y = (2 * 3 + 2 * 2 + 1) \times 1 = 11 \times 1.$$

Six alternatives for selecting the Z matrix elements, ($\dim Z = 2 \times 6$), are possible as shown below. For some of them the expected behavior of the control signal u is illustrated. The $Q^{(m)}$ coefficients are given in Table 2 and $P^{(m)}$ coefficients are in Table 3.

Alternative 1. $Z = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$, therefore the DB control signal takes values $u(2) = u(1) = u(0)$, as shown on Fig. 7.



Fig. 7. The output and control signal for Alternative 1

Alternative 2. $Z = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$, therefore the DB control signal takes values $u(1) = u(0)$ and $u(3) = u(2)$, as shown on Fig. 8.

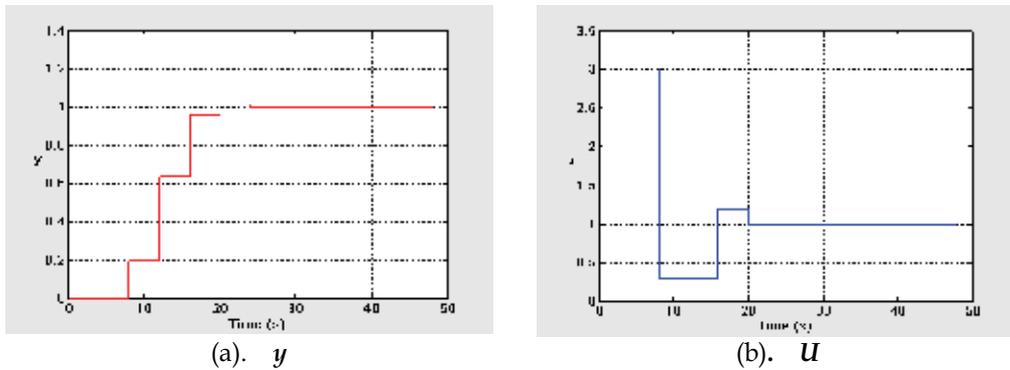


Fig. 8. The output and control signal for Alternative 2

Alternative 3. $Z = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$, therefore $u(1) = u(0)$ and $u(4) = u(3)$.

Alternative 4. $Z = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$, therefore $u(3) = u(2) = u(1)$.

Alternative 5. $Z = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$, therefore $u(2) = u(1)$ and $u(4) = u(3)$.

Alternative 6. $Z = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$, therefore $u(4) = u(3) = u(2)$.

Not all alternatives have the same importance. Those, which hold the initial two values $u(0)$ (particularly!) and/or $u(1)$, because these values contribute significantly to the reduction of the control signal magnitude.

Alternative	$q_0^{(2)}$	$q_1^{(2)}$	$q_2^{(2)}$	$q_3^{(2)}$	$q_4^{(2)}$	$q_5^{(2)}$
1	2.34196	0	0	-3.17382	2.19215	-0.36029
2	3.01725	0	2.72848	0	0.90316	-0.19193
3	3.49041	0	-4.64023	2.22379	0	-0.07397
4	5.13095	-4.50996	0	0	0.49397	-0.11496
5	5.94221	-5.82182	0	0.92321	0	-0.04360
6	7.68261	-9.95441	3.29384	0	0	-0.02204

Table 2. Coefficients in the numerator polynomial of the controller

Alternative	$p_1^{(2)}$	$p_2^{(2)}$	$p_3^{(2)}$	$p_4^{(2)}$	$p_5^{(2)}$
1	0.15281	0.34126	0.38626	0.14674	-0.02708
2	0.19687	0.43966	0.31961	0.05828	-0.01442
3	0.22775	0.50861	0.27290	-0.00370	-0.00556
4	0.33479	0.45338	0.18909	0.03138	-0.00864
5	0.38773	0.48600	0.13173	-0.00218	-0.00328
6	0.50129	0.46995	0.03152	-0.00110	-0.00166

Table 3. Coefficients in the denominator polynomial of the controller

The maximal and minimal values of the control signal during the transient response for the given example are collected in Table 4. The *max...min* span can be used as reference values in a criterion to select the appropriate alternative of the DB controller.

m	0	1	2	2	2	2	2	2
Alternative	1	1	1	2	3	4	5	6
$u_{\max} \equiv u(0)$	9.46	3.78	2.34	3.01	3.49	5.13	5.94	7.68
u_{\min}	-4.72	-2.05	-0.83	0.29	-0.15	0.62	0.12	-2.27

Table 4. Extreme values of the control signal for the considered alternatives

4.3. Block “Selection of controller under control signal constraints”

Each local controller submits its computed control signal to this block. Which of them will be transferred as a global control signal to the plant is selected by checking the conditions of getting control signal within the predefined constraints. For time-invariant plants significant changes in the control signal may be obtained due to rapid change of the reference signal or suddenly appearing “overloading” disturbances. Both these factors can be interpreted as step signals, which appear not so often, such that the controller succeeds to stabilize the controlled variable before the appearance of a new disturbance. This assumption aids the explanation about the nature of the logical decisions about the control signal and its constraints.

Let the step change appears at sampled time k causing a system error $|e(k)| = |r(k) - y(k)| \gg \varepsilon$, where ε is a threshold value determining the sensitivity of the algorithm. The local DB controllers with transfer functions

$$W_c(i; z) = \frac{Q(i; z)}{1 - P(i; z)}, \quad Q(i; z) = \{q_0 + q_1 z^{-1} + \dots + q_{nq} z^{-nq}\}_i, \quad P(i; z) = \{p_1 z^{-1} + \dots + p_{np} z^{-np}\}_i,$$

will yield control signals with extreme magnitudes $u_{i \max}, u_{i \min}, i = 1, 2, \dots, N$, which appear at time $k=0$ and $k=m+1$ according the well known property of DBC that $\sum_{j=0}^k q_j = u(k)$ unit step change of the reference

Therefore, the maximum control signal magnitude at the first sampling instant after the step change ($e(k) > \varepsilon$) is equal to the coefficient q_{i0} , ($\{u(0)\}_i = q_{i0} \equiv u_{i0}$). Having in mind this property, the $u_{i \max}$ value of the local control signal right after the step change from: $u_{i \max} = u(k-1) + (u_{i0} \times e(k))$, $i = 1, 2, \dots, N$. In parallel the $u_{i \min}$ value at the $(m+1)$ -th

sampling instant after the step change of reference from $\left\{ \sum_{j=0}^{m+1} q_j = u(m+1) \right\}_i \equiv u_{i m+1}$ and consequently $u_{i \min} = u(k-1) + (u_{i m+1} \times e(k))$, $i = 1, 2, \dots, N$.

The local DB controller, which complies the constraints, is selected:

- (a) When $e(k) > \varepsilon$ the local controller j among all controllers in the bank is decided according the produced by it maximal value of the control signal, which is less or equal to $u_{\max \lim}$: $u_{j \max} \equiv \max\{u_{1 \max}(k), u_{2 \max}(k), \dots, u_{N \max}(k)\} \leq u_{\max \lim}$. If the additional condition $u_{j \min} \geq u_{\min \lim}$ is satisfied, the selection of controller is confirmed, otherwise first the condition $u_{j \min} \equiv \min\{u_{1 \min}(k), u_{2 \min}(k), \dots, u_{N \min}(k)\} \geq u_{\min \lim}$ is checked and selection confirmed if $u_{j \max} \leq u_{\max \lim}$ is also true.
- (b) When $e(k) < -\varepsilon$ the local controller j is decided by checking first $u_{j \max} \equiv \min\{\text{sign}[e(k)] [u_{1 \max}(k), u_{2 \max}(k), \dots, u_{N \max}(k)]\} \geq u_{\min \lim}$. If the additional

condition $u_{j\min} \geq \text{sign}(\Delta r)u_{\max\text{lim}}$ is satisfied, the selection of controller is confirmed, otherwise first the condition $u_{j\min} \equiv \max\{\text{sign}(\Delta r)[u_{1\min}(k), u_{2\min}(k), \dots, u_{N\min}(k)]\} \leq u_{\max\text{lim}}$ is checked and selection confirmed if $u_{j\max} \leq \text{sign}(\Delta r)u_{\min\text{lim}}$.

4.4. Single-rate MMDBC under control signal constraint – a test example

Let us take the same continuous control plant given in Section 3.3 and formulate MMDBC containing multiple DBC tuned for the same sampled plant model, but, contrarily to the previous case, having different increments of the order, i.e. each DBC is $\text{DB}(3+m,1)$, $m=0, 1, 2, 3, 4, 5$. The DBC producing extreme values of the control signal within the predefined constraints is activated currently within the MMDBC. Figure 9 represents the performance of the system, where the reference and system output are compared on Fig. 9a, the control signal all the time being within the constraints $[-1, 7.5]$ on Fig. 9b. Figure 9c shows how the DBC order increment m varied when stepwise changing the reference signal.

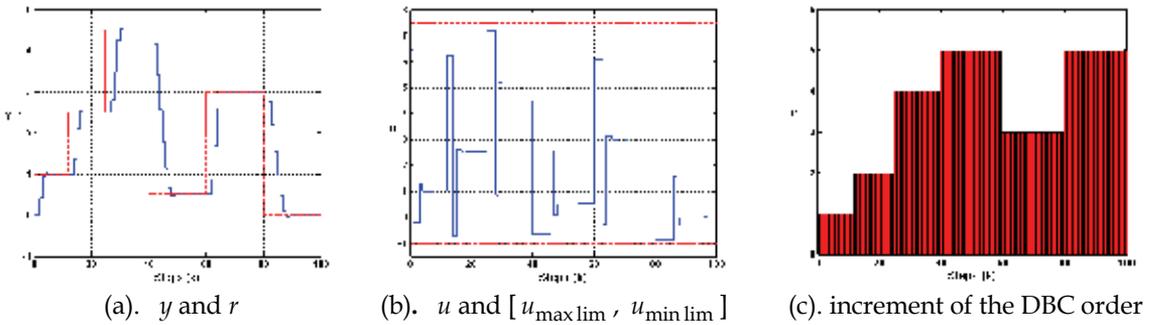


Fig. 9. Single-rate MMDBC control system with control signal constraints

For comparison of the proposed MMDBC, a standard DB control system with fixed increments, namely $\text{DBC}(3+0,1)$ and $\text{DBC}(3+1,1)$, is demonstrated on Fig. 10, where one can see worse performance under the same test conditions. This confirms the advantage of our MMDBC approach.

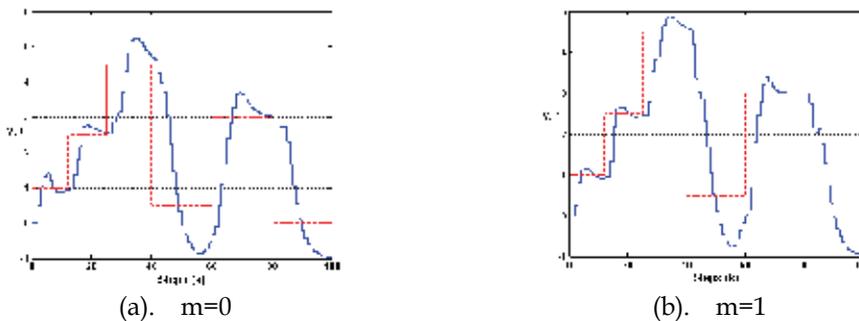


Fig. 10. System behavior y and r in control system with a standard fixed order DBC

4.5. Design of two-rate DB control system

The assumption in the sampled-data control systems theory is to define a sampling period T_0 , which is valid for the entire closed-loop system (let call it T_0^{CL}) and for the controller itself (let call it T_0^C). In other words $T_0^{CL} = T_0^C = T_0$. It is known that $T_0^{CL} = T_0$ should be small enough for achieving nearly continuous-time system behavior, or at least the Shannon sampling theorem should be satisfied. However, it is also known that a small sampling period $T_0^C = T_0$ yields large magnitudes of the control signal, which go beyond the physical constraints of the control valve, i.e. the nonlinear nature of the system becomes dominating. Therefore, certain lower bound of $T_0^C = T_0$ should be considered.

Garipov proposed in [Garipov, 2004] a control scheme for DB control of a continuous plant based on the following postulates:

- First, in order to form a sampled control signal for the continuous plant with a sampling period $T_0^C > \tau$ identical to the sampling period of the entire control system, the system error at the controller input to be sampled with the same sampling period as the controller is sampled. This means the control feedback has to be implemented in an inner closed loop containing the controller and a sampled-time model of the plant, both with the same sampling period. In other words, both blocks have to operate with synchronous sampling rate. A normal performance of the DBC is expected even when the sampling period of the controller is $T_0^P > \tau$.
- Second, in order to close the loop around the physical plant, an outer feedback loop is provided as well, based on the mismatch between the physical plant output (with all types of disturbances, measurement noise, etc.) and a sampled model of the plant (which is noise-free). The mismatch error could be considered much closer to zero when the sampling period of this second model of the plant is very small (nearly continuous system). This implies the recommendation the closed-loop sampling period to be selected always smaller, i.e. $T_0^{CL} < T_0^C$. A normal performance of the DBC in this two-rate control system is expected.

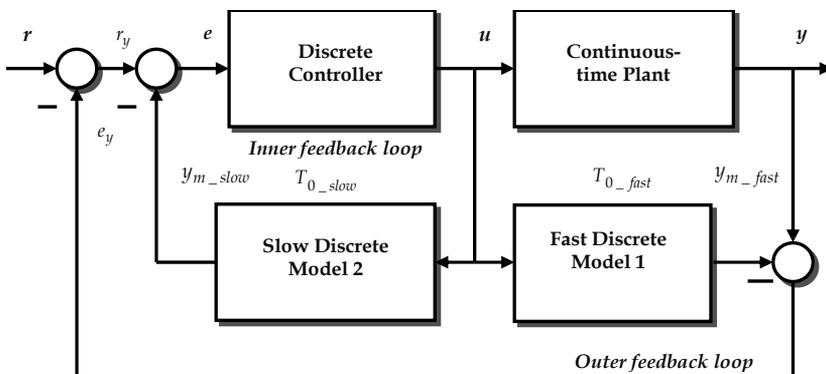


Fig. 11. Two-rate control system

The block diagram of two-rate control system is shown on Fig. 11. Two sampled-time models of the same continuous-time plant are included, namely Fast Discrete Model with sampling period $T_{0_fast} \equiv T_0^{CL}$, which is in parallel to the plant, and Slow Discrete Model with sampling period $T_{0_slow} \equiv T_0^C > T_{0_fast}$, which is used for the design of the controller. The step response of such system is demonstrated on Fig. 12, where one can identify the small sampling period $T_0^{CL} = 1$ sec, while the controller operates at sampling period $T_0^C = 8$ sec.

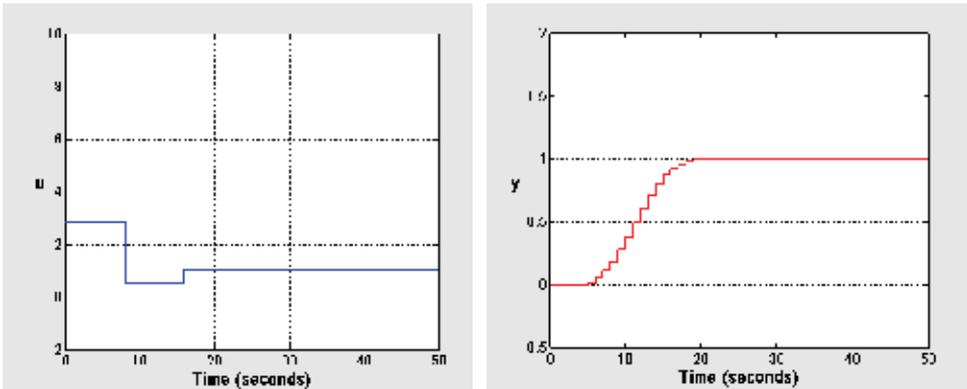


Fig. 12. Modified two-rate system for $T_0^{CL} = 1$ s and $T_0^C = 8 * T_0^{CL} = 8$ s

4.6. Design of a multi-rate DBC

The main idea of a multiple-model adaptive DB controller, in which every local DBC operates at a different sampling period that is not equal of the sampling period of the closed-loop system, is implemented in the block diagram on Fig. 13.

Algorithm

- Step 1. The continuous-time plant model is identified.
- Step 2. The small sampling period T_0 of the control system is selected and the respective sampled-time plant model obtained.
- Step 3. A number of sampled-time models of the plant with different sampling periods $T_0^{(i)}$, $i = 1, 2, \dots, N$, are computed and the corresponding DBC obtained. The respective extreme values of the control signal for each model are calculated.
- Step 4. The control signal constraints $[u_{\min \text{ lim}}, u_{\max \text{ lim}}]$ are defined and desired profile of the reference signal $r(k)$, $k=0, 1, 2, \dots, M$, is specified (for analysis of the system performance).

Step 5. The MMDBC system is started, then the described in Section 4.3 block “Selection of controller” at every step change of the reference signal checks and selects the DBC with the least sampling period providing extreme values of the control signal within the defined constraints to be active.

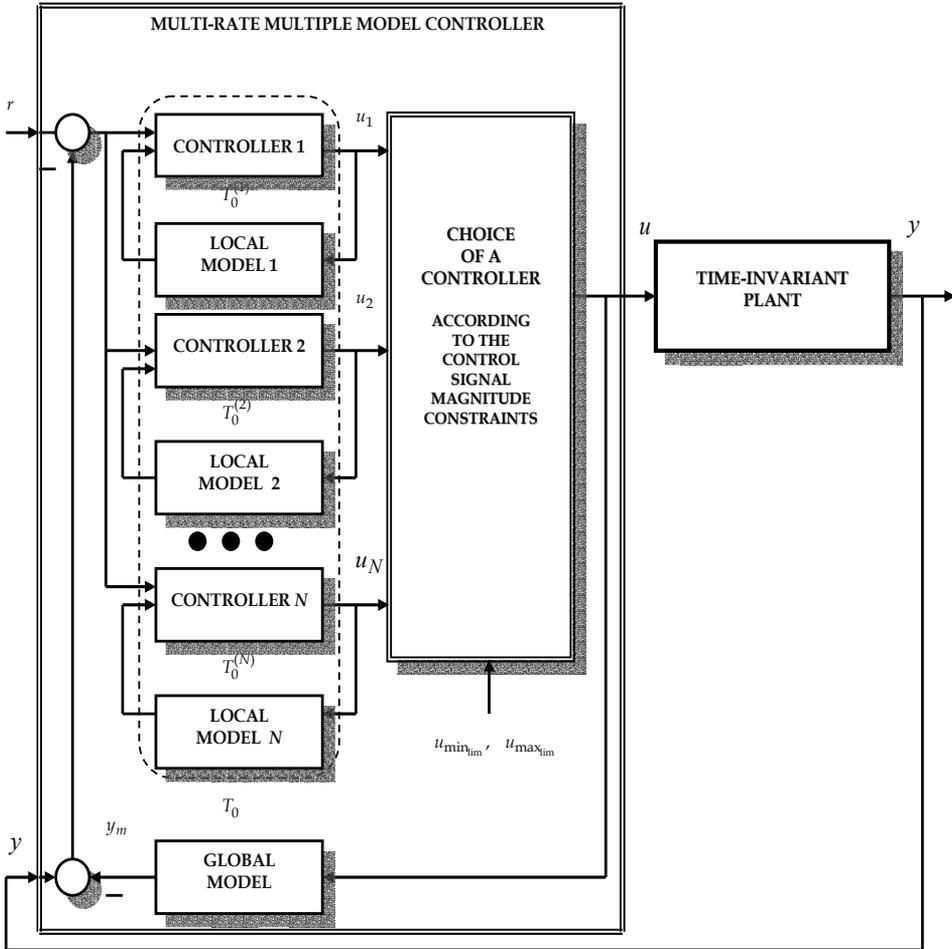


Fig. 13. Multi-rate multiple model control with control signal constraints

4.7. Multi-rate MMDBC under control signal constraint – a test example

Let us take the same continuous control plant given in Section 3.3 and formulate multi-rate MMDBC containing multiple DBCs tuned for the same sampled plant model, but, contrarily to the previous case, obtained at different sampling periods $T_0^{(i)} = 4, 6, 8, 10, 12, 14$ и 18 sec.

The Fast Discrete Model (Fig. 11) is obtained for sampling period $T_0 = 0.1$ sec. Figure 14 represents the performance of the system, where the reference and system output are

compared on Fig. 14a, the control signal all the time being within the constraints $[-1, 7.5]$ on Fig. 14b. Figure 14c shows which model and respective DBC was selected, namely the corresponding value of the sampling period T_0 when stepwise changing the reference signal.

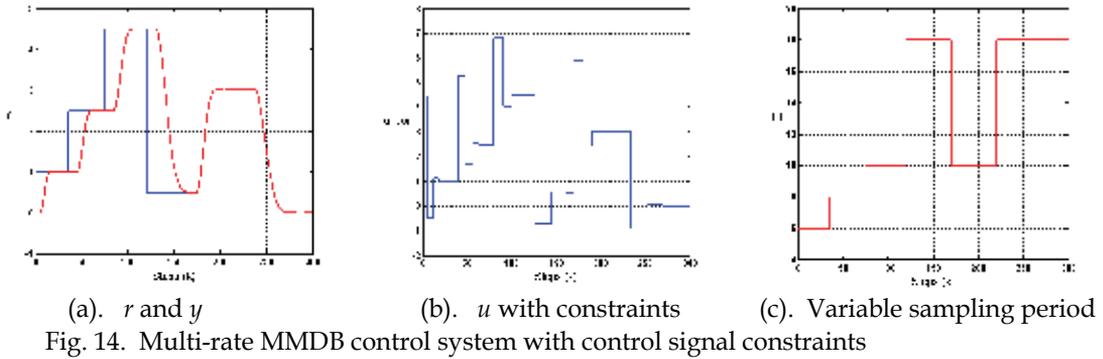


Fig. 14. Multi-rate MMDB control system with control signal constraints

For a comparison, a standard DB control system is demonstrated on Fig. 15 for three different sampling periods $T_0^{(i)} = T_0$, where one can see the poor performance under the same test conditions. This confirms the advantage of our multi-rate MMDBC approach.

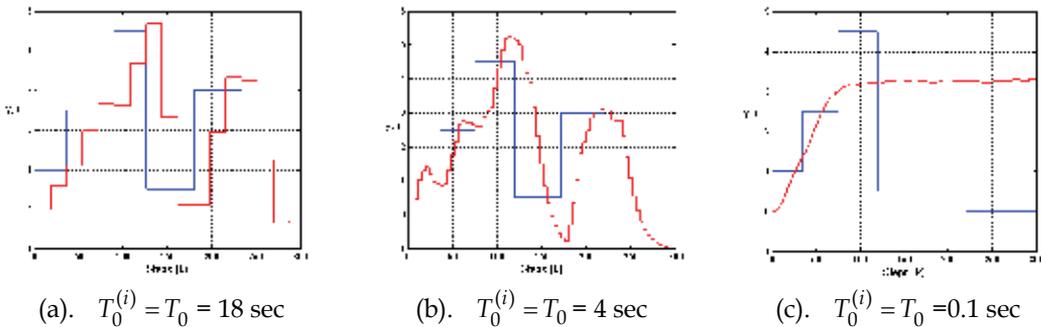


Fig. 15. System behavior y and r in control system with a standard DBC at various T_0

5. Conclusion

The essence of the ideas applied to this text consists in the development of the strategy for control of the arbitrary in complexity continuous plant by means of a set of discrete time-invariant linear controllers. Their number and tuned parameters correspond to the number and parameters of the linear time-invariant regressive models in the model bank, which approximate the complex plant dynamics in different operating points. Described strategy is known as Multiple Regressive Model Adaptive Control (MRMAC), and the implemented control system is known as Multiple Regressive Model Adaptive Control System (MRMACS). Its scheme is very traditional but attention is paid mainly on the novel

algorithm in the supervisory block that forms the final control action if the regimes of the plant are not previously known.

The existence of control signal constraints by the control valve clearly indicates the needs to guarantee a control magnitude that always fits within the control constraints for all operating regime of the system. A novel design procedure is proposed to tune dead-beat controllers (DB) with arbitrarily increased order so the closer is the operating point to the constraints the bigger should be the DB controller order. As the plant operating point continuously changes, the switched MRMAC DB controller of minimal order have to be select in order to satisfy the control signal constraints. The supervisor logical action is shown and tests are made for complex simulation plants. The same task can be solved if the DB order remains fixed but the control signal magnitude is reduced by the switched MRMAC DB controller of arbitrarily discrete time-interval in order to satisfy the control signal constraints. In this case a new multi-rate MRMACS scheme is prepared.

6. References

- Anderson, B. D. O.; Brinsmead, T. S.; Liberzon, D. & Morse, A. S. (2001). Multiple Model Adaptive Control with Safe Switching, *International Journal of Adaptive Control and Signal Processing*, August, 446 – 470.
- Blom, H. & Bar-Shalom, Y. (1988). The IMM Algorithms for Systems with Markovian Switching Coefficients, *IEEE Trans. AC.*, 33, 780-783.
- Boling, J.; Seborg, D. & Hespanha, J. (2003) *Multi-model Adaptive Control of a Simulated PH Neutralization Process*, Elsevier Science, Preprint, February 2003.
- Garipov, E. & Kalaykov, I. (1991). Design of a class robust selftuning controllers. *IFAC Symposium on Design Methods of Control Systems*. Zurich, Switzerland, 4-6 September 1991, 402-407.
- Garipov, E. (2004). Modified schemes of hybrid control systems with deadbeat controllers. *Proc. Int. Conf. "A&I'2004"*, 6-8 October, Sofia, 143-146 (in Bulgarian).
- Garipov, E.; Stoilkov, T. & Kalaykov, I. (2007). Multiple-model Deadbeat Controller in Case of Control Signal Constraints. *IV Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO'07)*, Angers, France, May 9-12.
- Henk, A.; Blom, P. & Bar-Shalom, Y. (1988). The Interacting Multiple Model Algorithm for Systems with Markovian Switching Coefficients, *IEEE Trans. AC*, 33, 8, 780-783 .
- Hespanha, J. P. & Morse, A. S. (2002). Switching Between Stabilizing Controllers, *Automatica*, 38, 11, 1905 – 1917.
- Hespanha, J. P.; Liberzon, D. & Morse, A. S. (2003). Hysteresis-Based Supervisory Control of Uncertain Linear Systems, *Automatica*, 39, 2, 263 – 272.
- Isermann, R. (1981). *Digital Control Systems*. Springer Verlag, Berlin, ISBN: 0387107282
- Li, X.R. & He, Ch. (1999). Model-Set Choice for Multiple-Model Estimation, *Proc. 1999 14th World Congress of IFAC*, Beijing, 169-174.
- Li, X.R. & Bar-Shalom Y. (1996). Multiple-Model Estimation with Variable Structure, *IEEE Trans. AC*, 41, 4, 478-493.
- Maybeck, P. & Hentz, K. P. (1987). Investigation of Moving-bank Multiple Model Adaptive Algorithms, *AIAA Journal of Guidance, Control and Dynamics*, 10, 1, 90-96.

- Morse, A. S. (1996). Supervisory control of families of linear set-point controllers: Part 1: exact matching. *IEEE Trans. AC*, 41, 10, 1413-1431.
- Murray-Smith (Ed.), R. & Johansen (Ed.), T. (1997). *Multiple Model Approaches to Modelling and Control*, Taylor&Francis, London, ISBN: 074840595X.
- Narendra, S. & Balakrishnan, J. (1997). Adaptive Control Using Multiple Models, *IEEE Trans. AC*, 42, 2, 171-187.
- Wenk, C. & Bar-Shalom, Y., (1980). A multiple Model Adaptive Dual Control Algorithm for Stochastic Systems with Unknown Parameters, *IEEE Trans. AC*, 25, 8, 703-710.

Block-synchronous harmonic control for scalable trajectory planning

Bernard Girau[•], Amine Boumaza[•], Bruno Scherrer[•], Cesar Torres-Huitzil^{*}

[•] LORIA – INRIA Nancy Grand Est, France

^{*} Polytechnic University of Victoria, Mexico

1. Introduction

Trajectory planning consists in finding a way to get from a starting position to a goal position while avoiding obstacles within a given environment or navigation space. Harmonic functions may be used as potential fields for trajectory planning (Connolly et al., 1990). Such functions do not have local extrema (unlike other potential based methods as in (Khatib, 1986)), so that control algorithms may reduce to *locally* ascend the potential until they reach a *global* maximum, when obstacles correspond to minima of the potential and goals correspond to maxima.

Harmonic control has had some impact on the robotics community (Masoud & Masoud, 2002; Zelek, 1998; Alvarez et al., 2003; Feder & Slotine, 1997; Huber et al., 1996; Kazemi et al., 2005; Sweeney et al., 2003; Wang & Chirikjian, 2000). Nevertheless, very few hardware implementations have been proposed. They are usually analog, therefore they suffer from a very long and complex design process, and a lack of flexibility (environment size, precision). This chapter presents a parallel hardware implementation of this navigation method on reconfigurable digital circuits. Trajectories are estimated after the iterated computation of the harmonic function, given the goal and obstacle positions of the navigation problem. The proposed implementation locally computes the direction to choose to get to the goal at any point of the environment. Dynamic changes in this environment may be taken into account, for example when obstacles are discovered during an on-line exploration.

To fit real-world applications, our implementation has been designed to deal with very large environments while optimizing computation time. To do so, iterated updates are performed in a block-synchronous mode that takes advantage of large embedded SRAM memory resources. The proposed implementation maps the massively distributed structure of the space-discretized estimation of the harmonic function onto the circuit. When the size of the environment of navigation exceeds the resources available on the circuit, this environment is split into several so-called blocks.

For each block, an area-saving serial arithmetic is used within a global scheme that simultaneously ensures pipelining and parallelism of the iterative computations. These

computations are scheduled for the different blocks so as to make a compromise between pipelining efficiency and block coherency. Moreover, an increasing precision is used throughout the convergence process, so as to further optimize computing times. Thanks to this increasing precision of the chosen serial arithmetic, first iterated updates are faster, the convergence of each block is reached sooner, and the serial detection of this convergence is also faster. When the whole process has converged for a given precision, iterated updates start again with an increased precision. This process is repeated until the necessary precision of the harmonic function estimation is reached.

Our implementation is able to handle up to 16 blocks of size 96×48 (73728 nodes) with a 64-bit precision on a single FPGA Xilinx Virtex XC4VLX160. The main module consists of an array of identical nodes that compute the iterated updates of the harmonic function estimation. Specialized modules have been designed to handle communications between contiguous blocks. Convergence detection is performed in a pyramidal way. All nodes communicate with a decision module that computes the navigation direction by means of a local interpolation of the discretized harmonic function. Obstacles and goals may be dynamically added and memorized for any block, resulting in a new iterated process to update the harmonic function estimation.

Besides all implementation works, we carefully justify our algorithmic and technological choices through both theoretical and empirical studies of the required precisions and convergence times. We study the asymptotic convergence of the proposed algorithmic scheme and we analytically derive some bounds on its rate of convergence. The implementation results together with this theoretical analysis show that the proposed architecture simultaneously improves speed, power consumption, precision, and allows to tackle large environment sizes.

Section 2 describes the principles of harmonic functions and of their use for trajectory planning. Section 3 summarizes the advantages of hardware parallel implementations on FPGAs for embedded navigation in autonomous systems, and it justifies the choice of serial arithmetics, especially with respect to the precision requirements of harmonic control. Section 4 defines the block-synchronous computation scheduling, and its optimization analysis through theoretical and experimental results. The proposed hardware architecture and its implementation results are described in section 5.

2. Harmonic control

In this part, we begin by a brief reminder of harmonic functions and some of their properties. Then we discuss their application to the navigation problem.

2.1 Harmonic functions

Let Ω be an open subset of \mathfrak{R}^n , $\partial\Omega$ its boundary and $\overline{\Omega}$ its closure such that $\overline{\Omega} = \Omega \cup \partial\Omega$.

Definition Let $u : \overline{\Omega} \rightarrow \mathfrak{R}$ be a real function, twice continuously differentiable, and

$\Omega \subset \mathfrak{R}^n$ with $n > 1$. Function u is harmonic iff $\Delta u = \sum_{i=1}^n \frac{\partial^2 u}{\partial x_i^2} = 0$ (Laplace's equation).

Harmonic functions satisfy interesting properties:

- The maximum principle states that: if u is a non-constant continuous function on $\overline{\Omega}$ that is harmonic on Ω , then u attains its maximum and minimum values over $\overline{\Omega}$ on the boundary $\partial\Omega$.
- Applying the divergence theorem on harmonic functions, the following equation holds: $\int_s \overrightarrow{\nabla} u \cdot \vec{n} ds = 0$

where s is the boundary of a closed region strictly in Ω and \vec{n} is a normal vector of s . This equation expresses that the normal flow of the gradient vector field through the region bounded by s is zero. It follows that there can be no local minimum or maximum of the potential inside a bounded region of Ω .

2.2 Application to navigation

To solve the navigation problem using harmonic functions, we consider the problem as a Dirichlet problem: its solution is to find the function u that is harmonic on Ω (navigation space) and that satisfies boundary conditions on $\partial\Omega$ (obstacles and navigation goal):

$$\begin{cases} \forall x \in \Omega, & \Delta u(x) = 0 \\ \forall x \in \partial\Omega, & u(x) = g(x) \end{cases} \quad (1)$$

where the function $g: \partial\Omega \rightarrow \mathfrak{R}$ represents boundary conditions on $\partial\Omega$. These conditions define the values of the navigation function on obstacles and goals. Without loss of generality we choose $g(x) = 0$ for obstacles and $g(x) = 1$ for goals. Solving the Dirichlet problem consists in finding the function u that is harmonic on Ω and that has value 0 on obstacle positions and value 1 on goal positions.

The navigation problem is then solved as follows: a simple ascent along the gradient of u provides a trajectory towards a given goal from any starting position. The properties of harmonic functions ensure that such a path exists and it is free of local optima.

2.2.1 Numerical method to solve Laplace's equation

In this part we consider the case where $\Omega = \mathfrak{R}^2$. Traditionally, Laplace's equation is solved using methods from finite differences on a regular grid (discrete sampling of Ω). Using a Taylor approximation of the second derivatives we obtain the following discrete form of $u(x_1, x_2)$

$$0 = \Delta_{\delta} u(x_1, x_2) = \frac{1}{\delta^2} [u(x_1 + \delta, x_2) + u(x_1 - \delta, x_2) + u(x_1, x_2 + \delta) + u(x_1, x_2 - \delta) - 4u(x_1, x_2)] \quad (2)$$

where δ is the sampling of the grid nodes that represents Ω . In this form, the equation can be solved using relaxation methods such as Jacobi or Gauss-Seidel whose principle is to iteratively replace each node value with the simple average of its four neighbours. Figure 1 shows different trajectories generated by simulations using this numerical scheme.

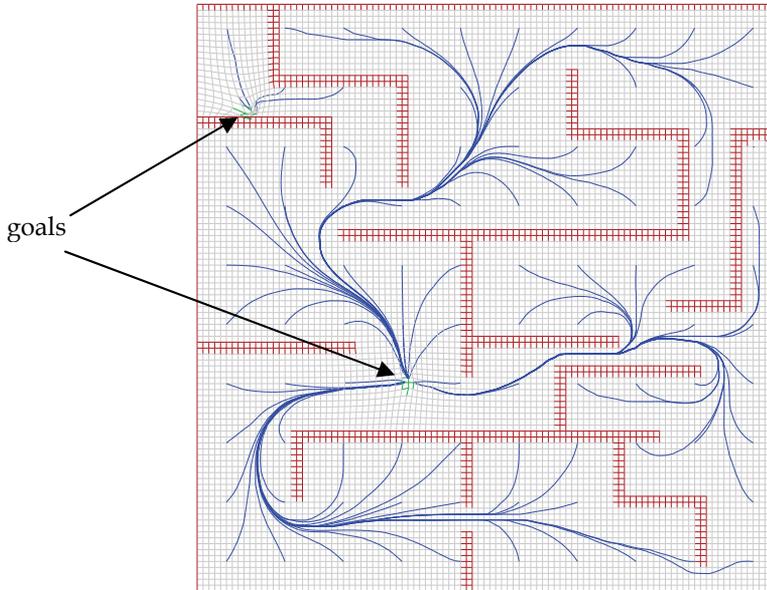


Fig. 1. Trajectories generated by harmonic control (100x100 grid, equally spread starting nodes, two goals)

2.2.2 Properties of harmonic navigation functions

Harmonic navigation functions have many interesting properties which motivated their use in numerous applications especially in robotics (Tarassenko & Blake, 1991; Connolly & Grupen, 1993; Masoud & Masoud, 2002; Zelek, 1998; Alvarez et al., 2003; Feder & Slotine, 1997; Huber et al., 1996; Kazemi et al., 2005; Sweeney et al., 2003; Wang & Chirikjian, 2000):

Global navigation: Complete trajectories may be generated towards a goal position from anywhere in the environment, since there are no local minima.

Dynamic trajectory planning: Unexpected updates of the environment may be taken into account, since harmonic functions are computed by iterative relaxation methods. Therefore newly detected obstacles may be integrated in the model as new boundary conditions during computation, so that harmonic control is available in dynamic environments or in environments explored on-line (Zelek, 1998; Boumaza & Louchet, 2003).

Parallel computation: An interesting property of the computations described above is their massively parallel distribution. Computing node values only requires local information of the neighbouring nodes. Fine-grain parallel implementations appear as an opportunity, as discussed in the next section.

3. Towards a rapid and scalable implementation

The results shown in figure 1 were obtained from software simulations carried out on a PC. The aim of our work is to design an embedded system for robot navigation. Computation speed is not the only criterion (trajectory decision must be performed in real time). Power consumption is also essential for autonomy. Besides, computation precision and scalability appear as critical issues for harmonic control, as discussed below. These combined aspects motivate the design of a parallel hardware implementation. In such a work, the number of inputs/outputs and above all the level of parallelism have a direct influence on the obtained implementation consumption and speed. A massively parallel implementation is a real challenge, taking into account constraints such as precision, grid size, dynamic updates, etc.

3.1 Implementation environment

Since the appearance of configurable circuits, such as *field programmable gate arrays* (FPGAs), algorithms may be implemented on fast integrated circuits with software-like design principles. VLSI designs lead to very high performances. But the time needed to realize an ASIC (application specific integrated circuit) is too long, especially when different configurations must be tested. The chip production time may take up to 6 months.

FPGAs, such as Xilinx FGPA (Xilinx, 2000), are based on a matrix of *configurable logic blocks* (CLBs) that contain a few logic cells. Each logic cell is able to implement small logic functions (4 or 5 inputs) with a few elementary memory devices (flip-flops or latches) and some multiplexors. Each logic cell is independently programmable. Similarly, the routing structure that connects the logic cells as well as the CLBs can be configured. A FPGA approach simply adapts to the handled application, whereas a usual VLSI implementation requires costly rebuildings of the whole circuit when changing some characteristics. A design on FPGAs requires the description of several operating blocks. Then the control and the communication schemes are added to the description, and an automatic "compiling" tool maps the described circuit onto the chip.

3.2 Technological choices

When a massively distributed model is mapped onto a FPGA, the main issues are the huge number of operators, and the routing problems due to the dense interconnections. A first standard technological choice to solve these problems is to use serial arithmetics: smaller operators may be implemented, and they require less connection wires, at the cost of several clock cycles to handle each arithmetic operation. Another essential technological choice is to estimate the minimum precision required to keep satisfactory results with as small as possible operators and memory resources.

3.2.1 Serial arithmetics

Serial arithmetics correspond to computation architectures where digits are provided digit after digit. Serial arithmetics lead to operators that need small implementation areas and less inputs/outputs, and that easily handle different precisions, without an excessive increase of the implementation area. Serial systems are characterised by their delay, i.e, the number d such that p digits of the result are deduced from $p+d$ digits of the input values.

Two main kinds of serial arithmetics are available: LSBF or MSBF (least/most significant bit first). Standard serial operators work in a LSBF mode. Though our model requires the

computation of maximum values (gradient ascent, detection of local maxima), that may only be computed in a MSBF mode, we mostly use standard LSBF serial operators to optimize the required area of the main computations¹. Nevertheless, our implementation simultaneously handles a read access in MSBF mode to detect local maxima: since we use dual port SRAM blocks with fully independent ports, we take advantage of two simultaneous read addresses (controlled by two reverse counters in the control modules).

3.2.2 Computation precision

Software simulations are usually performed to study the precision that is required by an application before its hardware implementation. Precision issues appear as a critical problem for harmonic control. It has already been mentioned as a major limitation for analog implementations (Tarassenko & Blake, 1991). Computing a harmonic potential over a large grid may result in gradients too small to use because the allowable precision is easily reached. Connolly (Connolly & Grupen, 1993) proposes a relationship between the precision required for floating point representation on a PC and the number of nodes on the grid. He argues that the precision should at least represent $1/N$ (requiring at least $\log_2 N$ bits), where N is the total number of grid nodes, to circumvent precision problems.

We may first discuss Connolly's estimation from a theoretical point of view. We argue that $1/N$ is not a sufficient precision for some kinds of environments. More precisely, we argue that the precision might have to represent at least $1/2^{O(L)}$ (therefore requiring some $O(L)$ bits), where L is the maximum trajectory length in the environment. To prove this assertion, let us consider a "corridor" of length L and width 1, with an obstacle on the left ($x_0 = 0$), and the goal at the other side ($x_L = 1$). At each intermediate node $0 < i < L$, according to equation (2), the following relation is fulfilled:

$$x_i = \frac{1}{4}(x_{i-1} + x_{i+1})$$

This is a linear recurrent series of order 2. The roots of the associated polynomial are:

$$r_+ = 2 + \sqrt{3} \quad r_- = 2 - \sqrt{3}$$

so that the generic term of the series is

$$x_i = \lambda r_+^i + \mu r_-^i$$

Since $x_0 = 0$ and $x_L = 1$, we finally obtain

$$x_i = \frac{(2 + \sqrt{3})^i - (2 - \sqrt{3})^i}{(2 + \sqrt{3})^L - (2 - \sqrt{3})^L}$$

When $L \rightarrow \infty$:

$$x_1 \approx \frac{2\sqrt{3}}{(2 + \sqrt{3})^L}$$

Since the used precision must at least ensure that $x_0 \neq x_1$, the computation of the harmonic function in such an environment requires at least $O(L)$ bits, as argued above. A similar result

¹ The only existing radix-2 MSBF serial arithmetics is called *on-line* arithmetics. It uses a redundant number representations system, which induces less area-saving operators.

would be obtained in a square grid containing for example a spiral of length L and width 1, which can be obtained with an environment of approximately $2L$ nodes. This further motivates the use of an embeded implementation in which high precisions may be handled.

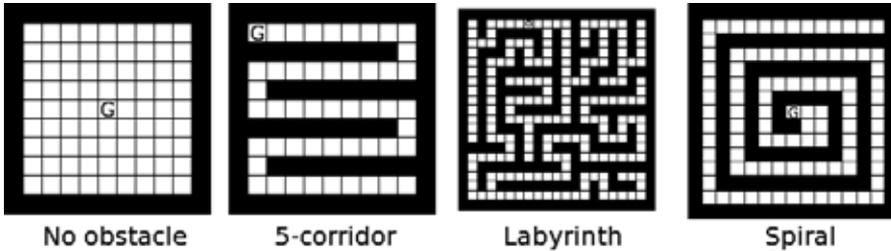


Fig. 2. Four different discrete environments: obstacles are in black, and 'G' denotes the goal.

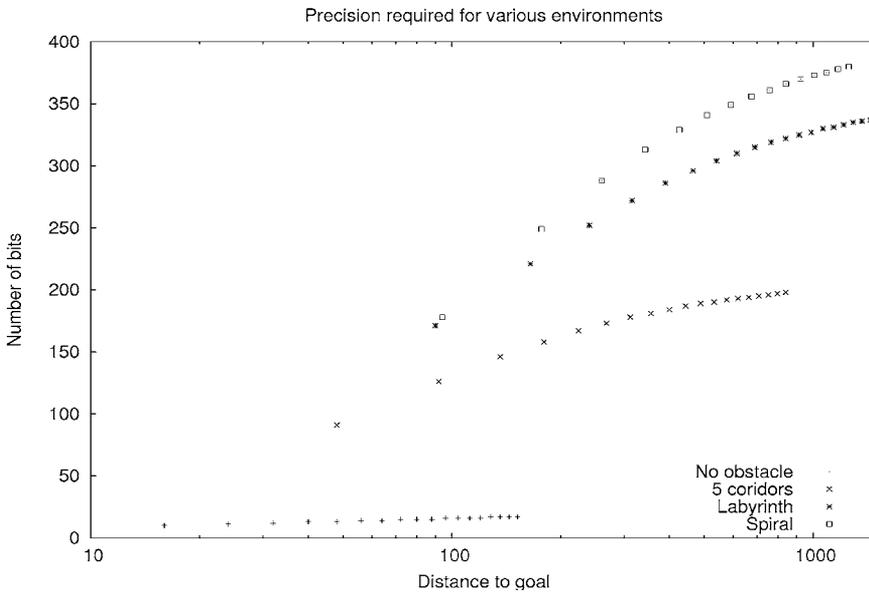


Fig. 3. Number of bits required with respect to the maximal distance L to the goal for different magnifications of the environments of Figure 2. The number of bits grows slightly slower than $O(\log(L))$.

Nevertheless, the study of the precision required by the computation of the harmonic function in a discretized environment should take “likely” environments into account. Therefore we have carried out numerous experiments with large randomly generated mazes. It follows that in most environments, the maximum distance to the goal is close to the square root of the environment size, and that a precision proportional to $1/L$ (i.e. a number of bits proportional to $\log(L)$) is generally sufficient to ensure that the computation of the harmonic function converges such that no local minimum or maximum exists (i.e. a trajectory is found from any node). Figure 2 shows some experimental environments, and figure 3 shows the minimum precision that is required with respect to distance L . It should

be pointed out that in case of insufficient precision, large areas of the grid may share the same value, hence a null gradient that results in incomplete trajectories.

Implementation note

Implementations based on serial arithmetics may be more easily extended to larger precisions than implementations based on parallel arithmetics. Since the size of serial adders and comparators does not depend on precision (unlike multipliers and elementary functions), our implementation may handle large precisions by means of rather simple changes in the control modules.

3.3 Dealing with very large environments

As mentioned above, scalability and precision are key issues for harmonic control in large environments. In this chapter, two major implementation choices deal with these issues.

3.3.1 Block-parallel computation

Despite the technological choices discussed above, the size of the discretized environment we are able to map in a fully parallel way onto FPGAs is limited (around 50x50 nodes in our preliminary work in (Girau & Boumaza, 2007)). To handle much larger environments (or finer discrete resolutions), we propose a block-synchronous (or block-parallel) implementation: the environment is partitioned into several blocks, each block of nodes being handled in a fully parallel way by the FPGA while the different blocks are sequentially handled. In this section, we formalize this computation scheduling, and we analyse both its constraints and its performance so as to optimize our implementation.

Let $N \times M$ be the total number of nodes (location units) in the discretized environment. Let $n \times m$ be the number of nodes that may be simultaneously handled on the FPGA. The environment is partitioned into $B = \frac{N}{n} \times \frac{M}{m}$ blocks of $n \times m$ nodes.

Implementation note

Our implementation performs computations in a block-synchronous way. Moreover up to I consecutive iterations are performed for each block before handling the next block, so as to improve the use of pipelining.

3.3.2 Increasing precision

Possible dynamic updates of the environment require fast computations so that the navigation trajectories adapt to these changes in real-time. Since the computation time in a serial implementation partially depends on the computation precision, and since several iterations are required to let our system converge to a good approximation of the expected harmonic function, we use an increasing precision so as to optimize the convergence time.

The first iterations are performed with a chosen reduced precision. When the whole system has converged for a given precision (for all blocks of nodes), iterated updates start again with an increased precision. This process is repeated until the necessary precision of the harmonic function estimation is reached.

Thanks to this increasing precision of the chosen serial arithmetic, the global computation time of our implementation is optimized:

- first iterations are faster: since they handle reduced precisions with serial operators, the first iterations are faster than with an immediate maximum precision,
- next updates take advantage of a "good" starting point: when the system has converged for a given precision, the next iterations use an increased precision, and the additional convergence time only corresponds to computing the additional bits of the harmonic function estimation,
- the convergence of each block is reached sooner: I consecutive iterations are performed for each block before handling the next block, except if the computations within this block converge before the I iterations, which may happen more rapidly with reduced precisions,
- the serial detection of the convergence of each block is also faster: convergence detection is based on a comparison between the old and new node states, and the comparison time is proportional to the handled precision; moreover, when many blocks have already converged, this earlier convergence detection allows to switch more rapidly to the blocks that still need updates.

4. Optimizing block-synchronous harmonic control

The implementation choices discussed in the previous section lead to a block-parallel algorithm that may be described as follows:

```

p=initial_precision
while (p<=required_precision) /* i.e. local maxima exist */
  converge:=false
  while (not(converge)) do
    converge:=true
    for b:=1 to B do
      i:=0
      block_cvg=false
      while ((i<I) and (not(block_cvg))) do
        block_cvg:=true
        forall nodes (x,y) in block b do
          update node(x,y)
          if (node(x,y) changed)
            block_cvg:=false
          endif
        endforall
        i++
      endwhile
      if (not(block_cvg))
        converge:=false
      endif
    endfor
  endwhile
  increase(p)
endwhile

```

In this section, we study the computation time of this algorithm, so as to optimize l and the increasing scheme of the precision. First of all, we define more precisely the convergence criteria that are used for the different loops of our algorithm. Then we analyze the convergence rate of the estimation method of the harmonic function as discussed in 2.2.1. We show this computation is contracting with a contraction coefficient that is strictly less than 1. We then take into account the error that corresponds to the computation approximations (considering reduced precisions) by analyzing it as a noisy iterative contraction. We finally introduce in this theoretical analysis the block-synchronous approach and the use of increasing precisions through the iterative process.

4.1 What are the convergence criteria ?

In the above algorithm, the main loop stops when the required precision is reached. Though this notion of required precision is studied in 3.2.2 with respect to a static analysis of the harmonic function along a trajectory, the iterated computation of equation (2) requires a dynamic evaluation of the convergence of the whole process. Since we use a block-synchronous version with an increasing precision, different kinds of convergence are used:

- The inner loop that handles the computations for one block uses the stabilization of the node states within the block as convergence criterion.
- Similarly, the stabilization of all blocks is used as convergence criterion in the intermediate loop that performs the computations of all blocks for a given precision.
- The main loop may not know in advance what is the required final precision. The goal of the all process is to provide valid trajectories. This is the case when *no local minimum or maximum exists* in Ω . Therefore we stop the algorithm when no such local extremum is detected.

4.2 How much contracting is the computation of a harmonic function ?

When computing a harmonic function on Ω (inner space) $\cup \partial\Omega$ (boundary), the system we need to solve may be written in matrix form:

$$H = \mathbf{H}H + C$$

where C is a vector that has non-zero values only for points in $\partial\Omega$, and where \mathbf{H} is a matrix that has zero lines for points in $\partial\Omega$ and that only has non-zero values (1/4) on 4 columns on the lines corresponding to Ω points.

Matrix \mathbf{H} is substochastic (coefficients are non-negative and on each row their sum is lower than 1). It corresponds to a (vanishing) random walk on Ω : when a point is inside the domain (in Ω), it moves uniformly towards its 4 neighbours; when a point is on the border (in $\partial\Omega$), it vanishes (its mass disappears at the next instant).

Since \mathbf{H} is substochastic, its eigenvalues take values in $[0,1]$. We now show that 1 cannot be an eigenvalue. To do so, consider a similar substochastic matrix \mathbf{H}' , which corresponds to the Markov chain which only vanishes on the borders of the $N \times M$ rectangular domain. A standard result (see (Young, 1971)) is that the biggest eigenvalue of \mathbf{H}' is

$$\gamma = \frac{1}{2} \left(\cos\left(\frac{\pi}{N}\right) + \cos\left(\frac{\pi}{M}\right) \right)$$

Intuitively, the biggest eigenvalue (and more precisely, its distance to 1) corresponds to the amount by which this Markov chain is vanishing at each time step. As the former Markov chain \mathbf{H} vanishes more than \mathbf{H}' , its eigenvalues should not be greater than γ . More formally, \mathbf{H} is equal to \mathbf{H}' on some lines and null on the others, thus for any vector d ,

$$|\mathbf{H}d| \leq \mathbf{H}|d| \leq \mathbf{H}'|d| \leq \gamma|d|$$

where $|x|$ is the componentwise absolute value of x , and where the first inequality comes from the fact that \mathbf{H} only has non-negative components (it is some sort of generalized triangle inequality). This means that the eigenvalues of \mathbf{H} cannot be greater than γ . As a consequence, the iterative algorithm

$$H_{k+1} \leftarrow \mathbf{H}H_k + C$$

is converging at a linear rate $\leq \gamma$. When N and M are big, we have:

$$\gamma \approx 1 - \frac{\pi^2}{4} \left(\frac{1}{N^2} + \frac{1}{M^2} \right) \quad (3)$$

4.3 General analysis of a noisy iterative contraction fixed point computation

Let us consider a contraction operator \mathbf{O} , with contraction factor at most γ . Let us assume that there is an error bounded by e at each iteration:

$$H_{k+1} \leftarrow \mathbf{A}\mathbf{O}H_k$$

where \mathbf{A} is some approximation operator satisfying:

$$\|\mathbf{A}f - f\| \leq e$$

In our case, \mathbf{A} is related to the round-off error.

4.3.1 Convergence: general case

Unfortunately, the process of mixing a contraction mapping and a round-off does not stabilize in general. For example, if the error is the round-off to the closest integer, let us assume that we have the following contraction (with fixed point 0):

$$x_{k+1} \leftarrow -0.6x_k$$

Then if $x_0 = 1$, we have the following sequence:

$$x_1 = -0.6 \quad \overline{x_1} = -1 \quad x_2 = 0.6 \quad \overline{x_2} = 1$$

in other words we have a loop.

One might think that the problem in the above example is due to the fact that the factor in front of x is negative. But one can also find multidimensional examples where convergence does not occur and which involves only positive terms.

4.3.2 Weak convergence, rate of convergence

Though not converging in general, it can be proved that the above system converges in some weak sense. Namely it leads to oscillations around the fixed point. At each iteration, let us assume:

$$\|H_{k+1} - \mathbf{O}H_k\| = \|\mathbf{A}\mathbf{O}H_k - \mathbf{O}H_k\| \leq e$$

Then, writing H_* the fixed point of \mathbf{O} , we have

$$\begin{aligned} \|H_{k+1} - H_*\| &\leq \|H_{k+1} - \mathbf{O}H_k\| + \|\mathbf{O}H_k - H_*\| && \text{(triangular inequality)} \\ &= \|H_{k+1} - \mathbf{O}H_k\| + \|\mathbf{O}H_k - \mathbf{O}H_*\| && (H_* \text{ fixed point of } \mathbf{O}) \\ &\leq e + \gamma \|H_k - H_*\| && \text{(definition of contraction)} \end{aligned}$$

Thus by induction,

$$\|H_k - H_*\| \leq e + \gamma e + \dots + \gamma^{k-1} e + \gamma^k \|H_0 - H_*\| \leq \frac{e}{1-\gamma} + \gamma^k \|H_0 - H_*\|$$

It shows that even if the process does not converge, it is asymptotically $\frac{e}{1-\gamma}$ -close to H_* .

We may exploit the above equation to derive some rate of convergence:

$$\|H_k - H_*\| \leq \varepsilon \Leftrightarrow \frac{e}{1-\gamma} + \gamma^k \|H_0 - H_*\| \leq \varepsilon \Leftrightarrow k \geq \frac{\log \left(\frac{\varepsilon - \frac{e}{1-\gamma}}{\|H_0 - H_*\|} \right)}{\log \gamma}$$

For instance, for any $\lambda > 0$, taking $\varepsilon = \frac{(1 + \lambda)e}{1 - \gamma}$, we see that:

$$k \geq \frac{\log\left(\frac{\lambda e}{(1 - \gamma)\|H_0 - H_*\|}\right)}{\log \gamma} \Rightarrow \|H_k - H_*\| \leq \frac{(1 + \lambda)e}{1 - \gamma} \quad (4)$$

We may interpret λ as a margin that is added to the $\frac{e}{1 - \gamma}$ -wide asymptotical interval around H_* . This margin defines a wider interval where H_k finally lies.

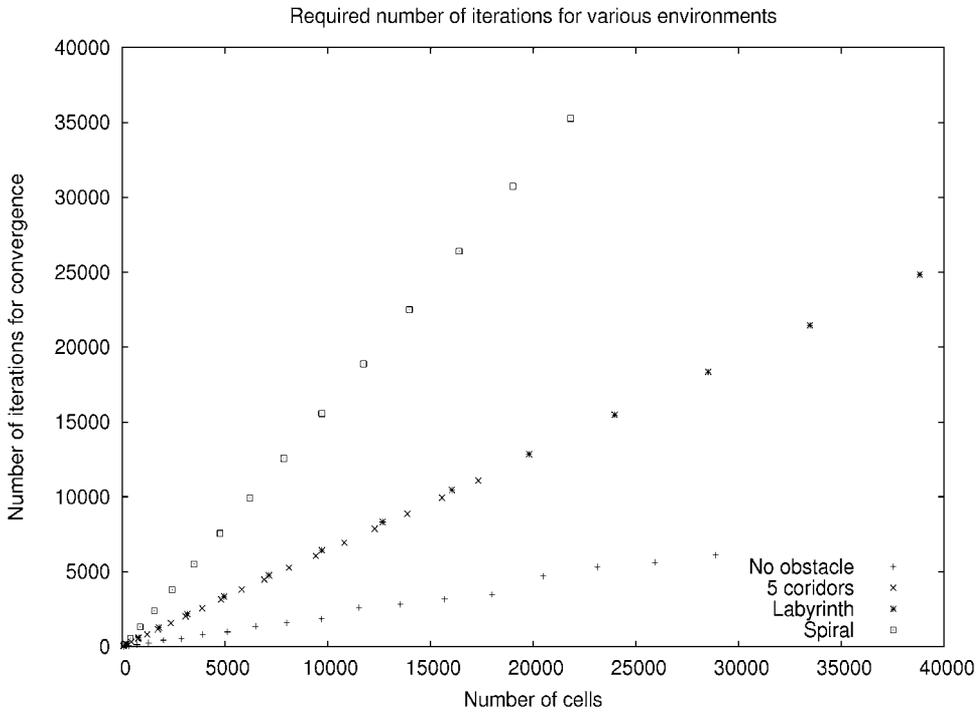


Fig. 4. Number of iterations required for full convergence (stabilization).

4.3.3 Experimental convergence, rate of convergence

Experimentally, the convergence of the iterated computation of the harmonic function (as in equation (2)) not only converges in a weak sense, but fully converges whatever the fixed precision. To validate this assertion, we have carried out numerous experiments with various environments. These experiments are illustrated by figure 4. They establish that the number of required iterations linearly depends on the number of nodes in the environment, which validates the above estimation of k (that depends on $1/\log(\gamma)$ which is roughly proportional to N^2 and M^2).

Implementation note

Following these experiments, we consider the complete stabilization of all bits of all nodes to define the convergence criterion within the inner loops of our algorithm (see 4.1).

4.4 An increasing precision algorithm

We now consider the case where the harmonic function is iteratively computed using arithmetics with different precisions: $e_1 = 2^{-p_1}$, $e_2 = 2^{-p_2}$, ..., $e_T = 2^{-p_T}$. Write $H^{(0)}$ the initial estimate. The analysis of the previous subsection suggests that we can estimate the sequence of potentials $H^{(1)}$, $H^{(2)}$, ..., $H^{(i)}$, ... with respective precisions

$$\frac{(1 + \lambda)e_i}{1 - \gamma} \approx (1 + \lambda)K.2^{-p_i}$$

with the constant

$$K = \frac{1}{1 - \gamma} \approx \frac{4}{\pi^2 \left(\frac{1}{N^2} + \frac{1}{M^2} \right)} \quad (5)$$

for some positive value λ .

4.4.1 Initial and final precisions

At the end, one will have: $\|H^{(T)} - H_*\| \leq (1 + \lambda)K.2^{-p_T}$

The final precision p_T can be set such that the eventual approximation is smaller than some given $\varepsilon > 0$:

$$(1 + \lambda)K.2^{-p_T} \leq \varepsilon \Leftrightarrow p_T \geq \frac{\log(1/\varepsilon) + \log(K) + \log(1 + \lambda)}{\log(2)} = p_{\max}(\varepsilon)$$

The initial precision p_1 should be at least such that $\varepsilon < 1$ (the harmonic function is between 0 and 1), so this means that we should take:

$$p_1 \geq \frac{\log(K) + \log(1 + \lambda)}{\log(2)} = p_{\min}$$

4.4.2 Worst-case analysis

Given the analysis of the previous subsection, estimating $H^{(i)}$ from $H^{(i-1)}$ with precision $\frac{(1 + \lambda)e_i}{1 - \gamma}$ will be done in less than k_i iterations with:

$$k_i = \frac{\log\left(\frac{\lambda e_i}{(1-\gamma)\|H^{(i-1)} - H_*\|}\right)}{\log \gamma} = \frac{\log\left(\frac{\lambda K \cdot 2^{-p_i}}{\|H^{(i-1)} - H_*\|}\right)}{\log \gamma}$$

Since $\log(\gamma) \approx \gamma - 1 = -1/K$, we finally get:

$$k_i \approx K\left(p_i \log(2) + \log(1/\lambda) + \log(1/K) + \log\left(\|H^{(i-1)} - H_*\|\right)\right)$$

Remark: $\log(1/K)$ tends to $-\infty$ when the environment gets bigger, but it is compensated by the fact that $p_i > \log(K)/\log(2)$ (see previous subsection).

The first number of iterations k_1 depends on the initial value $H^{(0)}$:

$$k_1 \approx K\left(p_1 \log(2) + \log(1/\lambda) + \log(1/K) + \log\left(\|H^{(0)} - H_*\|\right)\right)$$

The subsequent numbers of iterations depend on the previous precision: for $i \geq 2$

$$\begin{aligned} k_i &\approx K\left(p_i \log(2) + \log(1/\lambda) + \log(1/K) + \log\left(\|H^{(i-1)} - H_*\|\right)\right) \\ &\leq K\left(p_i \log(2) + \log(1/\lambda) + \log(1/K) + \log(e_{i-1})\right) \\ &= K\left((p_i - p_{i-1} + 1) \log(2) + \log(1/\lambda)\right) \end{aligned}$$

To summarize, k_1 may be estimated as an affine function of the initial number of bits p_1 , and each k_i may be estimated as an affine function of the precision increase (in terms of number of added bits). Both affine functions share the same linear coefficient $K \cdot \log(2)$.

$$k_1 = a + b p_1 \qquad k_i = a' + b(p_i - p_{i-1})$$

Let us now assume that the computation time of equation (2) is also an affine function of the number of bits $\alpha + \beta p$, when very large precisions are used. It is for example the case on a standard processor for which vectors of integers code for very large precision numbers, and it is of course the case with hardware serial operators. Then the global computation time when the final precision is used from the beginning is:

$$(a + b p_{\max}) \cdot (\alpha + \beta p_{\max}) = b \beta p_{\max}^2 + o(p_{\max}^2)$$

Whereas the global computation time with increasing precision is

$$(a + bp_1).(\alpha + \beta p_1) + \sum_{i=2}^T (a' + b(p_i - p_{i-1})).(\alpha + \beta p_i) \quad (6)$$

If the increase of the precision is such that the number of bits is an arithmetic series $p_i = i \frac{P_{\max}}{T}$, then equation (6) becomes

$$\begin{aligned} & (a + b \frac{P_{\max}}{T}).(\alpha + \beta \frac{P_{\max}}{T}) + \sum_{i=2}^T (a' + b \frac{P_{\max}}{T}).(\alpha + \beta i \frac{P_{\max}}{T}) \\ & = p_{\max}^2 \left(\frac{b\beta}{T^2} \left(1 + \sum_{i=2}^T i \right) + o(1) \right) = \frac{1}{2} b\beta p_{\max}^2 \left(1 + \frac{1}{T} \right) + o(p_{\max}^2) \end{aligned}$$

This shows that the increasing precision approach roughly divides the computation time by 2 when very large precisions are required (therefore when large environments are handled, as discussed in 3.2.2).

As such, this result holds without block-synchronous. The next subsection introduces the block-synchronous aspect to prove the relevance of the increasing precision approach within our main algorithm.

4.5 A block-synchronous algorithm with increasing precision

Let us now take into account the fact that an environment may be too large to fit within one single FPGA. Then one has to partition this environment into B blocks, as explained before. Since our FPGA implementation uses serial arithmetics with interleaved loops, its computation time for k iterations at precision $e = 2^{-p}$ is

$$\tau k (p + d)$$

where d is the serial delay for the computation of equation (2), and τ is its latency.

We can conclude that the overall time without increasing precision is:

$$B \tau ((a + bp_{\max})(p_{\max} + d))$$

4.5.1 Introducing an increasing precision

Following 4.4.2, we now assume that our implementation fully applies the algorithm depicted at the beginning of section 4, with I consecutive iterations for each block to ensure pipelining. We first consider that these consecutive iterations do not modify the total

number of iterations to perform for each precision (see below for an empirical study of this assertion). Assuming again that the number of bits is an arithmetic series, it follows that the overall computation time now is:

$$\begin{aligned}
 & B \tau \frac{(a + b \frac{P_{\max}}{T})}{I} \left(I \left(\frac{P_{\max} + d}{T} \right) \right) \\
 & + B \tau \sum_{i=2}^T \frac{(a' + b \frac{P_{\max}}{T})}{I} \left(I \left(i \frac{P_{\max} + d}{T} \right) \right)
 \end{aligned} \tag{7}$$

As in 4.4.2, this equation leads to roughly divide the computation time by 2 thanks to the increasing precision. Yet it does not appear as obvious that the I consecutive iterations provide a significant improvement (in the above equation, it just appears as a way to minimize the effect of the delay). But the above estimation considers that all I iterations are performed for all blocks in the algorithm. This is highly pessimistic, as shown below.

4.5.2 Taking advantage of “still” blocks

The above notion of pipelined interleaved loops within each block corresponds to the most inner loop of our algorithm:

```
while ((i<I) and (not(block_cvg))) do
```

Therefore, I iterations are performed only for blocks that have not converged so far. In most experiments, it clearly appears that large parts of the environment stay unchanged (still) for several iterations while distant blocks slowly propagate the changes. This is even more true when small precisions are handled.

We have performed experiments on a PC in order to compare the overall number of computations when one uses several blocks with detection of early stabilization within each block, to the case where there is only one block. These experiments are reported in figure 5 for the “Labyrinth” environment. This figure illustrates the speed-up obtained with different block sizes over the purely synchronous case. The curves show the ratio between the number of iterations needed to converge in the asynchronous case and the number of iterations needed to converge in the block-synchronous case with respect to the maximum number of iterations by block I . The experiments show that partitioning by blocks speeds up the computation time, although the successive iterations are performed by a block without updating the neighbouring blocks. This speedup is observed provided that I is not too large and an early detection of block stabilization is performed. Similar results have been obtained for all kinds of environments.

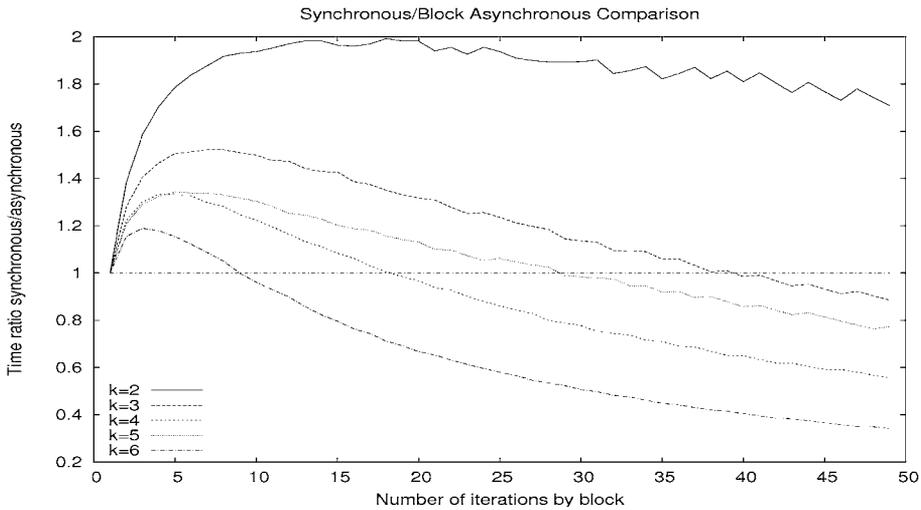


Fig. 5. Comparison between asynchronous and block-synchronous overall computation time. Environment: labyrinth. The total number of blocks is indicated by 'k': for example when 'k=4', there are B=4x4 blocks.

Figure 5 also shows that the speed-up appears greater when the block size is larger. This is confirmed by another series of experiments that are illustrated by figure 6. This figure shows the evolution of the ratio \hat{I}/I until convergence for different values of B on the same environment (labyrinth), where \hat{I} is the average number of iterations performed by the blocks before early detection of stabilization. The experiments show that the smaller the blocks, for example 6x6 (k=6), the larger the number of iterations required. They also show that \hat{I} rapidly reaches a constant value until convergence.

Following our experiments, it finally appears that the block-synchronous approach that uses an increasing precision divides the computation time by some coefficient between 2 and 4.

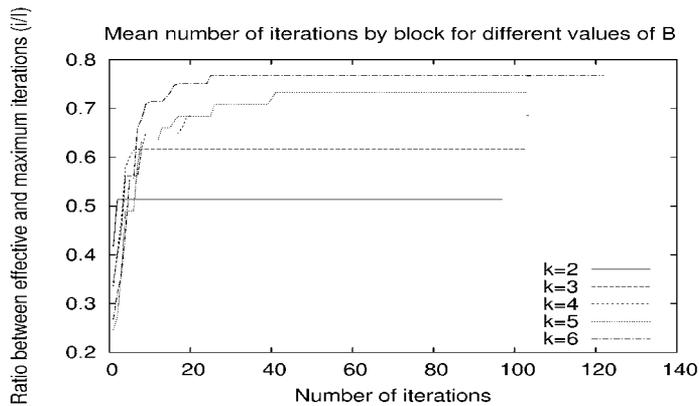


Fig. 6. Evolution of the number of successive iterations performed by the blocks before early detection of stabilization.

5. Hardware implementation of harmonic control

Though harmonic control has been widely used in robotics, few hardware implementations have been proposed. Their technological choices are mostly motivated by the fact that analog resistive grids may easily compute the harmonic function as in equation (2). For example in (Stan et al., 1994) an analog implementation of a 16×16 grid is proposed. The main limitation of this work is the precision (as for most analog implementations). To our knowledge, digital FPGA-based implementations have not yet been proposed.

In this work, the discretized computation of the harmonic function is performed to make navigation decisions for a robot in an environment that may be explored on-line. Navigation commands are obtained by means of linear interpolations of the values of the neighbouring nodes (north, east, west and south), so as to obtain the best navigation orientation in $[0, 2\pi[$. It results in global trajectories towards the goal position from any starting point.

5.1 General architecture

The fine grain pipelined internal structure of the proposed digital architecture was implemented using fixed point arithmetic. Since node values range from 0 to 1, an unsigned representation with $p+1$ bits is chosen, with a p -bit fractional part. As mentioned above, large wordlengths may easily be handled.

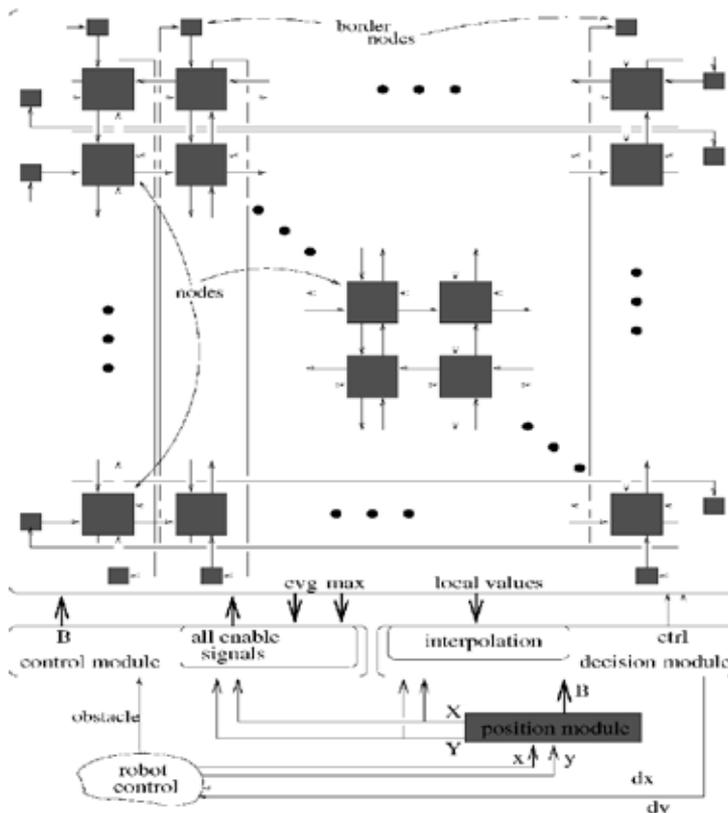


Fig. 7. General architecture

Figure 7 illustrates the general architecture of the implementation of harmonic control in a given environment. Since the environment is split into several blocks, this architecture mostly consists of a grid of $n \times m$ identical node modules (gathered 18 by 18 to handle on-chip data storage and access) surrounded by border node modules, a control module, a decision module, and a module to interact with the robot.

The role of each component is the following:

- Each node module computes its corresponding node value within the currently handled block. All node modules use interleaved loops that together perform all required computations within simultaneous local recurrent pipeline schemes. This kind of parallelism is particularly efficient for serial implementations of recurrent iterated computations within massively distributed models (Girau & Torres-Huitzil, 2007). The control of these computations are synchronized in the whole block so that node modules may serially communicate their values to their neighbours. In order to simplify the block-diagram of figure 7, only few buses and wires are shown: the signals that carry the node values from and to any neighbouring node (and possibly to opposite border nodes).
- The node modules are split in groups of 3×6 nodes that share common storage resources: a single dual port SRAM block stores the values of the 18 nodes, and its R/W accesses are controlled by a single set of counters (see 5.2).
- The border nodes are simpler than the node modules. They only store the values of the immediate neighbours of the most outer nodes within each block, and they serially generate these values when required. The only difficulty is to handle the addressing scheme so that the values stored within each of the 4 possible borders are updated when the block that contains them is being computed. This update requires the long-range connections from the node modules on each side of the block to the opposite border nodes. Moreover, when the borders lie outside the whole set of blocks, the border nodes simply generate the constant value 0 that denotes obstacles.
- The interaction with the robot has not yet been implemented. It strongly depends on the exact configuration of the robot and of the FPGA board. It includes a position modules, which role is mainly to compute the coordinates (B, X, Y) of the closest grid point (block, node) around the real coordinates (x, y) of the robot in its environment.
- The control module generates the enable signals that are sent to all node modules to control their individual behaviour when an asynchronous event occurs:
 - convergence of the computation of the harmonic function (when all blocks have confirmed that no local extremum has been detected),
 - early detection of the convergence of the computation within the currently handled block (it depends on the local convergence signals computed by all nodes within the block, see 5.2),
 - detection of an unknown obstacle (at node (B, X, Y)).
- The decision modules forwards the current position coordinates (B, X, Y) of the robot to the nodes. It collects the navigation information that are provided by the node modules in return (local values of all nodes in block B). Then it performs the

linear interpolation (see 5.3) to compute the navigation direction that must be given to the robot.

In the following subsections, the hardware architecture for the node module and its main components will be described in some detail.

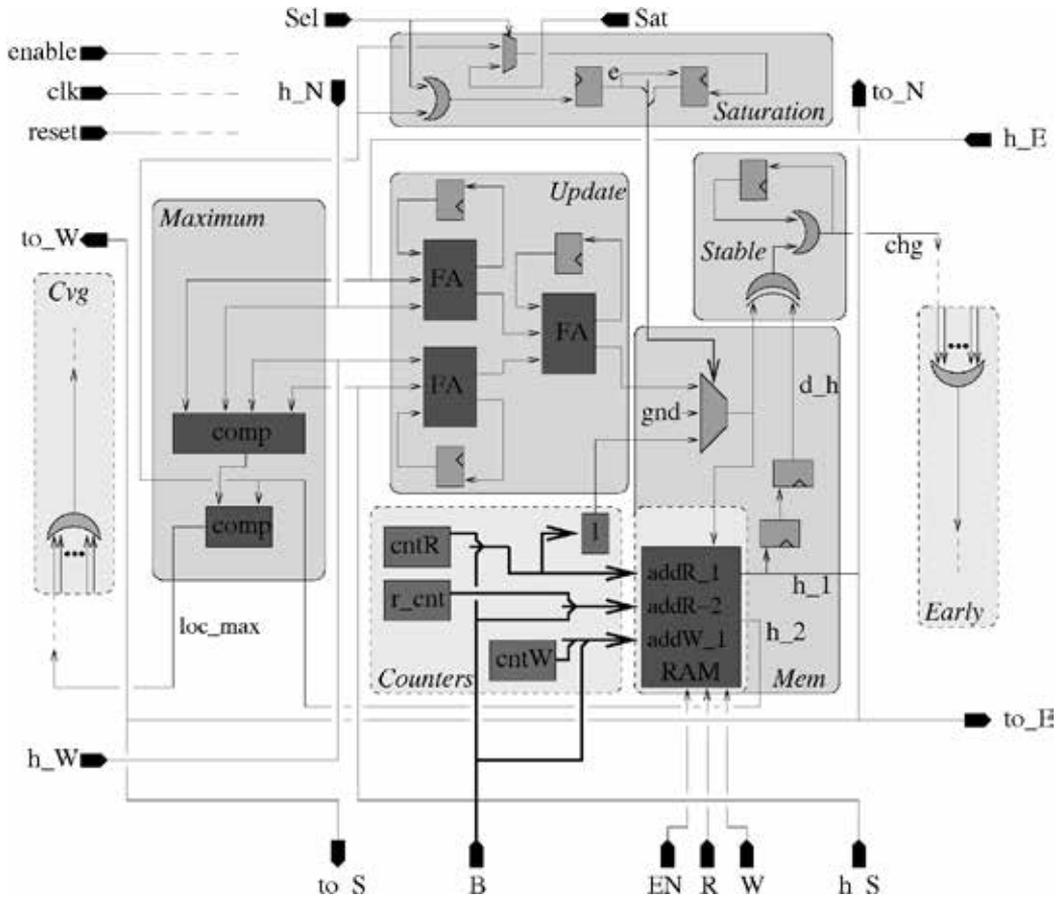


Fig. 8. Architecture of a node module

5.2 Node module implementation

The architecture for a node module is shown in the simplified block diagram on figure 8. It uses 1-bit inputs and outputs to exchange data among nodes and with the global modules. Inputs are mainly used to receive the neighbouring node values (signals h_N , h_E , h_W , h_S) and global control signals (standard signals **clk**, **reset**, **enable**, signals **Sel** and **Sat** to indicate obstacle/goal changes, and SRAM controls **EN**, **R**, **W**). The local value h of the harmonic function is sent to all 4 neighbours (signals **to_N**, **to_E**, **to_W**, **to_S**) as well as to the global interpolation module so as to compute the navigation orientation if the robot is found to be located in the area that corresponds to the local node (see 5.3 for the description of the interpolation process).

The proposed hardware node module is constituted by five main sub-modules: the iterative computation of the harmonic function is performed by the `Update` module, the local convergence of this computation (stabilization) is detected by the `Stable` module, the node receives orders to behave as an obstacle or a goal through the `Saturation` module, the `Maximum` module checks for the presence of a local maximum, and communication with the dual port SRAM block that stores the node value is controlled by the `Mem` module. Figure 8 shows this architecture, as well as its interaction with shared resources that are surrounded by dotted lines (the local `Counters` and `RAM` modules are shared by a group/cluster of 3×6 node modules, the `Early` module that detects the early stabilization of the block is part of the global control module, and the navigation decision is performed when all blocks have indicated no more local maximum through the `Cvg` module). The functionality of the main modules and their implementation details are described below.

Update: This module performs the iterated computation of the harmonic function value $h_{(i,j)}$ where (i,j) are the coordinates of the node in the environment. As described in (2), each iteration computes:

$$h_{(i,j)}(t+1) = \frac{h_{(i-1,j)}(t) + h_{(i+1,j)}(t) + h_{(i,j-1)}(t) + h_{(i,j+1)}(t)}{4}$$

Three standard Full-Adder cells compute this average, without any shift or division operator, since the output value is sent to the RAM with a write address that is delayed by 2 clock cycles (division by 4). Additional flip-flops are required to store the carry values. Since there is no more carry equal to 1 when writing the last bit of the result in the RAM, no additional reset clock cycle is required.

Stable: This module serially compares the output of the iterated computation to the stored value (delayed by two flip-flops in the `Mem` module). This local convergence test is then sent to a global OR gate (in the `Early` module) to disable the inner computation loop of the block when early stabilization has been detected before I iterations.

Maximum: This module uses a comparison between all neighbouring values and a comparison with the local value so as to determine whether the local node corresponds to a local maximum. This local information is sent to the `Cvg` module that uses a global OR gate so as to check for the presence of any local maximum in the current block. This global information is handled by the general `Control` module to finish the whole computation of harmonic values, so that the determination of the trajectory may start.

Counters: This module is shared by 18 node modules. It generates the read (resp. write) addresses for the dual port RAM by means of counters `cntR`, `r_cnt` (resp. `cntW`). Both read addresses are sent to the ports of the RAM to handle both LSBF (counter `cntR`) and MSBF (reverse counter `r_cnt`) modes. When handling d -bit precision data, these counters are reset each $d+2$ cycles (the RAM is written with a 2-clock cycles delay). Value 1 (for obstacles) is computed as a logical function of `cntR`.

Mem and Saturation: The local node value is not directly the output of the Update module. It may also be a constant 0 or 1 (goal or obstacle). A multiplexer selects the correct value with respect to a control given by the Saturation module that memorizes the Sat value to be the constant value of the grid point when the node is selected by the global control module (signal Sel). Since multiple blocks are handled, these constant values must also be stored in and retrieved from the RAM. To do that, we add a special bit (the MSB) to the values stored in memory (this bit is set to 1 when the local value is constant). The Saturation module also receives this information.

5.3 Convergence detection and determination of the navigation direction

Control: This global module performs the usual scheduling of the loops of the algorithm through various counters. It mostly computes the number B of the current block, and it takes into account the stabilization and the convergence (no maximum) of the different blocks to adapt the global scheduling. Moreover, it handles the different counters such that the computations are performed with an increasing precision until global convergence.

Decision: This module operates after convergence of the iterations. Knowing the coordinates of the node (B, X, Y) that corresponds to the current position of the robot, this module acts as a sequential program that computes the maximum slope among the four triangles that are defined by the node and two of its immediate neighbours. It simply reduces to the determination of the two consecutive neighbours (B, X', Y') and (B, X'', Y'') which values maximize the sum of their difference with respect to the center node value $(h_{(B,X,Y)} - h_{(B,X',Y')})^2 + (h_{(B,X,Y)} - h_{(B,X'',Y'')})^2$. Then the best trajectory direction (gradient ascent) is given by a vector which coordinates are equal to $(h_{(B,X,Y)} - h_{(B,X',Y')})$ and $(h_{(B,X,Y)} - h_{(B,X'',Y'')})$ or to their opposite values (it depends on the position of (X', Y') and (X'', Y'') with respect to (X, Y)).

5.4 Implementation results

In order to deal with larger environments, elemental node modules are gathered together to form a 2D grid of 3x6 clusters. Most of the connections among nodes are local with the four neighbours. The main reason to group in such a configuration is due to the 18-bit width of the shared block ram (technological constraints of the targeted FPGA) that is used to store the values of each node. The depth of the BlockRAMs is 1K. It allows handling a wide range of arithmetic precisions such as 64-128 bits per word without modifying the memory organization.

Node hardware resource utilization XC2V6000-5ff1517	
Number of Slice Flip Flops	11/67,584
Number of 4 input LUTs	22/67584
Number of occupied Slices	13/33792
Frequency	361 MHz

Table 1. Synthesis results for a node module

This work uses two PCI bus board. The first one is equipped with a Virtex XC2V6000-4FF1517 FPGA from Xilinx, with up to 6,000,000 system gates. Such a FPGA contains 67,584 logic cells. The second one is equipped with three FPGAs, the largest one being a Virtex-4 XC4VLX160ff1513-12 FPGA from Xilinx, that contains 135,168 logic cells, to be compared with the 200,448 ones of the current largest Virtex-4. The design was synthesized, placed and routed automatically in Xilinx Foundation ISE 7.1i. Results are shown in tables 1, 2 and 3. Each node module requires 26 logic cells, and each cluster of 18 node modules requires 470 logic cells (counters included). On a XC2V6000, 144 dual port SRAM blocks are available, that may be configured as 1Kx18 RAMs to be shared by clusters of 18 node modules. The whole architecture may implement a 36x66 grid on less than 92 % of the XC2V6000 logic cells.

3x6 cluster hardware resource utilization XC2V6000-5ff1517	
Number of Slice Flip Flops	198/67,584
Number of 4 input LUTs	398/67584
Number of occupied Slices	235/33792
Frequency	300 MHz

Table 2. Synthesis results for a 18-node cluster module

We use the remaining 12 SRAM blocks to implement the storage facilities of the 204 border blocks (for a 36x66 grid). Handling addresses for these special blocks mostly reduces to generate in a parallel way $B+1$, $B-1$, $B+k$, $B-k$ (all values modulo $k \times k$, where $k \times k$ is the total number of blocks). A few slices are sufficient (around 10 for up to 5x5 blocks). Then the `Control` module (844 slices) and the `Decision` module are added ($3p + 682$ slices for a p -bit precision). So that for example 99.4 % of this FPGA is finally used for the implementation of the whole algorithm with 4 blocks and $p=255$ (see below for the speed).

Larger blocks (up to 54x108 nodes) may be implemented on the current largest FPGAs with this approach (the available SRAM blocks being the critical resource). As an example, table 3 shows the hardware resource utilization for a 48x96 block on the Virtex-4.

48x96 block hardware resource utilization XC4VLX160ffff1513-12	
Number of Slice Flip Flops	50699/135168
Number of 4 input LUTs	101396/135168
Number of occupied Slices	59914/67584
Frequency	150 MHz

Table 3. Synthesis results for a 48x96 block

Software implementations of the harmonic function computation on a microprocessor based computer, Pentium 4,2 GHz, require around 100 μ s per iteration with a 36x66 block. In the proposed hardware implementation, $p+2$ clock cycles are required per iteration for precision p , with an estimated clock frequency of 150 MHz. Thus, the implementation on the Virtex-2 provides a speed factor up to 100x (for a 128-bit precision that corresponds to some average-sized environments in our reported experiments), that would even increase with the number of nodes in the grid (sequential vs parallel implementation). But the

implementation speed is not the only advantage of our implementation. Power consumption is a key factor for embedded implementations, and above all large precisions may be handled by the proposed serial implementation when few blocks are used (up to 1K bits when only one block is used).

When using multiple blocks (for large environments) together with an increasing precision, the computation time linearly increases with the number of blocks as in the sequential implementation on PC. Therefore, the speedup is not intrinsically changed. But following our experiments in section 4, the number of iterations decreases before convergence, so that a final speed factor is up to 400x. As an example, with a not too complex maze with 9500 nodes divided into 4 blocks, a $p=255$ precision, and $l=6$ consecutive iterations for each block at most, a speedup of 270x is obtained.

It might be noticed that the depth of the SRAM memory blocks appears as the main limitation to handle larger environments (more blocks). Considering that the most recent Virtex-5 FPGAs contain different SRAM blocks, some of them twice larger than the ones we use, this limitation should also evolve with the technological improvements of FPGAs.

6. Conclusion

This chapter presents an embedded architecture to solve the navigation problem in robotics, that computes trajectories along a harmonic potential, using a FPGA implementation. This architecture includes the iterated estimation of the harmonic functions. The goals and obstacles of the navigation problem may be changed during computation. The trajectory decision is also performed on-chip, by means of local computations of the preferred direction at each point of the discretized environment. The proposed architecture uses a massively distributed grid of identical nodes that interact with each other within mutually dependant serial streams of data to perform pipelined iterative updates of the local harmonic function values until global convergence.

When the environment size is too large for a fully parallel implementation on the used FPGA, our implementation takes advantage of the available SRAM to handle larger environments that are partitioned into blocks. It results in an iterated computation mode that is both globally asynchronous and block-synchronous.

The proposed architecture also introduces the use of an increasing precision. First of all, this approach enables our implementation to reach the required precision for convergence without having to over-estimate it initially (resulting in excessively long computations). Then it also enables an optimization of the overall computation time. This optimization is carefully studied from a theoretical and experimental point of view, with respect to both the block-synchronous approach and the increasing precision technique.

Despite all these results, our implementation may still appear as not able to handle particularly large and complex environments. This is intrinsically linked to the nature of the harmonic control that rapidly requires huge precisions for such environments. The main perspective of this work is to extend it to optimal control, a more generic (and tunable)

trajectory planning method, that uses similar computations without requiring such huge precisions.

References

- Alvarez, D.; Alvarez, J.C. and Gonzalez, R.C. (2003). Online motion planning using Laplace potential fields, *Proceedings of the Int. Conf. Robotics and Automation*, IEEE CNF.
- Boumaza, A. and Louchet, J. (2003). Mobile robot sensor fusion using flies, In: *Applications of Evolutionary Computing*, volume 2611 of LNCS, pages 357–367.
- Connolly, C.I.; Burns, J.B. and Weiss, R. (1990). Path planning using laplace's equation. *Proceedings of the Int. Conf. on Robotics and Automation*, pages 2102–2106, IEEE CNF.
- Connolly, J.C. and Grupen, R. (1993). On the applications of harmonic functions to robotics. *Journal of Robotic and Systems*, 10(7):931–946.
- Feder, H.J.S. and Slotine, J.J.E. (1997). Real-time path planning using harmonic potentials in dynamic environments, *Proc. of the Int. Conf. Robotics and Automation*, IEEE CNF.
- Girau, B. and Boumaza, A. (2007). Embedded harmonic control for dynamic trajectory planning on FPGA, *Proceedings of Int. Conf. on Artificial Intelligence and Applications*.
- Girau, B. and Torres-Huitzil, C. (2007). Massively distributed digital implementation of an integrate-and-fire LEGION network for visual scene segmentation. *Neurocomputing*, 70.
- Huber, M.; MacDonald, W.S. and Grupen, R.A. (1996). A control basis for multilegged walking. *Proceedings of the IEEE Int. Conf. Robotics and Automation*, IEEE CNF.
- Kazemi, M.; Mehrandezh, M. and Gupta, K. (2005). An incremental harmonic function-based probabilistic roadmap approach to robot path planning, *Proceedings of the IEEE Int. Conf. Robotics and Automation*, IEEE CNF.
- Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotic Research*, 5(1):90–98.
- Masoud, A.A. and Masoud, S.A. (2002). Motion planning in the presence of directional and obstacle avoidance constraints using nonlinear anisotropic, harmonic potentialfields: A physical metaphor. *IEEE Transactions on Systems, Man, & Cybernetics, Part A: systems and humans*, 32(6):705–723.
- Stan, M.; Burleson, W.; Connolly, C. and Grupen, R. (1994). Analog vlsi for robot path planning. *Journal of VLSI Signal Processing*, 8(1):61–73.
- Sweeney, J.D.; Li, H.; Grupen, R.A. and Ramamritham, K. (2003). Scalability and schedulability in large, coordinated, distributed robot systems, *Proceedings of the IEEE Int. Conf. Robotics and Automation*, IEEE CNF.
- Tarassenko, L. and Blake, A. (1991). Analogue computation of collision-free paths, *Proceedings of the Int. Conf. on Robotics and Automation*, pages 540–545, IEEE CNF.
- Wang, Y. and Chirikjian, G.S. (2000). A new potential field method for robot path planning, *Proceedings of the Int. Conf. Robotics and Automation*, vol. 2, pages 977–982, IEEE CNF.
- Xilinx, editor (2000). *The Programmable Logic Data Book*. Xilinx, Inc.
- Young, D. (1971). *Iterative solutions of large linear system*, Academic Press, New York.
- Zelek, J.S. (1998). Complete real-time path planning during sensor-based discovery, *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and systems*.

Velocity Observer for Mechanical Systems

Ricardo Guerra¹, Claudiu Iurian², Leonardo Acho²

¹*Universidad Autónoma de Baja California*, ²*Universitat Politècnica de Catalunya*
¹*México*, ²*España*

1. Introduction

Many controllers incorporate knowledge of both position and velocity, such as PD, PID and most robust controllers. In many applications direct access to actual velocity is not available. In certain scenarios the velocity signal may contain an excessive amount of noise and is not well suited for use in a control law; in the particular case of measuring robot joint velocities, direct measurement may even be undesirable (Arteaga and Kelly, 2004). Consequently it is necessary to estimate the velocity signal using an observer, and feed it into the controller.

Velocity observer design is a very important topic that continues to be studied, as in (Arteaga and Kelly, 2004; Berghuis and Nijmeijer, 1993; Canudas de Wit and Fixot, 1992). Discontinuous state observers for inexact nonlinear plants have been designed in (Choi et al., 1999; Xiong and Saif, 2001; Xian et al., 2004). However, little research has focused on the specific case of velocity observation for mechanical systems with friction working at low velocities. In these systems friction has been shown to cause mechanical difficulties which are usually unwanted phenomena (Armstrong-Hélouvry et al., 1994). Under such conditions, the use of the stated observers leads to high frequency oscillations in the estimated velocity signal, this can lead to accelerated degradation of system performance, which is why the observer in (Xian et al., 2004) is used as a basis for developing two new observers in (Guerra et al., 2007b) for the specific purpose of being used in mechanical systems with friction working at low velocities. Our objective is to mitigate the high frequency oscillations and increase the reliability of velocity observers.

The observers developed with this approach retain the stability qualities of their predecessor yet do not exhibit the oscillatory behaviour, as will be seen later. The observers presented in (Xian et al., 2004; Guerra et al., 2007b) are numerically compared before proceeding to an experimental verification. Afterwards, one of the observers from (Guerra et al., 2007b) is used as part of a control scheme for mechanical systems which includes a PD controller and an adaptive friction compensator that depends on knowledge of velocity.

2. Observer Design

2.1 Previous Work

Consider the class of mechanical systems described by (Xian et al., 2004):

$$\ddot{x} = h(x, \dot{x}) + G(x, \dot{x})u \quad (1)$$

where $x \in \mathfrak{R}$ is the system output, $u(t) \in \mathfrak{R}$ is the control input, and $h(x, \dot{x}) \in \mathfrak{R}$ as well as $G(x, \dot{x}) \in \mathfrak{R}$ are nonlinear functions¹. The system (1) satisfies the following assumptions (Xian et al., 2004):

Assumption A1. Both $h(x, \dot{x}) \in \mathfrak{R}$ and $G(x, \dot{x}) \in \mathfrak{R}$ are C^1 functions.

Assumption A2. The control input is a C^1 function and $u(t), \dot{u}(t) \in L_\infty$.

Assumption A3. The system state is bounded for all time; i.e., $x(t), \dot{x}(t) \in L_\infty$.

The velocity observer aims to estimate the inaccessible velocity signal $\dot{x}(t)$ using only the position $x(t)$ and assuming that $h(x, \dot{x})$, $G(x, \dot{x})$ and $u(t)$ are unknown (Xian et al., 2004). The objective is then to ensure that the estimation error tends to zero as time tends to infinity. Consider the velocity observer presented in (Xian et al., 2004):

$$\begin{aligned} \dot{\hat{x}} &= p + k_0 \tilde{x} \\ \dot{p} &= k_1 \operatorname{sgn}(\tilde{x}) + k_2 \tilde{x} \end{aligned} \quad (2)$$

where k_0 , k_1 , and k_2 are constant observer design parameters and $\operatorname{sgn}(\cdot)$ is the signum function, let:

$$N_0(x, \dot{x}, t) = h(x, \dot{x}) + G(x, \dot{x})u(t) \quad (3)$$

Theorem 1(Xian et al., 2004). The observer (2) ensures that the velocity estimation error $\tilde{x}(t)$ tends to zero as time tends to infinity provided that k_1 satisfies:

$$k_1 > \|N_0(x, \dot{x}, t)\|_\infty + \|\dot{N}_0(x, \dot{x}, t)\|_\infty \quad (4)$$

For detailed proof see Theorem 2 in (Xian et al., 2004).

2.2 Proposed observers

Consider now the following observer (Guerra et al., 2007b):

$$\begin{aligned} \dot{\hat{x}} &= p + k_0 \tilde{x} \\ \dot{p} &= -k_1 \operatorname{sgn}(\tilde{x}) + k_2 \tilde{x} \end{aligned} \quad (5)$$

where k_0 , k_1 , and k_2 are constant positive observer design parameters.

Theorem 2(Guerra et al., 2007b). The observer (5) ensures that the velocity estimation error $\tilde{x}(t)$ tends to zero as time tends to infinity provided that k_1 satisfies the same restriction as in Theorem 1. The proof is identical.

A third option for estimating velocity is given by (Guerra et al., 2007b):

¹ Without loss of generality, we have assumed a one Degree-of-Freedom mechanical system.

$$\begin{aligned}\dot{\hat{x}} &= p + k_0 \tilde{x} \\ \dot{p} &= k_1 \operatorname{sgn}(\hat{x}) + k_2 \tilde{x}\end{aligned}\quad (6)$$

Theorem 3 (Guerra et al., 2007b). The observer (6) also ensures that the velocity estimation error $\tilde{x}(t)$ tends to zero as time tends to infinity provided that k_1 satisfies the same restriction as in Theorem 1. The proof is the same as in Theorem 2.

The modifications implemented in these observers are: observer (5) proposes an inversion of the sign in the second term of the estimation dynamic which produces a filtering effect. Observer (6) introduces a change in the argument of the sign function to reduce the high frequency content present in observer (2).

3. Numerical Experiments

Consider a linear motion of unit mass:

$$\ddot{x} = u - f \quad (7)$$

where f is the friction force and u is the control force acting on the mass. Assuming that there is no friction in the system (i.e. $f = 0$) and that $k_1 = 10$. The PID controller:

$$u = -k_p(x - x_d) - k_i \int (x - x_d) dt - k_d \dot{x} \quad (8)$$

makes the closed loop system asymptotically stable with $k_d = 6$, $k_p = 3$, $k_i = 4$ and the constant reference set at $x_d = 1\text{m}$, for full details consult (Canudas de Wit et al., 1995). Since friction is to be expected in mechanical systems, we include it in our simulations using the LuGre model with the parameters given in (Canudas de Wit et al., 1995), thus the friction force f is obtained as a non linear dynamic. The observer design is completed by setting $k_0 = k_2 = 10$. Figure 1 depicts the position and velocity of the system considering that the velocity is available for use in the PID controller, as shown in (Canudas de Wit et al., 1995).

We repeat the experiment in order to test the observers. At this point the actual velocity, not the observers, is used in the control law. The results obtained are shown in Figure 2, where it can clearly be seen that the observer (2) generates a small amplitude chattering (high frequency oscillation). In mechanical systems such signals are undesirable because they can cause damage and accelerate wear, as well as activate un-modelled dynamics. Since the premise of the observers is to be used in systems where velocity is not available, the previous experiment was repeated using instead of the actual velocity, the observed velocity \hat{x} employing Theorems 1, 2 and 3, the results are shown in Figures 3, 4 and 5, where the slight differences in reached position show that the observer influences system performance. Figure 6 shows the results of modifying the observer gains to $k_1 = 5$ and $k_0 = k_2 = 1$; Figures 7 and 8 show the results of using the estimated velocity from observers (5) and (6) respectively, in the PID controller (observer (2) becomes unstable).

4. Application to an Industrial Emulator

We proceed to evaluate the observers previously discussed on an experimental testbed.

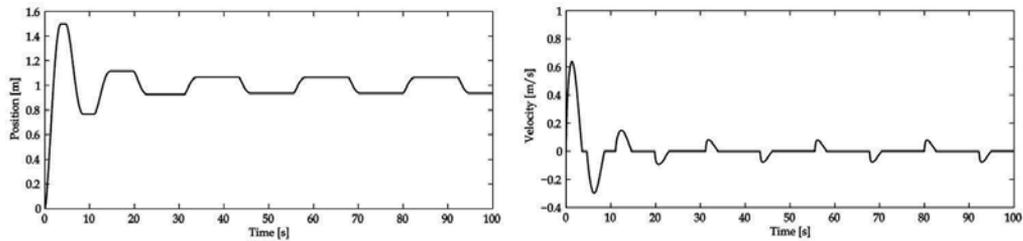


Fig. 1. Positioning experiment presented in (Canudas de Wit et al., 1995).

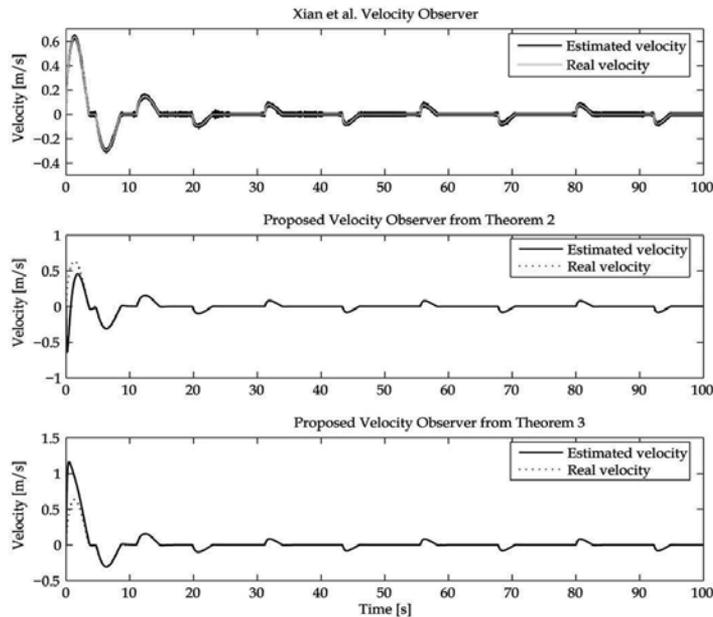


Fig. 2. Simulated performance of velocity estimators for observation only.

4.1 Experimental Platform

The experimental evaluation was carried out on an ECP Model 220 industrial emulator which includes a PC-Based control platform and a DC brushless servo system (ECP, 1995). The system includes two motors, one as a servo actuator and one as a disturbance input (not used here), a power amplifier, and two encoders which provide accurate position measurements, i.e., 4000 lines per revolution with $4\times$ hardware interpolation yielding 16000 counts per revolution to each encoder, 1 count (equivalent to 0.000392 radians or 0.0225 degrees) is the lowest measurable angular displacement (ECP, 1995). The system was set up to incorporate inertia and friction. The friction coefficients for the system were found to be

$F_v = 0.05772$ [Nms/rad] (viscous friction coefficient) and $F_c = 0.43043$ [Nm] (Coulomb friction level) using the procedure presented in (Kelly and Campa, 2000). The drive and load disks were connected through a 4 : 1 speed reduction (Figure 9).

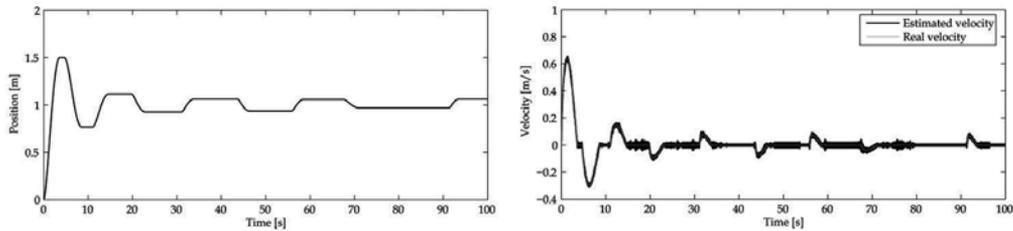


Fig. 3. Simulated performance of observer (2) when used in the control law.

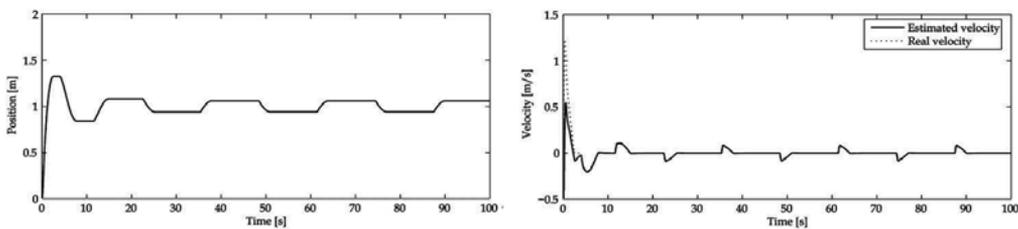


Fig. 4. Simulated performance of observer (5) when used in the control law.

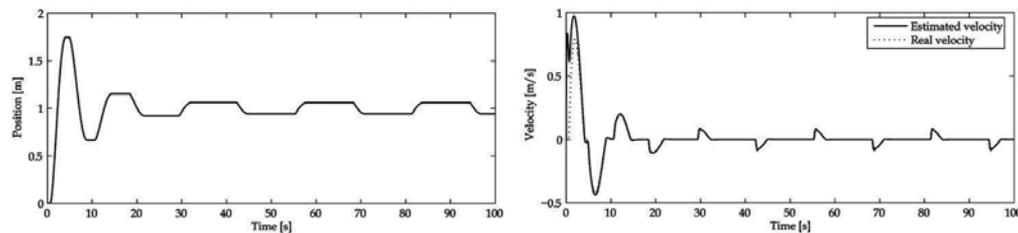


Fig. 5. Simulated performance of observer (6) when used in the control law.

To implement the control algorithms, a Pentium 4, 2.80 GHz CPU, with 512 MB RAM computer running windows XP is programmed using the interface medium ECP USR Executive 5.1, a programming language similar to C (ECP, 1995). The system contains a data acquisition board for digital to analog conversion and a counter board to read the position encoder outputs from the servo system. The minimum servo-loop sampling time is $T_s = 0.884$ ms.

The output voltage signal generated by the system is in the range of ± 5 V and is delivered to the motor drive through the DAC, the measurement feedback is a position signal (in counts or radians) measured at the shaft of the two disks by the optical rotary incremental position encoders, which is then read by the microcomputer by means of the counter board and delivered into the PC. A software interface has been built to easily transfer the data collected from the plant (using the ECP USR Executive program) to the Matlab workspace environment, in order to display the results. Four weights of 0.5 Kg each were placed on the

load disk at a radius of 10 cm, while the drive disk remained unweighted. It is worth mentioning that the mechanical system has encoders that provide accurate position

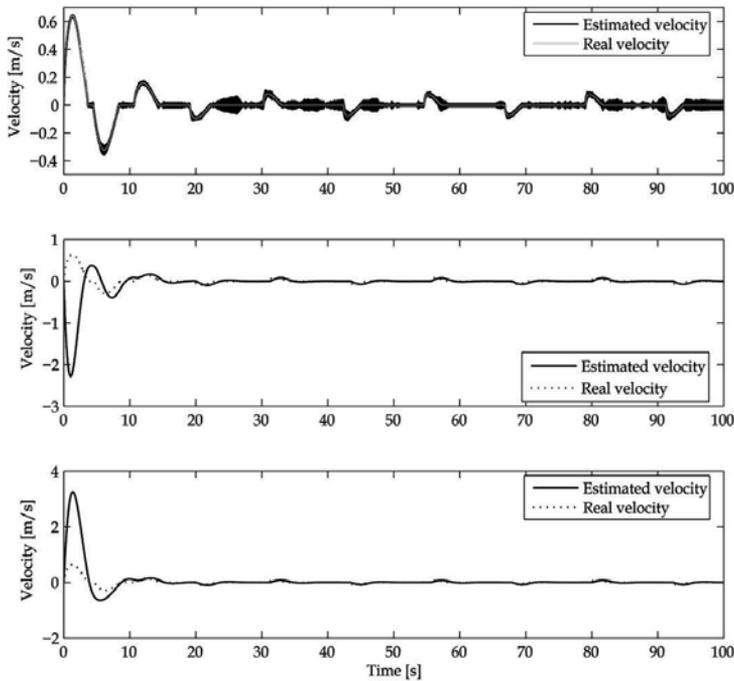


Fig. 6. Simulated performance of estimators (for observation only) using modified observer gains. Top: observer (2). Middle: observer (5). Bottom: observer (6).

measurements but not velocity sensors, i.e., there is no direct access to velocity (ECP, 1995). Under these circumstances we proceed to implement the aforementioned velocity observers, the results obtained are shown in Figures 10 through 12.

4.2 Experimental Results

The control law implemented in all three cases is (Guerra et al., 2007b):

$$u = -k_p(x - x_d) - k_i \int (x - x_d) dt - k_d \dot{x} \tag{9}$$

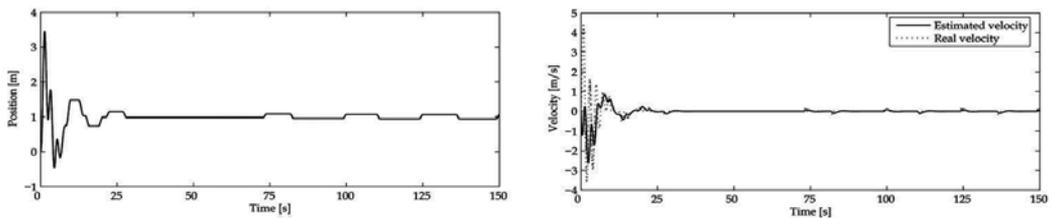


Fig. 7. Simulated performance of observer (5) with modified gains when used in the control law.

with $k_d = 0.0011$, $k_p = 0.135$, $k_i = 0.4$. The desired position for the load disk was set to $x_d = 100$ [counts] = 0.0392 [radians] = 2.25 [degrees]. It can be seen in Figure 10 that observer

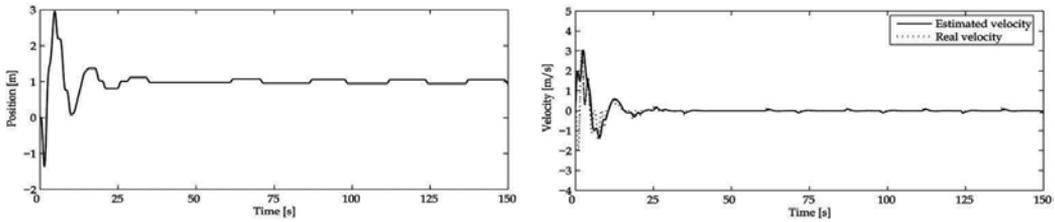


Fig. 8. Simulated performance of observer (6) with modified gains when used in the control law.

(2) produces oscillations of relatively large amplitude high frequency; as previously stated, this is undesirable in mechanical systems. The observer from Theorem 2 significantly reduces these unwanted behaviours after a transient, as seen in Figure 11. The observer (6) further reduces both the amplitude and the duration of the transient and it eliminates the chattering effect, as seen in Figure 12. It should be noted that the rectangular shaped limit cycles clearly visible in Figures 11 and 12 follow the behaviour presented in (Canudas de Wit et al., 1995) whereas in Figure 10 they are indistinguishable (Guerra et al., 2007b).

5. Inclusion in Friction Compensation Strategies

Having seen that observer (6) exhibits better performance in the mechanical system, it was decided to include it in friction compensators that rely on knowledge of the unavailable velocity. Four compensators were selected: two for positioning applications and two for trajectory tracking applications. The purpose of this experiment is to compare the performance of the friction compensators in a real system, this would not have been possible without an adequate velocity observer, as both the PD control law and the friction compensator depend on knowledge of the velocity. The stability analysis and full details for each compensator can be found in the mentioned references.

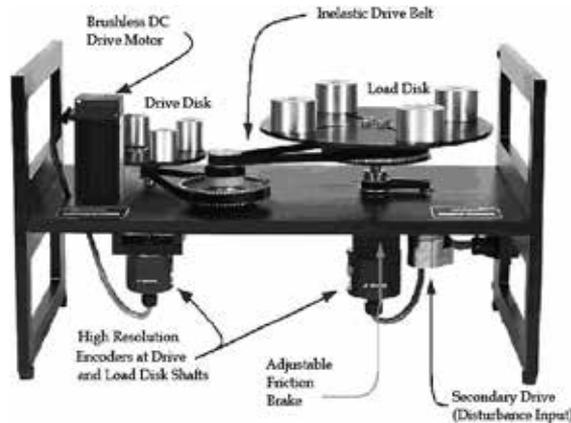


Fig. 9. Mechanical System.

5.1 Positioning Applications

The two positioning application friction compensators considered were presented in (Friedland & Park, 1992) and (Guerra et al., 2007a). These compensators are very similar and both have very good performance in simulated experiments. In our system they achieved the results shown in Figure 13.

5.2 Trajectory Tracking Applications

The trajectory tracking application friction compensators considered were presented in (Liao & Chien, 2000) and (Guerra & Acho, 2007). Again the structure of the compensators is very similar as are the simulated results that show very good performance. The experimental results obtained are shown in Figure 14.

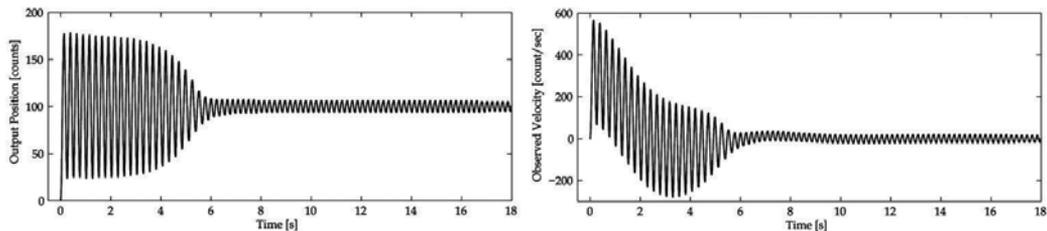


Fig. 10. Experimental results with the observer from Theorem 1.

6. Conclusions

Two velocity observers are presented in this chapter that are based on the one presented in (Xian et al., 2004) but have practical advantages. It has been shown with both numerical and experimental results that the proposed observers can accurately estimate velocity and avoid chattering that is undesirable in mechanical systems, when there is only access to position measurements. The observers (5) and (6) are especially interesting for industrial applications, since it has been shown that velocity sensing hardware can be replaced with reliable inexpensive software without difficulty.

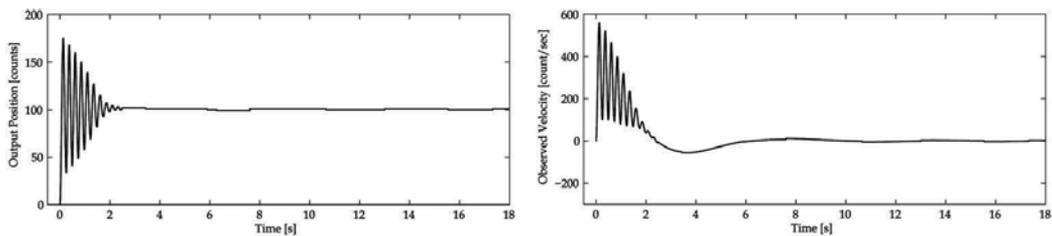


Fig. 11. Experimental results with the observer from Theorem 2.

7. Acknowledgements

The work of Dr. Claudiu Iurian and Dr. Leonardo Acho was supported by CICYT through Grant DPI2005-08668-C03-01. The work of Dr. Ricardo Guerra was supported by CONACYT

by means of a Doctoral Scholarship.

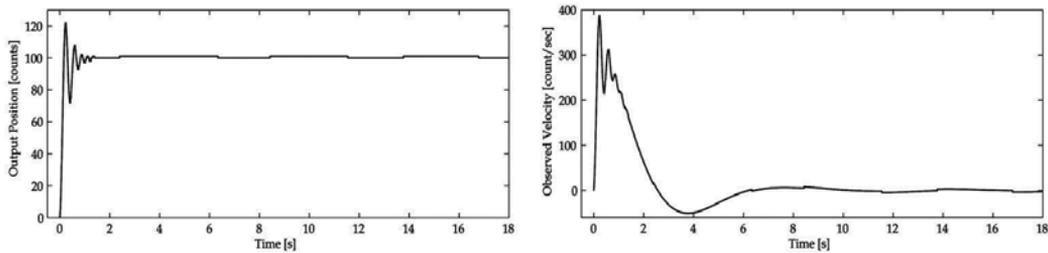


Fig. 12. Experimental results with the observer from Theorem 3.

8. References

- Armstrong-Hélouvry, B; Dupont, P.; & Canudas de Wit, C. (1994). A Survey of Models, Analysis Tools and Compensation Methods for the Control of Machines with Friction. *Automatica*, 30, 7, (July 1994) 1083–1138, ISSN:0005-1098
- Artega, M. A. and Kelly, R. (2004). Robot Control without Velocity Measurements: New Theory and Experimental Results. *IEEE Transactions on Robotics and Automation*, 20, 2, (April 2004) 297–308, ISSN: 1042-296X
- Berghuis, H. & Nijmeijer, H. (1993). Global Regulation of Robots Using only Position Measurements. *Systems and Control Letters*, 21, 4, (October 1993) 289–283, ISSN: 0167-6911
- Canudas de Wit, C. & Fixot, N. (1992). Adaptive Control of Robot Manipulators via Velocity Estimatedfeedback. *IEEE Transactions on Automatic Control*, 37, 8, (August 1992) 1234–1237, ISSN: 0018-9286
- Canudas de Wit, C.; Olsson, H.; Åström, K. J. & Lichinsky, P. (1995). A New Model for Control of Systems with Friction. *IEEE Transactions on Automatic Control*, 40, 3, (March 1995) 419–425, ISSN: 0018-9286
- Choi, J.; Misawa, E. and Young, G. (1999). A Study on Sliding Mode State Estimation. *Journal of Dynamic Systems, Measurement and Control*, 121, 2, (June 1999) 255–260, ISSN: 0022-0434
- ECP (1995). *Manual for model 220 industrial emulator/servo trainer*, Educational Control Products, ISBN:, California, USA.
- Friedland, B. & Park Y. J. (1992). On Adaptive Friction Compensation. *IEEE Transactions on Automatic Control*, 37, 10 (October, 1992) 1609–1612, ISSN: 0018-9286
- Guerra, R.; Acho, L. & Aguilar, L. (2007a) Adaptive Friction Compensation for Mechanisms: A New Perspective. *International Journal of Robotics and Automation*, 22, 2, (July 2007) 155–159, ISSN: 0826-8185
- Guerra, R. & Acho, L. (2007) Adaptive Friction Compensation for Tracking Control of Mechanisms. *Asian Journal of Control*, 9, 4 (December 2007) 422–425, ISSN: 1561-8625
- Guerra, R.; Iurian, C.; Acho, L; Ikhouane, F. & Rodellar, J. (2007b) Global Asymptotic Velocity Observation of Nonlinear Systems: Application to a Frictional Industrial Emulator. *Proceedings of the fourth International Conference on Informatics in Control, Automation and Robotics ICINCO 2007*, 85–91 ISBN: 978-972-8865-82-5, Angers, France, May 2007, INSTICC PRESS

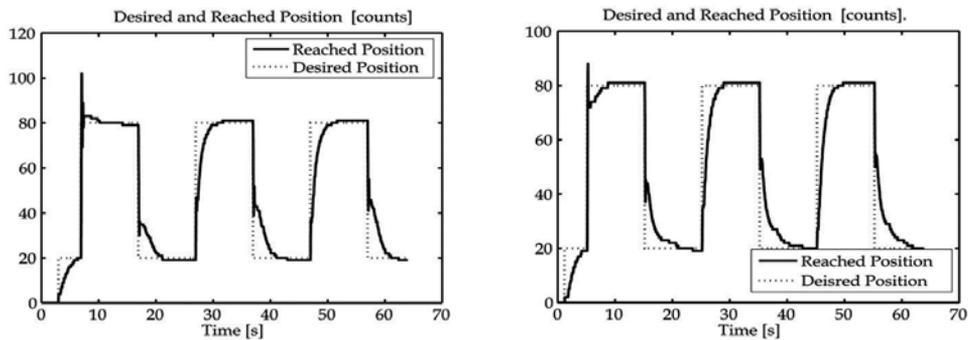


Fig. 13. Experimental results using observer (6) in a friction compensation scheme for positioning applications. Left: using the compensator from (Friedland & Park, 1992). Right: using the compensator from (Guerra et al., 2007a).

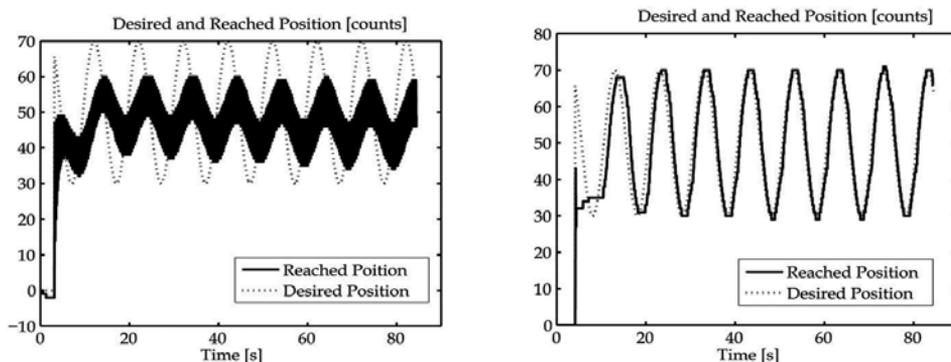


Fig. 14. Experimental results using observer (6) in a friction compensation scheme for trajectory tracking applications. Left: using the compensator from (Liao & Chien, 2000). Right: using the compensator from (Guerra & Acho, 2007).

- Hung, J. C. (1993). Chattering Handling for Variable Structure Control Systems. *Proceedings of IECON '93, 1968-1972*, ISBN: , Maui, Hawaii, USA. 1993
- Kelly, R. & Campa, R. (2000). A Measurement Procedure for Viscous and Coulomb Friction. *IEEE Transactions on Instrumentation and Measurements*, 49, 4, 857-867, ISSN: 0018-9456
- Liao, T. & Chien, T. (2000). An Exponentially Stable Adaptive Friction Compensator. *IEEE Transactions on Automatic Control*, 37, 10 (October, 1992) 1609-1612, ISSN: 0018-9286
- Xian, C.; de Queiroz, M. S.; Dawson, D. M. & McIntyre, M. L. (2004). A Discontinuous Output Controller and Velocity Observer for Nonlinear Mechanical Systems. *Automatica*, 40, 4, (April 2004) 695-700, ISSN: 0005-1098
- Xiong, Y. & Saif, M. (2001). Sliding Mode Observer for Nonlinear Uncertain Systems. *IEEE Transactions on Automatic Control*, 46, 12, (December 2001) 2012-2017, ISSN: 0018-9286

Evolution of Neuro-Controllers for Trajectory Planning Applied to a Bipedal Walking Robot with a Tail

Álvaro Gutiérrez¹, Fernando J. Berenguer² and Félix Monasterio-Huelin¹

¹*Universidad Politécnica de Madrid*

²*Fundación PRODINTEC*

Spain

1. Introduction

Bipeds are complex hybrid dynamical systems, in the sense that they mix both continuous and discrete-event phenomena (Hurmuzlu et al., 2004). The main characteristic of biped walkers is the abrupt kinematics change between the aerial phase and the support phase accompanied of dynamical impacts. The main problem is how to achieve a rhythmical or periodical walk. One of the main problems of these robots is their high power and energy consumption, which limits mainly their autonomy. It could be attributed to, for example, the high number of actuated joints (about 20), and also because the study of energy consumption is not often considered during the planning of movements.

The construction of a locomotion controller for completely actuated legged robots is performed in many forms. Firstly by obtaining predetermined trajectories, using techniques based on kinematical models or Artificial Intelligence methods like the evolution of locomotion based on fuzzy controllers (Magdalena & Monasterio, 1995) or spline controllers (Boeing & Bräunl, 2004) for example. These studies are oriented to solve trajectory generation problems for the active control centred approach. In (Löffler et al., 2004) there is an example of this two-part methodology showing that an impedance control is better than the computed torque method. Despite this better performance, the second algorithm needs both a position sensor and a velocity sensor, and the impedance controller needs also force-torque sensors.

The conventional approach has been questioned by researchers inspired by biomechanical models (McMahon, 1985 and Kuo, 2007). The discovery by McGeer (McGeer, 1990) of passive dynamic walking by building a biped without any motors or sensors opens the doors to a new design concept based on morphological considerations. The interaction or trade-off between morphology and control is in the centre of the more recent research and debates in robotics (Matsushita et al., 2006 and Pfeifer & Bongard, 2007). Exploiting the intrinsic passive dynamics has many advantages against the classical two-part methodology (trajectory generation plus active controller). Two of them are the energy consumption reduction (the only energetic cost occurs in step-to-step transitions) and the control simplicity (low computational cost).

Nevertheless, some theoretical and practical studies (Collins & Ruina, 2005) show that it is difficult to achieve the complex dynamic exhibited by the human and animal locomotion taking into account only the properties of the simple passive-dynamic walking, but might help to design walker robots. How to exploit the above-mentioned passive properties of biped robots with the incorporation of sensors is studied in (Iida & Pfeifer, 2006). Other studies make use of passive trajectories to design active controls under the classical methodology. Some researches are based on kinematics (Asano et al., 2004) and other on energy constraints (Asano et al., 2005).

Conventional models of bipedal walking generally assume rigid body structures, while elastic material properties seem to play an essential role in nature (Alexander, 2008). Spring-damper elements have been proposed to advance one solution under the compliant leg concept (Iida et al., 2008, Geyer & Seyfarth, 2006 and Iida et al., 2007) or through an adaptable compliance (inverse of stiffness) mechanism (Van Ham et al. 2007).

Passive biped robots have stable limit cycles when they go down a small slope (3° - 5°). In (Siqueira & Terra, 2006) a torso was added to a passive biped robot. Due to this fact a stable limit cycle cannot be found. Then the authors add two PD controllers to keep the torso in a fixed position to achieve the walk. In (Berenguer & Monasterio-Huelin, 2006, 2007a, 2007b, 2007c and 2008) an actuated tail is added to produce the walk. In both cases a spring is added to the ankle to transform the stored potential energy into kinematics one.

In this work we use the last model to study the problem of going up a small slope (0° - 10°). To achieve this objective it is necessary to solve a trajectory planning and a control problem. As has been said, classical methods solve both problems separately adding to the robot one actuator (motor) for each degree of freedom. This technical solution could be avoided taking into account a different philosophy. Instead of generating an explicit desired path defined as a function of kinematics variables (positions and velocities) or even dynamic variables (forces and torques) we can design the robot having the ability to solve the task by changing in real-time some mechanical parameters in function of the environment variations (in our case the slope of the ground). The path will then be generated implicitly by exploiting its intrinsic dynamic. Following this philosophy we can say that modifying some local characteristics of the robot it is possible to achieve a global desired behaviour adapted to the environment.

In this work we investigate the use of dynamical recurrent neural networks to vary in real-time the parameters of the springs of the ankles, and also the frequency of oscillation of the tail, without calculating the exact trajectory the robot must follow. This class of neural networks (Beer & Gallagher, 1992), consists on a coupled set of first order non-linear differential equations whose parameters must be tuned. To do so we use an off-line optimization method through genetic algorithms (Holland, 1975 and Goldberg, 1989). This so-called evolutionary methodology (Nolfi & Floreano, 2000) will be described in Section 4.

In the rest of this chapter, we present the details of our solution and analyse its performance. Robot description is shown in Section 2. Section 3 relates the objective of the framework and the study on the dynamic variation of the robot's parameter. In Section 4, the evolutionary algorithm proposed is explained. Robot implementation on the simulator is shown in Section 5. Results are shown in Section 6. Finally the conclusions can be found in Section 7.

2. Robot Description

2.1 Kinematical description

The system studied in this chapter is a mechanism with the morphology of a biped with a tail (with ankles and without knees) which is able to walk using just one actuator. Each leg and foot forms a four link parallel kinematics chain with only one passive degree of freedom when the flat rectangular foot is attached to the ground. The ankle has a spring which will be described later as it is the main element of the present study. The mechanical structure of the system is shown in Figure 1. Both legs are joined by a body/torso through two revolute joints. The torso mechanism is a new parallel kinematics chain included into the system to maintain both legs parallel to the frontal view (see Figure 2). The tail is attached to the torso through the only actuated joint.

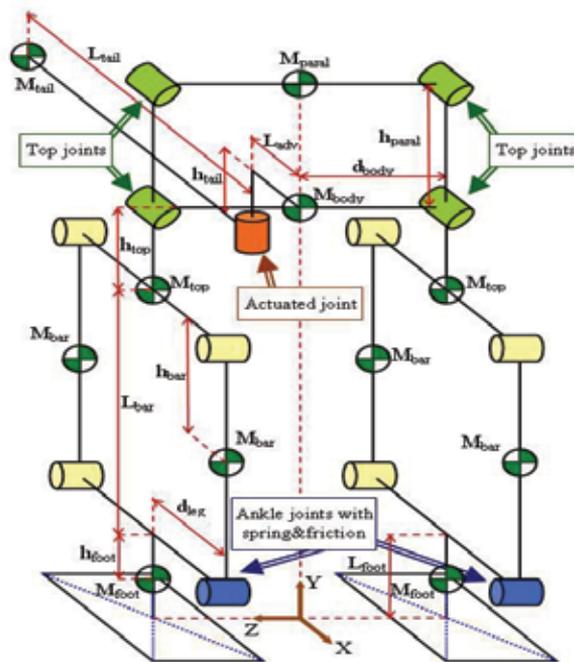


Fig. 1. Model of the biped mechanism.

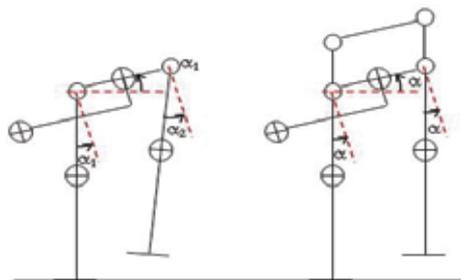


Fig. 2. Torso parallel kinematics chain

Due to the characteristics of the mechanism the model has thirteen joints with only four degrees of freedom, three passive and one actuated. When both feet are on the ground the parallel mechanism of the torso is redundant. The feet contact point with the ground adds two new passive joints. Nevertheless the dimensions of the feet and the model selected for the eight point contact forces with the ground allows us not to consider them from a static point of view. The mechanism remains standing if the springs of the ankles are properly selected.

Some of the important characteristics that this mechanism provides are the following ones:

- The springs in the ankle joints provides the needed force to keep the robot in a stable position, in the case it is not actuated (e.g. switched off). In this state the robot does not need energy consumption.
- When the tail is designed following the condition presented in (Berenguer & Monasterio-Huelin, 2007b), it acts as a counterbalance where the length of a single support phase is no limited in time. It allows the system to remain with a foot raised during an indefinite time. Then, the turn of the mechanism can be reached during a single support phase by adding new joints in the feet or in the hip of the mechanism.
- When all the kinematics parameters have been defined, the behaviour depends only on the tail trajectory and the ankle parameters (friction, stiffness and equilibrium position). We suppose the friction negligible for the rest of joints. In this study we consider the ankle friction (b), the stiffness (K) and the equilibrium position (θ_0) variable parameters. In a real robot we can modify these parameters using a system like MACEPPA (Van Ham et al. 2007) for example.
- If θ_0 is zero, the biped also walks thanks to the force exerted by the tail over the body when the tail is in a lateral position (Berenguer & Monasterio-Huelin, 2007c). Normally the system walks by a combination of this force and the gravity effect.
- Finally, as we showed in (Berenguer & Monasterio-Huelin, 2007b), the robot parameters can be adjusted to obtain a smooth ground contact at the end of each single support phase. This is an interesting characteristic if we try to reduce the dissipated energy during its walk.

The main parameters of the mechanism used on simulations are presented in Table 1. Simulations have been made using MATLAB™ and the SimMechanics toolbox.

Model parameters for simulations					
Name	Value	Name	Value	Name	Value
M_{body}	50.0gr	L_{adv}	0.0mm	H_{paral}	50.0mm
M_{top}	50.0gr	L_{bar}	400.0mm	h_{top}	30.0mm
M_{bar}	200.0gr	L_{foot}	10.0mm	h_{bar}	200.0mm
M_{foot}	150.0gr	L_{tail}	150.0mm	h_{foot}	5.0mm
M_{leg}	450.0gr	d_{body}	50.0mm	h_{tail}	15.0mm
M_{tail}	700.0gr			d_{leg}	20.0mm
M_{paral}	5.0gr			H_{T}	510.0mm
M_{T}	1955.0gr				

Table 1. Parameters for simulations.

2.2 Gait description

The tail of the robot moves in an almost horizontal plane. When the tail is in a lateral position of the mechanism, its mass acts as a counterbalance and produces the rise of one of the feet, where a step movement begins. We will define and describe here seven phases during a stride. Figure 3 shows these phases starting at an equilibrium position of the system with the tail in its central position.

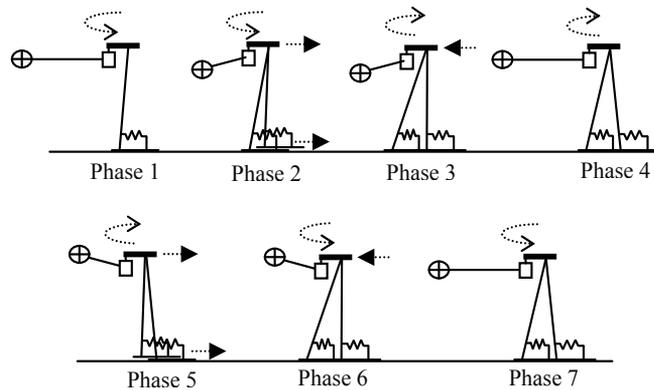


Fig. 3. Phase during a stride

- *Phase 1: Displacement of the tail to a lateral of the mechanism:* Both springs hold the weight of the mechanism, and this one stays almost vertical. We use linear springs in Figure 3 for a better understanding of their effect and because they have been used in the construction of our first real prototype Zappa.
- *Phase 2: Rise of one foot and single support phase:* When a foot rises, only one spring holds the body, so the stance leg falls forward to a new equilibrium position. In this phase, kinetic and potential energies are transformed into elastic energy and stored in the ankle springs. The swing leg moves forward as a pendulum.
- *Phase 3: Contact of the swing leg with the ground:* At this moment the greater kinetic energy losses are due to the collision. The designer must calibrate the mechanism trying to reduce the velocities at this moment and provide a smooth contact between the foot and the ground.
- *Phase 4: Movement of the tail to the other side:* In this double support phase, the projection of the centre of mass of the mechanism moves from one foot to the other. The body moves backwards to a position in which both springs generate opposite torques.
- *Phase 5: Rise of the second foot:* In this phase, the spring of the foot that is in the ground produces enough torque to take the body forward again.
- *Phase 6: New contact of a swing leg with the ground:* Same as phase 3.
- *Phase 7: New displacement of the tail during a double support phase:* If a new stride is desired, this phase represents returning to phase 1. If the tail stops in the middle position, the system will stay in a steady configuration with no energy cost.

3. The objective framework

This paper is centred on the study of the dynamic variation of the parameters of the torsional spring situated at the ankles and the frequency of the tail oscillation. Previous investigations show that these parameters allow the robot to change the step length and the velocity of the walking (Berenguer & Monasterio-Huelin, 2008). To get this dynamic variation, a torque is applied to each ankle following a function given by equation 1.

$$\tau_{ankle} = -K(q - \theta_0) - b\dot{q} \quad (1)$$

The three time varying parameters (K , θ_0 , and b) are the same for both ankles. Bellow we explain this time variation. The spring-actuator is compressed and stores some part of the dynamic energy. At the stand situation this torque does not move the robot but changes the initial angle between the legs and the foot.

The tail is actuated trough a PD controller that follows a sine reference signal,

$$q_{tail} = \frac{\pi}{2} \sin(2\pi ft) \quad (2)$$

The frequency (f) is also a time-varying parameter of the model.

The objective of this paper is to make the robot to go up flat surfaces of different slopes. The hypothesis we envisaged is that the initial inclination or body posture is crucial to get this objective. This hypothesis is bio-inspired because the adaptation of postural orientation to changes in surface inclination is a known reflex behaviour of many animals including humans (Prentice et al., 2004, Kluzik et al., 2007b and Edwards, 2007). It is also known that there exist postural after-effects of stepping on an inclined surface. This effect could be due not to a local adaptive mechanism (i.e. changes in ankle angles) but to a central adaptive mechanism that affects the postural geometry more globally (Kluzik et al., 2007a). In this work we do not consider completely this phenomenon, but because we change local parameters in function of an environment variable (the ground slope) we are not contradicting these discoveries.

Some simulations studies described later show that the hypothesis is true when the robot begin to walk on a constant slope flat surface. But, how to select the parameters when the slope of the ground varies while the robot is in movement? We have decided to add to the robot a simple continuous time recurrent neural network to obtain the time variation of the three parameters of the torque applied to the ankles and the frequency of the tail oscillation (K , θ_0 , b , f).

This approach is different from others explained previously in the sense that the postural phenomenon is here implemented by adjusting mechanical parameters in function of the ground slope. On the contrary, in (Maurer & Peterka, 2005), a two degrees of freedom PID controller (Aström & Häggglund, 1995) is tuned to follow a constant reference signal based on the ground slope. The output of the controller is the torque applied to the ankle.

We have simplified the input to the neural network to only one variable, the ground's slope. To move these ideas into a real robot it is necessary not only to add a mechanism that would

be able to change dynamically the parameters of the ankle, but a sensor that can measure the slope of the ground (a perceptual sensor). In real studies with the robot Zappa (Berenguer and Monasterio-Huelin, 2008) this measurement has been made with accelerometers sensors showing that it is possible to estimate the inclination of the legs on a flat surface with a null slope. Figure 4 shows the accelerometers location and their sensing directions over the model in a sagittal view. (The accelerometers measure $a-g$, where a is the translational acceleration vector of the body and g the gravity vector.)

Due to the kinematical design of the mechanism the reference coordinate system attached to the torso always has the X axis (the direction of the robot advance) parallel to the ground. As a consequence it is possible to get a realistic measure of the ground slope putting an accelerometer on the torso

4. Evolutionary Robotics

Evolutionary Robotics is a methodological tool to automate the design of robots' controllers based on the use of artificial evolution to find sets of parameters for artificial neural networks that guide the robots to the accomplishment of their task (Nolfi & Floreano, 2000). In most cases, the control systems of evolutionary robots are artificial neural networks evolved through genetic algorithms, explained in the next sections.

4.1 Genetic Algorithms

Genetic algorithms are a general-purpose search algorithms inspired by natural population genetics to evolve solutions to problems (Holland, 1975). A genetic algorithm operates on a population of artificial chromosomes by selectively reproducing the chromosomes of individuals with higher performance and applying random changes. Each structure in the population represents a candidate solution to the problem and is ranked according to a *fitness* function. This ordered list is used to create new individuals and repeat the process until the optimization problem is satisfied.

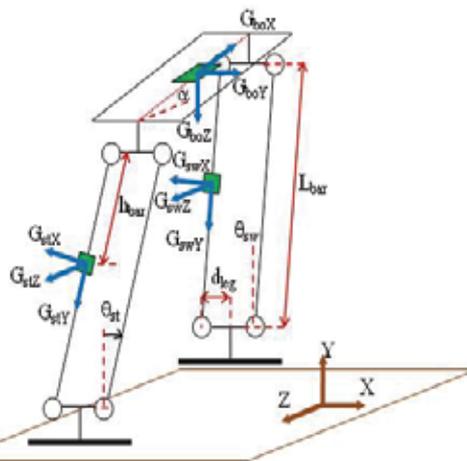


Fig. 4. Accelerometers location and their sensing directions over the model in sagittal view (Tail and frontal parallel link mechanism are not shown).

A genetic algorithm starts with a fix size population of randomly generated individuals and advance towards better solutions by applying genetic operators. Each individual is coded in a chromosome which might implied several variables such as the connection weights of a neural network, gains to apply to a specific input or a time-delay constant for an output. A fitness function must be devised for each problem to be solved.

Given a particular chromosome, the fitness function returns a single numerical fitness which is supposed to be proportional to the adaptation of the individual that the chromosome represents. Once a generation of individuals has been evaluated, selective reproduction starts to create new copies of the best chromosomes in the population. Individuals with higher fitness values tend to leave a higher number of copies for the next generation. Given an individual x_i and its fitness f_i , chromosome reproduction probability p_i is denoted by Equation 3.

$$p_i = \frac{f_i}{\sum_{j=1}^n f_j}, i = 1 \dots n \quad (3)$$

Commonly, each generation has the same number of individuals. The process which determines how the new generation is created depends on many factors. Firstly, the selection algorithm must be chosen. The new generation can be obtained from the last generation whole population, or choosing a given number of best individuals which will create copies of themselves. Some of the best individuals (*Elites*) could also be inherited directly by the next generation.

After selection has been carried out, the construction of the new generation is completed by recombination and mutation. Recombination is achieved by the *crossover* operator, which combines the chromosomes of two parents. Parents are chosen randomly from the new generation and a random point is selected around which the genome is swapped between the two individuals (Figure 4a). The *mutation* operator arbitrarily alters one or more genes of a selected chromosome (Figure 4b).

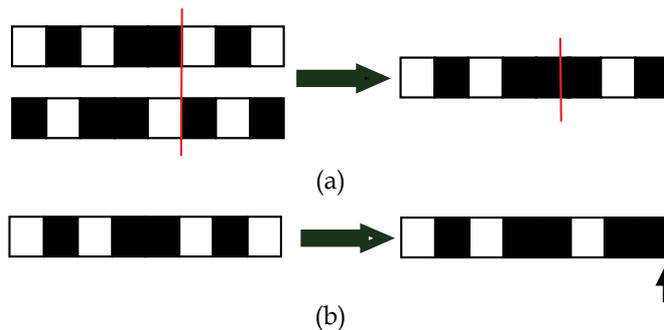


Fig. 5 (a) Crossover: Given two selected genotypes, genetic material is swapped around a random selected point. (b) Mutation: Given one genotype, a random selected gene is mutated. Once the new generation has been created a new evaluation of all individuals

occurs, and the process is repeated for several generations during which the average and fitness values are monitored.

The basics of this simple genetic algorithm and several variations are well described in many previous books (Goldberg, 1989)

4.2 Neural Network

The scientific study of the nervous system led to the need to obtain mathematical models of the neurons. The first model (McCulloch & Pitts, 1943) used a two-state threshold system that followed a stochastic algorithm involving sudden changes of its state at random times. Later (Hopfield, 1984) a deterministic continuous-time mathematical model was proposed for better approaching to the functioning of the real neurons.

Neural networks are interconnected neurons that have many interesting properties (Amit, 1989 and Rumelhart & McClelland, 1986).

In this work we use the continuous-time recurrent neural network (CTRNN) (Beer & Gallagher, 1992) that obeys the following set of non-linear first-order differential equations:

$$\tau \dot{y} = -y + W\sigma(y + \theta) + I \quad (4)$$

where y , I , σ and θ are $[n \times 1]$ vectors and W is an $[n \times n]$ matrix, where y_i is the state variable of the i th neuron (the mean membrane potential); w_{ij} , $j \neq i$ represent synaptic connections from neuron j to neuron i ; the self-interaction w_{ii} represents a simple active conductance; θ_j is a constant threshold or bias; τ_i is a time constant (resistance x capacitance); I_i the external input, and σ_j is a sigmoid function (the mean firing rate of the neuron):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (5)$$

CTRNN's are a special case of the general class of additive neural network models (Grossberg, 1988) which have a complex dynamical behaviour. In (Beer, 2006) the parameter space structure of the CTRNN is systematically studied.

From a mathematical point of view the input to the system is $(I_i + \theta_i)$ since equation 4 can be rewritten in the form

$$\tau \dot{z} = -z + W\sigma(z) + I + \theta \quad (6)$$

using the transformation $z_i = y_i + \theta_i$.

With the transformation $y = Wx + I$ (and supposing W is not a singular matrix) we obtain another useful general representation for analytical studies (Vidyasagar, 1993):

$$\tau \dot{x} = -x + \sigma(Wx + I + \theta) \quad (7)$$

These networks are perfectly suitable for evolutionary robotics tasks due to many characteristics:

- The network is composed of a large number of highly interconnected processing neurons working in parallel to solve a specific problem.
- Neural networks have the ability to learn from experience (that is, from historical data collected in some problem domain).
- Sensors and actuators can be mapped directly into the inputs and outputs neurons.
- Smooth changes on the parameters will correspond to gradual changes of its behaviour.
- After a neural network has been tuned, it can be deployed within an application and used to make decisions or perform actions when new data is presented.
- The network is robust to noise. Due to the weighted sum of input signals, oscillations in individual values do not severely concern the behaviour of the network.

4.3 Implementation

Our robot is composed of a CTRNN of 2 inter-neurons and an arrangement of 1 input neuron and 4 output neurons. Continuously the network receives as input the ground slope. The inter-neuron network is fully connected. Additionally, each inter-neuron receives one incoming synapse from the input neuron. Each output neuron receives one incoming synapse from each inter-neuron (see Figure 6). There are no direct connections between input and output neurons. The states of the output neurons are used to set the frequency of the robot's tail (f), and the three parameters (K , θ_0 and b).

The neural network obeys the state equations 8, 9 and 11, and the output equation 12.

$$x(t) = \gamma(t)g \quad (8)$$

where g is the gain factor, and γ is the ground slope.

$$\tau_i \dot{y}_i(t) = -y_i(t) + \sum_{j=1}^2 w_{h_j h_i} \sigma(y_j(t) + \theta_j) + I_i(t), i = 1, 2, \quad (9)$$

where

$$I_i(t) = w_{I h_i} \sigma(x(t) + \theta_I), i = 1, 2 \quad (10)$$

$$o_i(t) = \sum_{j=1}^2 w_{h_j o_i} \sigma(y_j(t) + \theta_{h_j}), i = 1 \dots 4 \quad (11)$$

$$out_i(t) = \sigma(o_i(t) + \theta_{o_i}), i = 1 \dots 4 \quad (12)$$

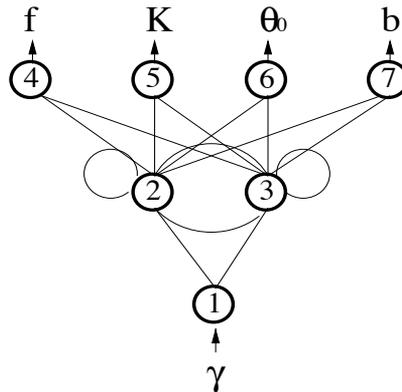


Fig. 6. Neural network architecture implemented. All the feedforward and recurrent connections are drawn (See the text for meaning of the labels).

A simple generational genetic algorithm is employed to set the parameters of the networks (Goldberg, 1989). The population contains 20 genotypes. Genotypes of the first generation are generated randomly. Generations following the first one are produced by a combination of selection with elitism, recombination and mutation. For each new generation, the five highest scoring individuals ("the elite") from the previous generation are retained unchanged. The remainder of the new population is generated by fitness-proportional selection (also known as roulette wheel selection) from the individuals of the old population. Each genotype is a vector comprising 24 real values (i.e., 14 connection weights, 2 decay constants, 7 bias terms, and 1 gain factor). Initially, random genotypes are generated by initializing each of their components to values chosen at random by using a continuous uniform distribution in the $[0,1]$ interval. New genotypes, except "the elite", are produced by applying recombination with a probability of 0.5, or if there has been no recombination, a mutation rule is applied with probability 1.

During evolution, all vector component values are constrained to remain within the range $[0,1]$. Genotype parameters are linearly mapped to produce network parameters with the following ranges: biases $\theta_i \in [-4, 4]$, weights $\omega_{ij} \in [-6, 6]$, gain factor $g \in [-5, 5]$ and decay constants $\tau_i \in [0.1, 100]$.

During evolution each genotype is cloned into the simulated robot and evaluated for a trial. Within a trial, the robot life-span is 25 simulated seconds. In each trial the robot is rewarded by an evaluation function F (see equation 15 below) which seeks to assess the ability of the robot to perform the task.

Once each population has been evaluated the different chromosomes are ranked according to the value obtained from the fitness function, and all the genetic operators are performed according to the rules described above.

5. Simulator

Robot is simulated thanks to the *SimMechanics* toolbox included in the *Matlab* software. Previous studies have depicted the simulation schema and only modifications on the previous structure are shown in this chapter. For further details see (Berenguer & Monasterio-Huelin, 2007b).

First modification is the mechanisms included for the on-line slope modification. In principle modifying the ground was a hard task due to the simulator architecture. The problem's solution was the inclusion of a mechanism to alter the gravity shown in Figure 7.

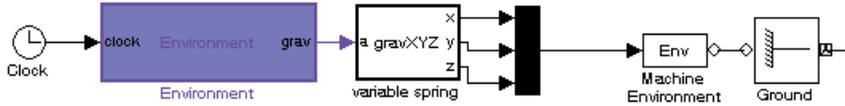


Fig. 7. Online gravity modification.

The gravity acts as a function of the real slope, where its 3 components (g_x, g_y, g_z) are defined in equation 13.

$$\begin{cases} g_x = -9.81 \sin(\gamma) \\ g_y = -9.81 \cos(\gamma) \\ g_z = 0 \end{cases} \quad (13)$$

Another enhancement implemented is the change of the previous static springs by the variable ones. As shown in Figure 8, this modification allows the robot to control its spring force. This spring is implemented following the Equation 1 explained in previous sections, where parameters q and dq/dt come from the ankle sensor and K , b and θ_0 are outputs from the neural network.

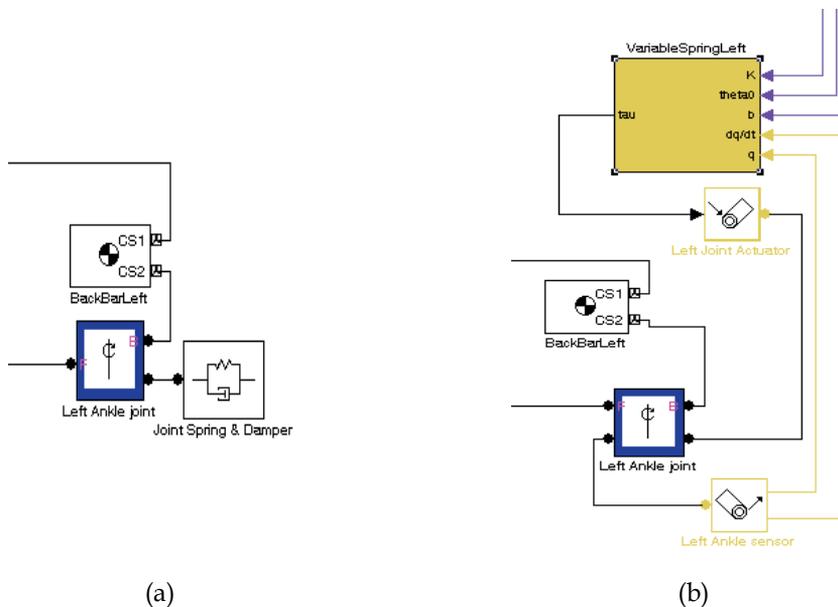


Fig. 8. (a) Spring implementation on previous works where the spring has constant parameters. (b) Variable spring implementation. The spring has time varying parameters.

Final modification is the inclusion of the neural network controller explained in Section 4.3. This CTRNN is in charge of getting the environment input (ground slope) and give the actuating parameters to the tail and variable springs. We observe that equation 9 can be rewritten as shown in Equation 14 and the block diagram is shown in Figure 9.

$$\tau_i \dot{y}_i(y) = -y_i(t) + \Delta_{h_i}, i = 1,2 \tag{14}$$

The task performs in the simulator for 25 simulated seconds. During this time the slope of the ground (translated into gravity changes) is increased each 5 seconds in 2°, starting in 0° and finishing in 8°. And external program is in charge of providing the genotype to the network and collecting the necessary output data. Once the trial has finished, the program stops the simulation, scores the individual and run next individual genotype. The individual is evaluated according to Equation 15, which measures the distance travelled by the robot during the experiment.

$$F = \sqrt{(x_f - x_0)^2 + (z_f - z_0)^2} \tag{15}$$

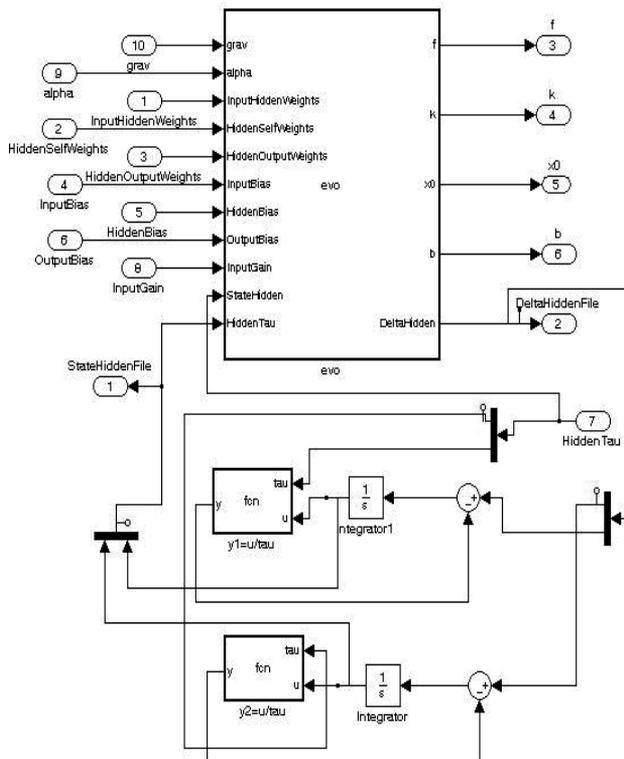


Fig. 9. CTRNN implemented on the simulator.

When a generation has been evaluated, the system rearranges the genotypes creating a new generation based on the elitism, mutation and crossover. This cycle is repeated for several evaluations until the robot reaches its travelled distance limits.

6. Results

Through a collection of patient simulation tests we have discovered that the robot can go up different constant slopes in the ranges shown in Table 2. This study has been made to accelerate the evolution process, but it has the handicap that must be applied to unnormalize the output of the neural network. This empirical study might be avoided in the future because it could not be necessary that the parameters change suddenly with the slope.

f	K	b	θ_0	slope (γ)
0.5-0.7	9-10	0.4-0.6	0.05-0.07	0°-6°
			0.07-0.12	6°-8°

Table 2. Output neural network parameters range.

We run simulations for 70 generations. As shown in Figure 10, the fitness reached a plateau around the value of 3.0 after generation 59.

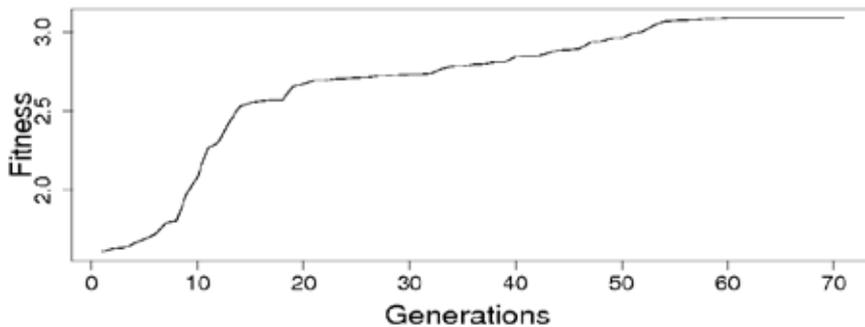


Fig. 10. Fitness of the robot at each generation.

The best performing genotype resulting from the evolutionary process was decoded again into the simulated robot to make measurements about the behaviour of the robot. We first studied the behaviour of the neural network, observing the performance of the hidden

neurons. Figure 11 shows the state values and delta values of both hidden neurons.

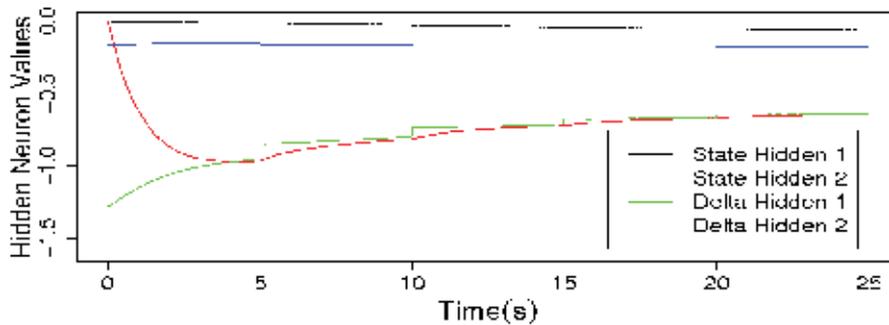
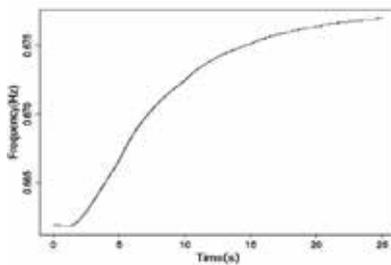


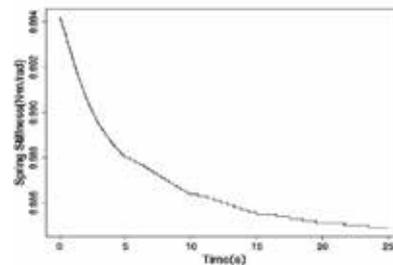
Fig. 11. Delta and State values for both hidden neurons in the best performing genotype.

As shown in the previous figures, hidden neuron delta values change abruptly with input changes (each 5 seconds) while the state values do it smoothly.

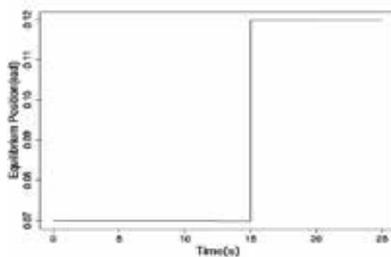
Output values for the neural network are shown in Figure 12. These values change over the time due to the environment changes. We observe that frequency is changing its value slowly following a transient state towards a final steady-state which is not able to reach. Stiffness (K) (figure 12b) is also following an effortlessly change, but adjusts on its inclination are found on the ground slope changes. Finally θ_0 (figure 12c) and b (figure 12d) modify their values abruptly, just one change for θ_0 matching one ground slope change ($t=15s.$) due to the output value range forced by design.



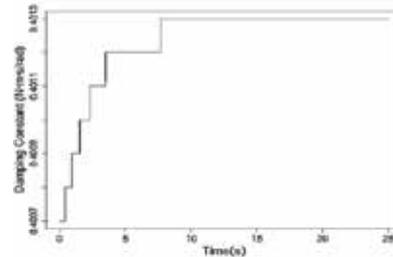
(a)



(b)



(c)



(d)

Fig. 12. (a) Tail frequency oscillation (f) (b) Spring stiffness (K) (c) Equilibrium position (θ_0) (d) Damping constant (b) values for the best genotype in a complete trial.

The spring values coming from the network go to the variable spring which sensing the ankles' angular position and velocity (Figure 13a) impinge a torque on both ankles (Figure 13b). We observe periodic oscillations on all the signals, due to the movement of the robot. While one foot is on the ground the other is moving forwards with some small oscillations backwards. Once the previous foot has touched the ground the other one starts its movement. We can also notice small modifications on the position and velocity of the ankle joint depending on the different ground slopes. During each stable ground inclination the ankle keeps the oscillatory movement as a periodic signal and changes its strength when the inclination varies. These modifications will alter the body velocity and step's length as shown in Figure 14.

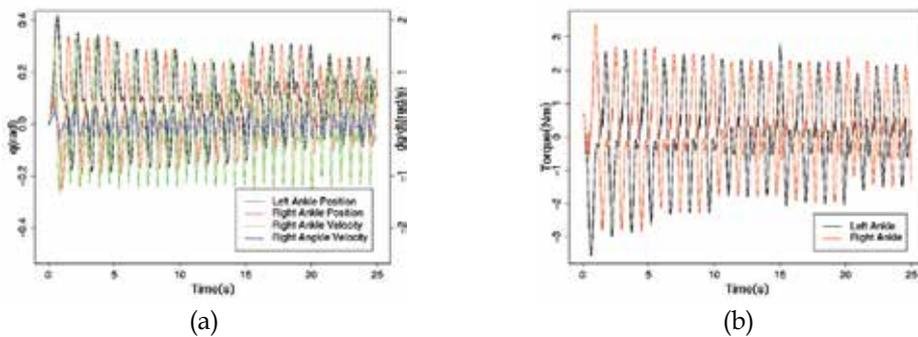


Fig. 13. (a) q , dq/dt and (b) torque values of the variable spring.

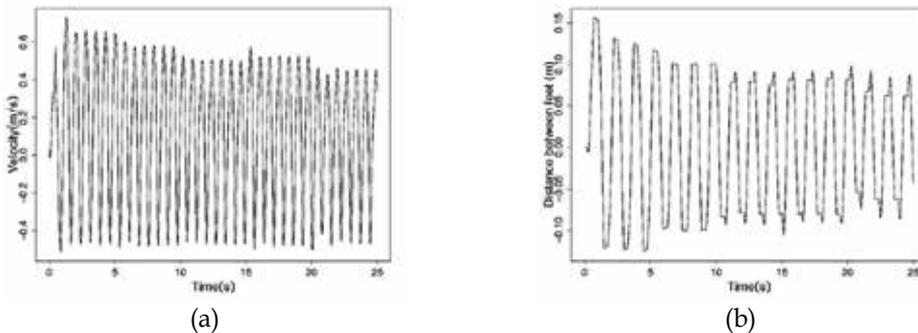


Fig. 14. (a) Body velocity and (b) distance between robot's feet during a trial.

Figure 14a shows that the robot's average speed on a flat terrain is 0.17m/s but it is reduced to 0.08m/s on an 8 degrees ground slope. This variation can also be appreciated in Figure 14b where we observe a starting step size of 0.12m and is reduced to 0.08m at the end of the trial. The maximum distance walked, average speed and average acceleration are shown in Figure 15, related to the instantaneous ground slope.

One of the important characteristics on the robots is its quasi-passiveness which allows the robot to have a reduced consumption as can be appreciated in Figure 16. The instantaneous power consumed by the actuator of the tail has been calculated using equation 15, and the energy by equation 16.

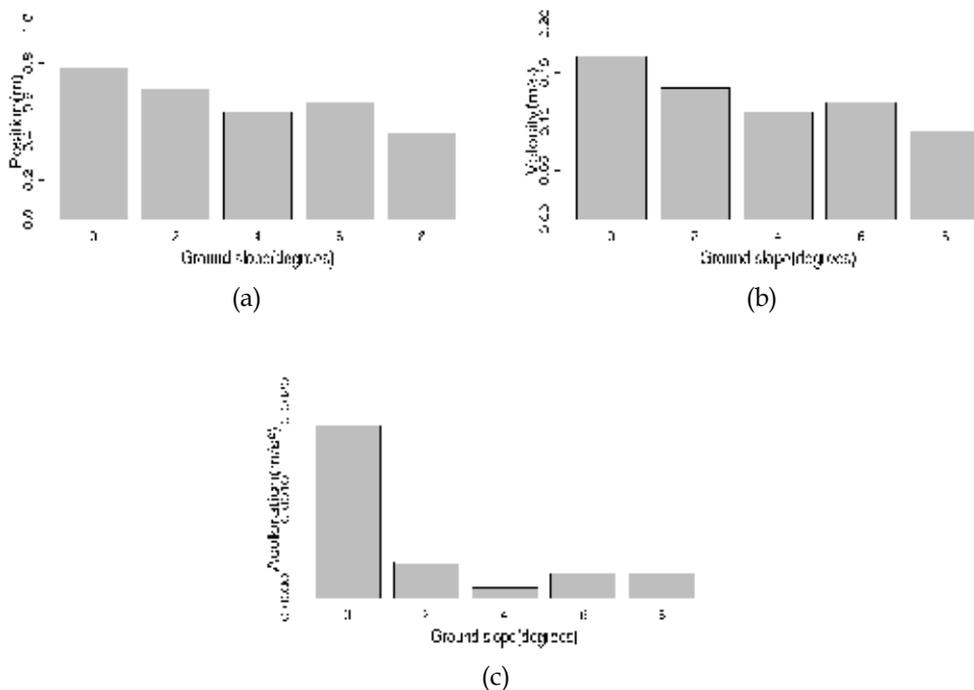


Fig. 15. (a) Distance travelled (b) Average velocity and (c) Average acceleration of the centre of mass of the robot related to each different inclination angle.

$$P(t)=\tau(t)\dot{q}(t) \tag{15}$$

$$E=\int_0^t |\tau(t)\dot{q}(t)| dt \tag{16}$$

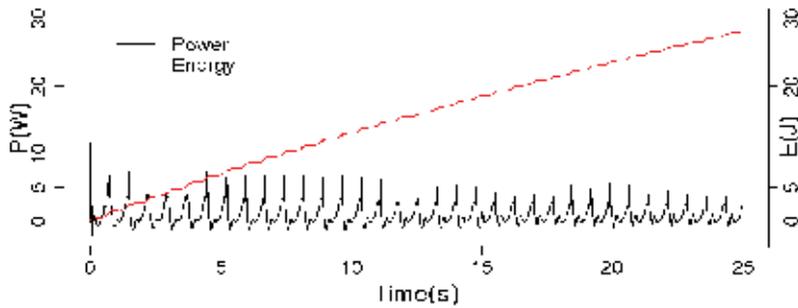


Fig. 16. Tail joint mechanical instantaneous power and energy consumption during a trial. Finishing with the results evaluation we show the limit cycle of the robot during the trial for the different ground inclinations on Figure 17.

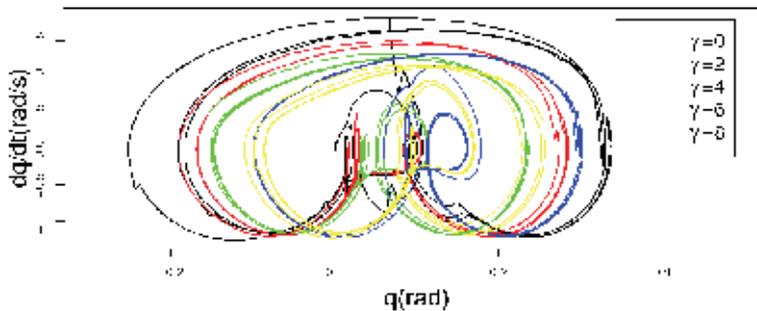


Fig. 17. Limit cycle on different inclinations during a trial.

Once the evaluation of the best genotype has been finished, we tried to evaluate it in different situations. Firstly we tried to test the robot on different slopes. Tests show that the robot is able to go up slopes up to 12° and goes down slopes of -2° , but no more.

Later we tried to test initial conditions problems, about seeing if the robot was able to start in different slope conditions. Results show that robots can start in positive slopes up to 10 degrees but no 12 degrees even if it is able to walk over them. The robot can satisfactorily start in -2° slopes.

Finally Figure 18 shows two different trials for these configurations: In (a) robot start over a 10° slope which reduces 2° each 5 seconds. (b) shows a robot starting at a -2° ground slope and finishing in a 12° positive inclination.

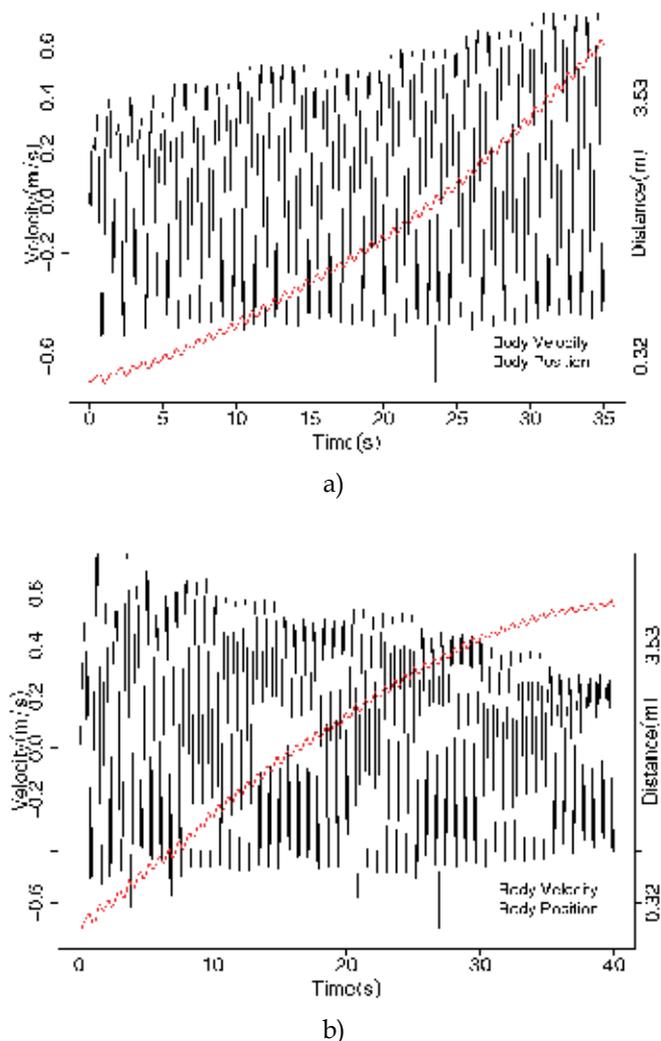


Fig. 18. (a) Post-evaluation for a 35 seconds simulation, starting on a 10 degrees slope and going down to -2 degrees. (b) Post-evaluation for a 40 seconds simulation, starting on a -2 degrees slope and going up to 12degrees.

7. Conclusions

It is shown through computer simulations that it is possible for a biped robot with a tail to go up a small slope.

We have followed an evolutionary methodology to tune a continuous-time recurrent neural network. The evolution is made inside an environment that changes suddenly from one slope to another without stopping the robot.

The neural network changes in real-time the parameters of the ankle springs and the oscillation frequency of the tail, without giving an explicit path in the configuration space of the robot. It is also shown that the robot performs well for new conditions of the environment not considered in the evolution phase: for higher slopes and for different initial conditions.

This work almost ends the first necessary investigations to build a new real Zappa robot.

New mechanisms for the ankles and modifications on the electronics of the robot are needed. Some studies on the sensors of the robots must be carried out. These studies could offer a better performance on the task and give some perceptions to the robot about its position (inclination) related to the ground on its postural state.

Nevertheless some theoretical studies, both kinematical and dynamical are still needed.

8. References

- Alexander, R. M. (2005). Physiology Walking made simple, *Science Magazine*, Vol. 308, No. 5178 (April, 2005) 58-59, ISSN 0036-8075.
- Asano, F., Luo, Z. & Yamakita, M. Some Extensions of Passive Walking Formula to Active Biped Robots, *Proceeding of the 2004 IEEE International Conference on Robotics and Automation*, pp. 3797-3802, ISSN 1050-4729, New Orleans, LA, April 2004, IEEE Press, Washington D.C.
- Asano, F., Luo, Z. & Yamakita, M. (2005). Biped Gait Generation and Control Based on a Unified Property of Passive Dynamic Walking. *IEEE Transactions on Robotics*, Vol. 21, No. 4, (August, 2005) 754-762, ISSN 1552-3098.
- Beer, R. D. & Gallagher, J. C. (1992). Evolving Dynamic Neural Networks for Adaptive Behaviour. *Adaptive Behavior*, Vol. 1, No. 1 (June, 1992) 91-122, ISSN 1059-7123.
- Beer, R.D. (2006) Parameter Space Structure of Continuous-Time Recurrent Neural Networks. *Neural Computation*, Vol. 18, No. 12, (December, 2006) 3009-3051, ISSN 0899-7667.
- Berenguer, F. & Monasterio-Huelin, F. (2006). Easy design and construction of a biped walking mechanism with low power consumption, *Proceedings of the 9th Int. Conf. on Climbing and Walking Robots*, pp. 96-103, Brussels, Belgium, September 2006.
- Berenguer, F. & Monasterio-Huelin, F. (2007a). Trajectory planning using oscillatory chirp functions applied to bipedal locomotion, *Proceedings of the 4th Int. Conf. on Informatics in Control, Automation and Robotics*, pp. 70-75, ISSN 0738-4602, Angers, May 2007, AAI, La Canada CA.
- Berenguer, F. & Monasterio-Huelin, F. (2007b). Stability and smoothness improvements for an underactuated biped with a tail, *Proceedings of the 2007 IEEE Int. Symp. on Industrial Electronics*, pp. 2083-2088, ISBN 1-424-40754-0, Vigo, Spain, June 2007, IEEE Press, Piscataway N.J.
- Berenguer, F. & Monasterio-Huelin, F. (2007c). Locomotion of an Underactuated Biped Robot Using a Tail, *In: Bioinspiration and Robotics: Walking and Climbing Robots*, Maki K. Habib (Ed.), 15-38, I-Tech Education and Publishing, ISBN 978-3-902613-15-8, Vienna, Austria.
- Berenguer, F. & Monasterio-Huelin, F. (2008). Zappa, a Quasi-passive Biped Walking Robot with a Tail. Modeling, Behavior and Kinematic Estimation using Accelerometers. *IEEE Transactions on Industrial Electronics*, (September, 2008), (in press).

- Boeing, A. & Bräunl, T. (2004). Evolution of Locomotion Controllers for Legged Robots. *Robotic Welding, Intelligence and Automation (LNCIS)*, No. 299, (2004) 228–240, ISSN 0170-8643.
- Collins, S. H. & Ruina, A. (2005). A Bipedal Walking Robot with Efficient and Human-Like Gait". *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp.1983-1988, ISSN 1050-4729, Barcelona, Spain, April, 2005, IEEE Press, Washington D.C.
- Edwards, W.T. (2007) Effect of joint stiffness on standing stability. *Gait & Posture*, Vol.25, No. 3, (March 2007) 432-439, ISSN 0966-6362.
- Geyer, H. & Seyfarth, A. (2006). Walking and running dynamics explained by compliant legs: consequences, general insights, and future directions. *Journal of Biomechanics*, Vol. 39, Supp. 1, (2006) 361, ISSN 0021-9290.
- Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, ISBN 0-201-15767-5, New York, NY.
- Grossberg, S. (1988) Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural Networks*, Vol. 1, No. 1, (1998) 7-61, ISSN 4083-6601
- Holland, J.H. (1975). *Adaptation in Natural and Artificial System*, MIT Press, ISBN 0-472-08460-7, Ann Arbor, MI.
- Hopfield, J.J. (1984) Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings Of the National Academy of Sciences of the United States of America*, Vol. 81, No. 10, (May 1984) 3088-3092, ISSN 0027-8424.
- Hurmuzlu, Y., Génot, F. & Brogliato, B. (2004). Modeling, stability and control of biped robots—a general framework. *Automatica*, Vol. 40, No. 10, (2004) 1647-1664, ISSN 0005-1098.
- Iida, F. & Pfeifer, R.. Sensing through body dynamics. *Robotics and Autonomous Systems*, Vol. 54, No. 8, (August,2006) 631–640, ISSN 0921-8890.
- Iida, F., Rummel, J. & Seyfarth, A. (2007). Bipedal Walking and Running with Compliant Legs, *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pp. 3970-3975, ISBN: 2-130-31308, Rome, Italy, April 2007, IEEE Press, Piscataway, NJ.
- Iida, F., Minekawa, Y., Rummel, J. & Seyfarth, A. (2008). Toward a Human-like Biped Robot with Compliant Legs. *Robotics and automation systems*. (2008), (in press).
- Kluzik, J., Horak, F.B. & Peterka, R.J. (2007a). Postural After-effects of Stepping on an Inclined Surface. *Neuroscience Letters*, Vol. 413, No. 2, (February 2007), 93-98, ISSN 0304-3940.
- Kluzik, J., Peterka, R.J & Horak, F.B..(2007b) Adaptation of postural orientation to changes in surface inclination. *Experimental Brain Research*, Vol. 178, No. 1, (March 2007) 1-17, ISSN 0014-4819.
- Kuo, A. D. (2007). The six determinants of gait and the inverted pendulum analogy: A dynamic walking perspective. *Human Movement Science*, Vol. 26, No. 4, (2007) 617–656, ISSN 0167-9457.
- Löffler, K., Gienger, M., Pfeiffer F. & Ulbrich, H. (2004). Sensors and Control Concept of a Biped Robot. *IEEE Transactions on Industrial Electronics*, Vol. 51, No. 5 (October 2004) 972-980, ISSN 0278-0046.
- Magdalena, L. & Monasterio, F. (1995). Evolutionary-based learning applied to fuzzy controllers. *Proceedings of 1995 IEEE International Conference on Fuzzy Systems: the*

- international joint conference of the Fourth IEEE International Conference on Fuzzy Systems and the Second International Fuzzy Engineering Symposium*, pp. 1111 – 1118, ISBN: 0-780-32462-5, Yokohama, Japan, (March, 1995), IEEE Press, Piscataway, NJ.
- Matsushita, K., Yokoi H. & Arai, T. (2006). Pseudo-passive dynamic walkers designed by coupled evolution of the controller and morphology. *Robotics and Autonomous Systems*, Vol. 54, No. 8 (2006) 674-685, ISSN 0921-8890
- Maurer, C. & Peterka, R.J. (2005). A New Interpretation of Spontaneous Sway Measures Based on a Simple Model of Human Postural Control. *Journal of Neurophysiology*, Vol. 93, No. 1, (January, 2005) 189–200, ISSN 0022-3077.
- McCulloch, W.S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, Vol. 5, No. 4, (December, 1943), 115-133, ISSN 1522-9602
- McGeer T. (1990). Passive Dynamic Walking. *The International Journal of Robotics*, Vol. 9, No. 2, (1990) 62-82, ISSN 0278-3649.
- McMahon, T. A. (1984). *Muscles, reflexes, and locomotion*, Princeton University Press, ISBN 0-691-08322-3, Princeton, NJ.
- Nolfi, S. & Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence and Technology of self-Organizing Machines*, MIT Press, ISBN 0-262-14070-5, Cambridge, MA
- Ohashi, E., Aiko, T., Tsuji, T., Nishi, H. & Ohnishi, K. (2007). Collision Avoidance Method of Humanoid Robot with Arm Force. *IEEE Transactions on industrial electronics*, Vol. 54, No. 3, (June, 2007) 1632-1641, ISSN 0278-0046.
- Pfeifer, R. and Bongard, J. (2007) *How the body shapes the way we think. A new view of Intelligence*, MIT Press, ISBN 0-262-16239-5, London England.
- Prentice S.D., Hasler, E.N., Groves, J.J. & Frank, J.S. (2004). Locomotor adaptations for changes in the slope of the walking surface. *Gait and Posture*, Vol. 20, No. 3, (December, 2004) 255-265, ISSN 0966-6362.
- Rumelhart, D.E. & McClelland, J.L. (1986). *Parallel Distributed Processing. Vol1 : Foundations.*, MIT Press, ISBN 0-262-18120-7, Cambridge, MA
- Van Ham, R., Vanderborght, B., Van Damme, M., Verrelst, B. & Lefeber, D. (2007). MACCEPA, the Mechanically Adjustable Compliance and Controllable Equilibrium Position Actuator: Design and implementation in a biped robot. *Robotics and Autonomous Systems*, Vol. 55, No. 3, (2007) 761-768, ISSN 0921-8890.
- Vidyasagar, M. (1993). *Nonlinear Systems analysis*, Prentice-Hall, ISBN: 0-13-623513-1, Englewood Cliffs, NJ.

Robotic Proximity Queries Library for Online Motion Planning Applications

Xavier Giralt, Albert Hernansanz, Alberto Rodriguez and Josep Amat
*Technical University of Catalonia
Spain*

1. Introduction

One of the most important problems to solve in robotics is the collision avoidance between a robot and its environment. A robot should perceive the risk and have a reactive behaviour before an imminent collision occurs. Path planning is a hard computational problem, so having a fast tool to calculate collisions is a key factor to decrease the necessary time to generate safety trajectories. In applications where no path planning exists, for instance in manual guidance or teleoperation, a real time collision detector is needed so as to avoid collisions and to be able to interact with the environment, for example sliding over a surface. The knowledge of minimum distances between robots or objects that share a workspace enables robots to behave in a predictive way. In the human-robot interaction field, virtual fixtures can be used both to prevent collisions and help the human operator by increasing his performance. In this kind of applications minimum distance and collision detection must be known in real time.

A new library: Robotic Proximity Queries (RPQ) package has been developed to deal with these requirements, using PQP as the proximity query engine. The original package has been used to optimize the queries when working with open kinematic chains, like robots. These optimizations have been done with the aim of improving the time performance of the generic package and simplifying its use in robotic environments. A system composed of two robots has been used as a test bed to show the performance of the RPQ library.

Two applications that benefit from RPQ performance are presented. First, a robotic assisted surgical application, with an assisted cut of a rigid tissue. The surgeon guides freely the driller held by a slave robotic arm that avoids undesired drillings by means of virtual protections. The second application is a dynamic expansion of the working space by means of multirobot cooperation. With these applications, not only proximity queries are shown, but also the graphical interface and the use of a virtual robot based on RPQ.

2. Related Work

During the last years, great efforts have been devoted to the development of efficient collision detection algorithms due to their wide range of applications, such as CAD/CAM, manufacturing, robotics, simulation and computer animation. A wide study of the performance and applicability of such methods can be found in (Geiger, 2000). Proximity

query algorithms vary in terms of their range of applicability and the type of queries they can solve, mainly collision detection, minimum distance computation and interpenetrations modelling. Although most algorithms allow as input triangulated meshes of 3D points, they differ in the way those points are pre-processed and represented internally in order to speed up specific queries.

There is a wide set of methods that rely on Lin-Canny or Gilbert-Johnson-Keiethi like algorithms for computing minimum distances between pairs of objects as I-Collide, Swift, Swift++, SOLID, DEEP..., but they are only applicable to convex polytopes (Ehmann et al., 2000),(Ehmann et al., 2001) and (Kim et al., 2002). This restriction makes them inappropriate for RPQ purposes, due to the need to deal with more general geometric models.

More general collision detection methods usually base their efficiency on pre-computed representations by means of hierarchies of bounding volumes. Their differences rely on the specific type of bounding volumes used, ranging from binary space decompositions, spheres trees to oriented bounding boxes (OBB).

Among this set of algorithms, RAPID and PQP turn to be those that have both, fewer restrictions in the range of allowable geometric models and an easier application programming interface (API). Both of them use oriented bounding boxes for performing collision tests, and have similar time performances. However, the fact that PQP offers a wider range of queries, including minimum distance computation and tolerance tests makes PQP the best option for the proximity queries engine of RPQ, the Robotics Query Package presented in this paper.

3. Library Description

The goal of the Robotic Proximity Queries (RPQ) library is to offer an easy, modular and fast proximity query package oriented to robotics. As explained above, the aim of the project is not the development of a new collision detector, but specialize an existing one into the robotics field.

As described in the previous section, there is a wide set of general purpose proximity query packages. The criterions used to choose PQP as the best candidate for the development of RPQ are:

- Types of proximity queries available.
- High time performance on proximity queries.
- Ability to use geometrical models based on triangulated meshes of points.
- Lack off restrictions on possible geometrical models.
- Open source library.
- Easy API.

The PQP library has been developed by UNC Research Group on Modelling, Physically-Based Simulation and Applications and offers three different kind of queries:

- Collision detection: detecting whether two models overlap, and optionally, give the complete list of overlapping triangle pairs.
- Distance computation: computing the minimum distance between a pair of models.

-Tolerance verification: determining whether two models are closer or farther than a given tolerance distance.

RPQ has been implemented in C++ language and its graphical interface has been developed using OpenGL. The RPQ library can be easily integrated into any software application.

The library interface allows non expert programmers to use it in an easy manner. The graphical interface is a separate module, allowing the programmer to decide whether using it or not. Fig. 1. shows the integration of the library and its graphical interface into a generic application.

3.1 Class Implementation

The RPQ library is based on the Object Oriented paradigm. Focused on this paradigm, and based on robotic environments, three main classes have been developed: Scenario, Object and Robot.

Scenario

Scenario is the workspace where the objects cohabit. Concerning its implementation, Scenario is a class that contains all the objects (Robots and generic objects), a global reference frame, and all the methods necessary to generate the proximity query.

Object

An Object is the minimum entity that exists in a Scenario. There are two types of Objects: simple and complex. A simple Object is represented by a geometrical model composed of a set of triangles referred to a frame tied to the Object. The Object has also a transformation matrix to refer itself to the world reference frame. A complex Object is an Object composed of a set of geometrical models with joints (rotational or prismatic) between them. Thus, a complex Object is an open kinematic chain composed of sub objects. The transformation matrix M_i refers *subobject_i* to *subobject_{i-1}*. The transformation matrix M_0 refers the object base *subobject₀* to the world. The object stores its own geometrical model. Concerning its implementation, an Object is a class containing the geometrical model, the transformation matrix and a set of methods to position and to orient itself in space. This class also contains methods to calculate the different detail representations of its geometrical model.

Robot

A Robot is a particularization of a complex Object where each of its links is represented by a simple Object. A Robot has a set of functions to make a complex Object as similar as possible to a real robot. For instance, the spatial relationship between links is described using the Denavit-Hartenberg notation. Direct and inverse kinematics can be calculated considering the robots own restrictions (joint limitations, configurations, etc). Concerning implementation, the class Robot is derived from the class Object. Robot adds all the functions that are necessary to control a robot. For instance joint positioning of the robot (direct kinematics), position and orientation of its tool center point (inverse kinematics), change of the robot configuration, joints overshoot...These added functions with respect an Object are very helpful when a new robot is created or used in robotic applications like simulators, path planners, etc.

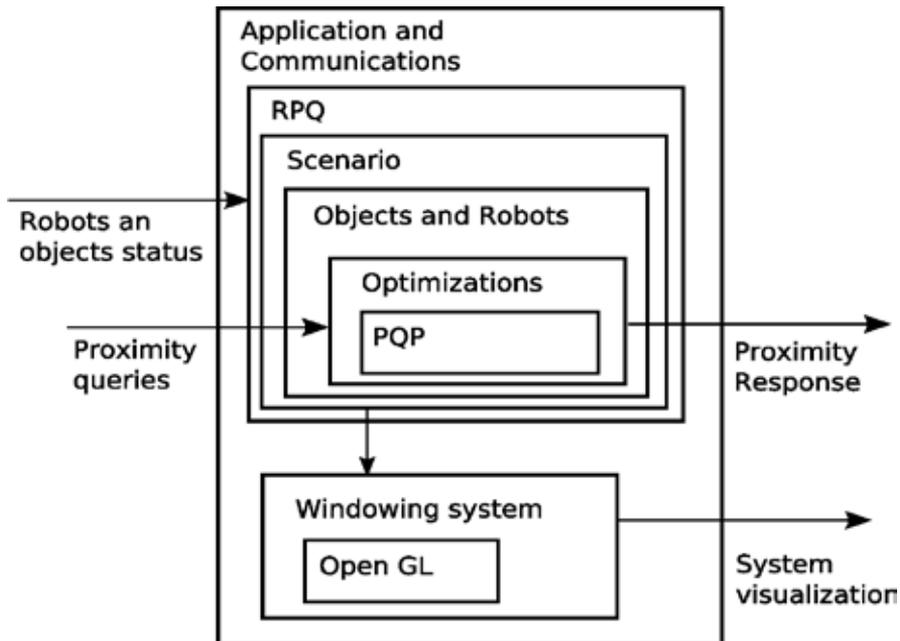


Fig. 1. Schema of integration of RPQ into a generic application

3.2 Optimizations

PQP is a generic package that does not use the knowledge of the object's kinematics. In contrast, RPQ is oriented to robotics, and the knowledge of robot's kinematics is the base of the optimizations that specialize it. RPQ is designed to answer proximity queries between two robots or between a robot and any kind of rigid object. Three optimizations have been developed and tested to improve the performance offered by PQP.

Different resolution levels of object's representation

Objects can be represented in very different resolution levels. The idea of this optimization is to use the simplest representation models (minimum number of triangles) to discard collisions. The lower the number of triangles of the geometric model is, the faster the collision queries are executed.

Three resolution levels are used to represent robots and two for the rest of objects. The highest resolution level is the complete geometrical model. The second level is the oriented bounding box (OBB) of each sub object in which a complex object is divided. The lowest resolution level is the bounding box of the whole complex object. This level is only defined for complex objects with more than a sub object, as in robots with several links. There are other possible intermediate resolution levels that can be used, for instance the convex hull. It offers a good ratio between resolution and the quantity of triangles, although it does not reduce it as drastically as the low resolution levels chosen.

This optimization is useful in two different situations. First, in applications where no high precision is required, for instance when the precision of the OBB or the convex hull of each link is enough. The second situation occurs when the different resolution levels are used in a complementary manner.

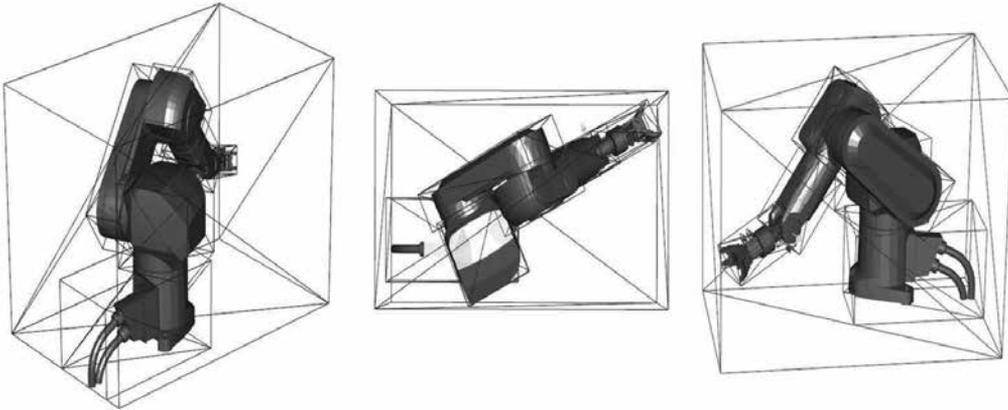


Fig. 2. Robot with three resolution level representation: The geometrical model of each link (L3), the OBB of each link (L2) and the whole bounding box(L1).

When a collision query is performed, a low to high resolution level list of collision queries is generated. Starting with the lowest resolution level, queries are generated until any collision can be completely discarded. For instance, if a possible collision between two 6 DOF robots is studied, the first query is done between the bounding boxes of each robot. If the collision can not be discarded, then the bounding box of each link is used. If at this level collisions still can not be discarded, the geometrical models of each link are checked. Fig. 2. shows a Robot with its three resolution levels of representation.

A test has been developed for the evaluation of the performance of the optimizations. It consists on a couple of virtual robotic arms (Staubli RX60B) that are placed one close to the other in a scenario. The geometrical models used for the test are high resolution (composed of 23012 triangles each). The robots are placed at a variable distance between them and the scenario is checked for collisions for an equidistributed set of joint positions in their 6 DOF. This test allows us to study the dependency on the performance of the proposed improvements in terms of the probability of collision.

Collision queries sorted using a Weight Matrix

This optimization is based on two assumptions. The first one is that the goal of the query is just to know whether there is a collision or not, but not the number of them. The second assumption is that the kinematics and the morphology of the robots are well known.

Given these assumptions, the objective is to find quickly whether there is collision or not, by means of minimizing the number of partial collision queries. The knowledge of the kinematics and the morphology of the robots give us the possibility of assigning an a priori collision probability to each link of the robot with respect to the rest of robots and objects present in the same workspace. During execution time, these probabilities are automatically updated depending on the result of the collision queries (Probability increases in case of detecting a collision and decreases otherwise). Therefore, a weight matrix C is generated combining the probability of collision between each pair of objects in the workspace. These weights determine the order of the collision queries. A simple way to assign a collision probability to the links of a robot is to assign higher probability to those links that are farther in the kinematic chain, with respect to the base of the robot.

Collision Matrix

The Collision Matrix is a binary matrix that reflects the possibility of collision between two objects. If the collision matrix indicates that a collision between two objects is impossible, its correspondent collision query is not performed. Of course, a matrix query is much less expensive than a collision query in computational terms. This optimization improves the performance of the system when a high number of collision queries are discarded by the Collision Matrix.

The performance of the Collision Matrix has been studied using the same test designed for testing the use of different levels of representation. The results are shown in Fig. 3. The farther the robots are, the lower is the number of links that can collide, and therefore, the higher is the number of queries that are solved with the Collision Matrix. As it is shown in the figure, using the Collision Matrix the number of collision queries decreases, so does the time to solve the query.

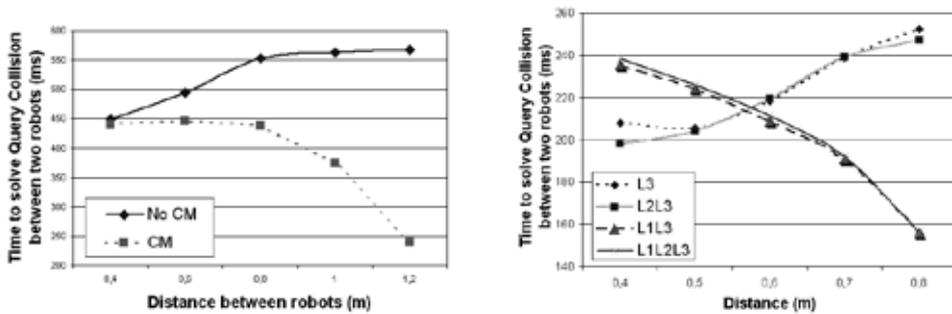


Fig. 3. Time necessary to solve a collision query between two Staubli RX60B robots.

a) Comparison between using the Collision Matrix (CM) or not (NoCM)

b) Comparison using different resolution levels. L3: Geometrical model of each link. L2: The OBB of each link. L1: Bounding box of the robot.

Each one of the proposed optimizations improves the performance of the original library, PQP. However, the combined use of all of them improves even more the global performance of the application. First of all, a query collision between the whole bounding boxes of both robots is performed. If at this level the collision cannot be solved then it is necessary to study collisions among the whole set of links of both robots. The order in which these queries must be performed is given by the Weight Matrix. The query finishes as soon as a collision appears between a pair of links either in the second or third level, or when all pairs have not reported any collision. For each pair of links, the second and third representation levels are studied consecutively, so if a collision is detected in the second level, the third level has to be studied as well.

4. Applications

One of the advantages of making RPQ generic (although it is optimized for robots) is that this library can be applied in a wide set of applications. For instance, in Fig. 4. a robotic aid for laparoscopic surgery is shown. In this application the robot manipulates a laparoscope attached at the end effector. In this section, two specific applications are described in detail. First one has the aim of helping the surgeon to make a cut on a rigid tissue, for example in the cranium, avoiding undesired collisions between the patient and the driller. The surgeon

guides freely the driller that is held by the robot acting in a passive mode, allowing all movements except those which produce undesired collisions. The second application describes a multirobot cooperation system that allows the dynamic expansion of the working space.

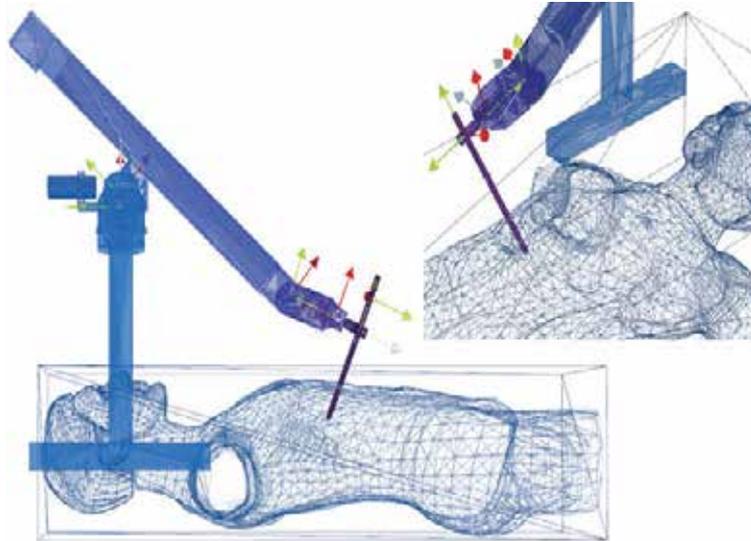


Fig. 4. RPQ library applied in robotic aids for laparoscopic surgery procedures.

4.1 Virtual Fixture and Surface Navigation

Virtual fixtures are constraints that rule the behaviour of the robot that are specifically designed to prevent motion into a forbidden region of the workspace. In this work, the surgeon guides freely a driller held by the slave robot. The main idea is to develop a system helpful for the surgeon that prevents undesired drillings.

Strategy Description

The main goal of the system is to give the robot a reactive behaviour by means of the definition of virtual objects in the PQP Scenario with two objectives:

- Protect the patient from the robot and the driller.
- Help the surgeon to locate the desired cutting area.

PQP's ability to check collisions in real time allows us not only to achieve these objectives but to operate in an efficient manner.

Throughout a simple interface, in the pre-operative phase, the surgeon specifies the exact location and shape of the cut that must be done. With that information, the system generates a shield that covers the patient while it adapts perfectly to the target area defined by the surgeon, as shown in Fig. 5.

The system gives the surgeon the confidence of knowing that the robot will never let him approach the patient in a non-desired area. However, while the surgeon does not attempt to cross the shield, the robot can be moved freely.



Fig. 5. Detail of a virtual shield and the scenario including the robot arm.

The library developed is useful not only to avoid collisions but also to navigate over an object's surface. This surface navigation allows the surgeon to feel smooth movements of the robot when the tool is in contact with the virtual fixtures. The navigation algorithm helps the surgeon not only avoiding the forbidden regions defined in the pre-operative phase but also guiding him to the desired cutting path.

The navigation algorithm modifies the position of the robot tool, but not its orientation. The algorithm is based on three steps: Knowing the new desired destination of the robot tool, the first step consists of detecting all collisions between the tool and the object's surface. When all collisions are detected, the second step consists of projecting the desired point to the plane of the collision triangle, Fig. 6.a. Finally the projected point that is closer to the desired one is selected. A problem can occur when the projected point falls outside the triangle region. In this situation it is not possible to ensure that this new projected point is always outside the object Fig. 6.c. In this case the new point is projected to the perimeter of the triangle. To accomplish this, the outside region of the triangle is divided into six new regions ($R_1, R_{12}, R_2, R_{23}, R_3, R_{31}$), which are defined by the normals of the edges of the triangle applied to the vertices of the triangle. The point is then projected to the closest point of the triangle border of its region Fig. 6.b. Now, the new destination point is collision free Fig. 6.d.

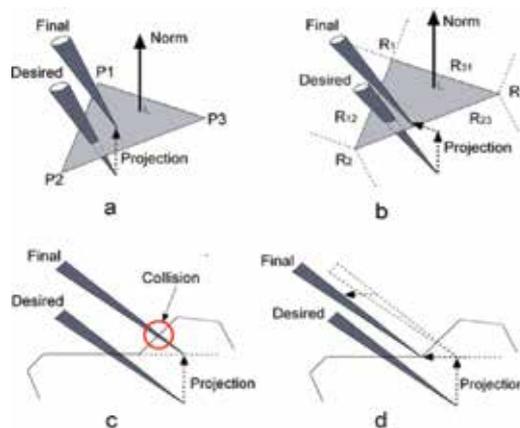


Fig. 6. Different contact situations between the tool and a triangle or set of triangles

4.2 Virtual robot

The concept of the Virtual Robot

In this section, an application that makes an intensive use of the proximity queries package RPQ is presented. During the last years, robotic applications are expected to solve new and more complex tasks. These greater demands requiring more dexterity and accessibility to larger and more complex working areas may imply that a single robot is not enough to perform the desired task. Let's consider as an example of application the need to remotely manipulate an object in front of a camera, for its inspection, in a complex workspace by means of teleoperation. The visualization of the object from all the desired points of view may require a great accessibility and manoeuvring. This manipulation may present some problems like singularities, joint limits, obstacles, or even occlusions produced by the own robotic arm. To deal with such applications the proposed solution is a virtual robot, constituted by a combination of a given number of robots acting as only one. These robots transfer the inspected object from one to another only when necessary, allowing the user to execute the desired task. To follow a continuous trajectory, at least one robotic arm must hold the inspected object at a time. When the user moves the inspected object at will, the robot that holds the object can fall into a singularity or collide with an obstacle. If this happens, the task cannot be accomplished. To prevent this situation, another robot should grasp the object and continue the trajectory before the previous one reaches the critical situation.

The virtual robot is expected to act as if there was only one physical robot, operating in such a way that its internal mode of cooperative working is transparent to the user. To accomplish this goal, the virtual robot has to automatically decide which real robot, or robots, is operative at each successive step during the execution of a task. The virtual robot also has to decide the best configuration for each real robot in order to facilitate the task transfer if necessary and, what is even more important, avoid collisions. In summary, the virtual robot is a set of real robot arms automatically controlled acting as if they were only one, letting the user to concentrate its efforts in the task.

Obviously, two or more cooperative robots can increase the range of tasks that a single robot can execute. In the other hand, the risk of collision between them and with other possible obstacles in the workspace increases respect to work with a single robot.

Collision avoidance in Cooperative Robots

When two or more robots share the workspace, the collision avoidance is done depending on if the trajectory is planned or not. If the trajectory is defined a priori, the path planning algorithm must deal with the collision problem. There is a wide range of different solutions in the literature. In (LaValle, 2006) a detailed study of the most important path planning algorithms is done. In those cases, the time spend on detect collisions is important, but not critical. In the other hand, when the trajectory is not predefined, the collision detection must be done during the execution time. In this case, the computational performance of the collision detection algorithm is critical. In summary, in both cases, but even more in the second one, a fast proximity queries package is necessary.

Most of the proposed solutions in real time collision detection use reactive behavior algorithms. In (J. TaeSeok et al., 2004) a virtual impedance method using a virtual mass-spring-damper model is proposed to avoid collisions in a teleoperated mobile robot. This method is also used as virtual guidance to the robot to achieve the desired end point of the trajectory. This virtual model is applied between the obstacles and the robot and its proportional to the distance between them and the approximation velocity. This principle is

easy to extend to robotic arms working in a six degrees of freedom workspace. In Fig. 7.a an example of the virtual mass-spring-damper model applied to a couple of robots is shown. In this example two virtual restrictions are applied: one between the end effectors of the robots and the second one between the robotic arms. Another technique to avoid collisions and similar to the concept explained above is based on applying a virtual repulsion force field. For instance, in (Jih-Gau Juang, 2004) a virtual repulsion force is proposed to avoid collisions. In this work, the force is proportional to the distance between the robot and the obstacle. This virtual force field can be applied to the whole robot arm, as shown in Fig. 7.b. All of those methods require the knowledge of the geometrical model of the robotic arms and the objects in the workspace. They also need sensorial information of the workspace and the configuration of the robots at each instant of time. Using this information and with a real time proximity query package the collision avoidance algorithms can work properly.

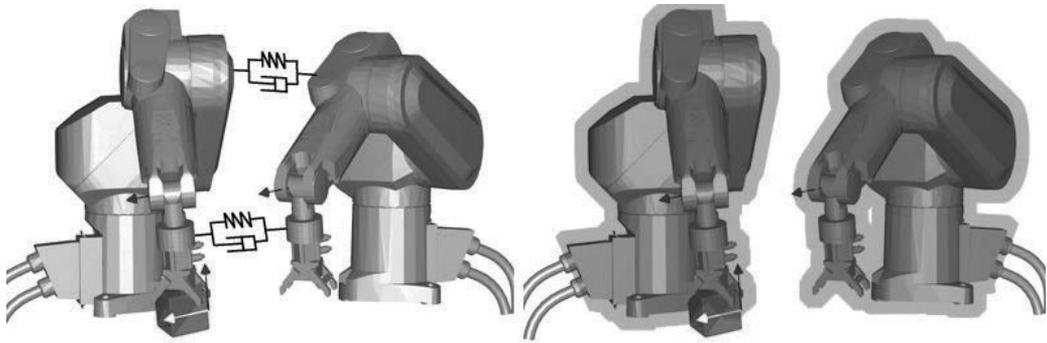


Fig. 7.a. Two virtual mass-spring-damper model are applied to two robotic arms

Fig. 7.b. A repulsive force field, illustrated as an aureole recovering each arm, is applied to the robots in order to avoid collisions.

Applied Algorithm to prevent collisions in Virtual Robot.

The virtual robot is an excellent example of the need and usage of a fast and accurate proximity query package like RPQ. The first reason is, as explained above, that the virtual robot is composed by a set of real robotic arms sharing their workspace, so it implies high risk of collision. Second reason is the absence of path planning, which implies that the collisions must be detected in real time.

Another important reason to use RPQ is because the virtual robot tries to alter the user's desired trajectory as less as possible. To accomplish it, the robot arm that is executing the task does not modify its trajectory to avoid collisions. The rest of the robots must change, if necessary, their configuration to evade collisions and also to facilitate the task transfer. In order to prevent collisions between the robotic arms, at each step of the task the minimum distance between the robot that executes the task and the rest of the robots is calculated. If this distance is lower than a boundary distance, a repulsion force is applied to avoid collision. The boundary distance is defined by the difference between the approximation velocity between the robots and the maximum acceleration and velocity allowed to each robotic arm in the direction of the minimum distance vector. This method implies to calculate at each step of the trajectory the minimum distance vector between the robotic arms. Finally, to improve the time to make the task transfer, the manipulated object has an

attraction virtual force which is applied to the end effector of the robots that does not execute the task. With this force the end effector are near the object. In order to ensure the safety, the attractive force is not applied if it is in conflict with the repulsive force. The combined use of repulsion force, F_{rep} , around the robotic arm and the attraction force, F_{atr} , around the manipulated object is shown in Fig. 8.

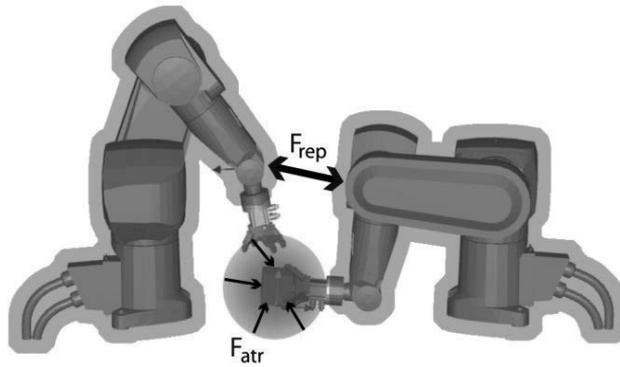


Fig. 8. Application of virtual forces in a robotic cooperative task

5. Conclusion

This paper presents the development of RPQ, a proximity queries library optimized for applications where robots share a common workspace and interact with objects. Due to the amount of collision detection packages, RPQ is built above a generic collision package, PQP. It is the generic collision package that better fits RPQ purposes.

The goal of developing RPQ was to fill an existing gap in computer tools for robotic applications, where robots interact with objects in their environment. Three optimizations have been performed to a generic collision library: working with different resolution levels of representation, the use of a weighted matrix for choosing the best order for collision checking and the definition of a binary matrix that determines the possibility of collision between objects. RQP has been validated in different applications such as multirobot collision avoidance, virtual robot controller and surface navigation.

As expected, optimizations improve the time performance of the system, although this improvement is highly application dependent.

The introduction of different levels of resolution in the geometric models of the objects and robots generally decreases the computational time for collision checking. The use of bounding boxes decreases drastically the number of high resolution queries needed. This is a really important point taking into account that they are much more time consuming. There are cases where low resolution queries do not solve the whole collision query. This increases the computation time. However, choosing a suitable order for checking collisions helps finding them in a quicker manner. The precomputation of impossibilities of collision between different objects (Collision Matrix) increases the performance of the system in case of having objects with restricted mobility in the workspace.

The combined use of optimizations generates good results in workspaces shared by at least two robots and objects.

RPQ has a wide range of applicability. RQP library is not only useful for proximity queries but has also proved to be a good tool for surface navigation and virtual representations, due to its ability to introduce virtual objects in the shared workspace. The virtual fixtures developed in the paper are examples of how RPQ can be used to modify robot's behaviour. As proved in the applications presented, RPQ is not only useful for developers of robotic applications, but also for users of robotic applications, i.e. surgeons that require new robotic tools for improving surgical procedures.

6. References

- Bergen, G. V. D. 2002. Solid collision detection library users guide.
- B.Geiger (2000). Real-time collision detection and response for complex environments. *International Conference on Computer Graphics*. IEEE Computer Society.
- Giralt, X. and Hernansanz, A. (2006). Optimization of proximity queries in robotic environments. *AVR - 2es Jornades UPC de Recerca en Automtica, Visio I Robotica* (in catalan).
- Kim, Y., Lin, M., and Manocha, D. (2002). Deep: Dualspace expansion for estimating penetration depth between convex polytopes. *International Conference on Robotics and Automation*. IEEE.
- S.Ehmann and Lin, M. (2000). Accelerated proximity queries between convex polyhedra by multilevel voronoi marching. *Technical report, Department of Computer Science, University of North Carolina*.
- S.Ehmann and Lin, M. (2001). Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Eurographics*, volume 3.
- Stanisic, Z., Jackson, E., and Payandeh, S. (1996). Virtual fixtures as an aid for teleoperation. *9th Canadian Aeronautics and Space Institute Conference*.
- UNC. (1999). Pqp - a proximity query package by research group on modeling, physically-based simulation and applications.
- La Valle. (2006). *Planning Algorithms Cambridge University Press*.
- J. TaeSeok, L. JangMyung, and H. Hashimoto. (2004). Internet-based obstacle avoidance of mobile robot using a force-reflection. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4 ed 2004, pp. 3418-3423.
- Jih-Gau Juang. (2004). Repulsive force control based on distance computation algorithm. *Computers and Industrial Engineering*, vol. 47, no. 1, pp. 45-60, Aug.2004.

Takagi-Sugeno Fuzzy Observer for a Switching Bioprocess: Sector Nonlinearity Approach

Enrique J. Herrera-López¹, Bernardino Castillo-Toledo²,
Jesús Ramírez-Córdova¹ and Eugénio C. Ferreira³

¹*Centro de Investigación y Asistencia en Tecnología y Diseño del Estado de Jalisco*

²*Centro de Investigación y de Estudios Avanzados del I.P.N, Unidad Guadalajara*

³*Institute for Biotechnology and Bioengineering, Universidade do Minho, Braga*

^{1,2}México, ³Portugal

1. Introduction

In a bioprocess it is desired to produce high amounts of biomass or metabolites such as vitamins, antibiotics, and ethanol, among others. The measurement of biological parameters as the cell, by-product concentrations and the specific growth rate are essential to the successful monitoring and control of bioprocesses (Horiuchi & Kishimoto, 1998). Adequate control of the fermentation process reduces production costs and increases the yield while at the same time achieve the quality of the desired product (Yamuna & Ramachandra, 1999). Nevertheless, the lack of cheap and reliable sensors providing online measurements of the biological state variables has hampered the application of automatic control to bioprocesses (Bastin & Dochain, 1990). This situation encourages the searching of new software sensors in bioprocesses.

A state observer is used to reconstruct, at least partially the state variables of the process. Two classes of state observers or software sensors for (bio)chemical processes can be found in the literature (Dochain, 2003). A first class of observers called asymptotic observers, is based on the idea that the uncertainty in bioprocess models is located in the process kinetics models. A second class is based on the perfect knowledge of the model structure (Luenberger, Kalman observers and nonlinear observers). Different applications of state observers in bioprocess are reported in the literature (Cazzador & Lubenova, 1995; Farza et al., 2000; Guay & Zhang, 2002; Lubenova et al., 2003; Oliveira et al., 2002; Soh & Cao, 1999; Veloso et al., 2007).

Fuzzy logic has become popular in the recent years, due to the fact that it is possible to add human expertise to the process. Nevertheless, in the case where the nonlinear model and all the parameters of a process are known, a fuzzy system may be used. A first approach can be done using the Takagi-Sugeno fuzzy model (Takagi & Sugeno, 1985), where the consequent part of the fuzzy rule is replaced by linear systems. This can be attained, for example, by linearizing the model around operational points, getting local linear representation of the nonlinear system.

Another way to obtain a model can be achieved using the method of sector nonlinearities, which allows the construction of an exact fuzzy model from the original nonlinear system by means of linear subsystems (Tanaka & Wang, 2001). From this exact model, fuzzy state observers and fuzzy controllers may be designed based on the linear subsystems. Different fuzzy logic applications to bioprocesses can be found in the scientific literature (Genovesi et al., 1999, Ascencio et al., 2004; Karakazu et al., 2006). In this chapter a Takagi-Sugeno fuzzy observer based on sector nonlinearities is proposed and applied to a continuous nonlinear baker’s yeast fermentation process. The observer gains are calculated using linear matrix inequalities. An interesting feature of this model is that it can be divided in two models: a respiro-fermentative (RF) model with ethanol production and a respirative (R) model with ethanol consumption. The model can switch to the RF -R- RF model depending on whether the yeasts are producing or consuming ethanol.

2. Fuzzy Systems Preliminaries

A nonlinear system may be represented by linear subsystems called Takagi-Sugeno, (figure 1). The Takagi-Sugeno fuzzy models are used to represent nonlinear dynamics by means of a set of IF-THEN rules. The consequent parts of the rules are local linear systems obtained from specific information about the original nonlinear plant.

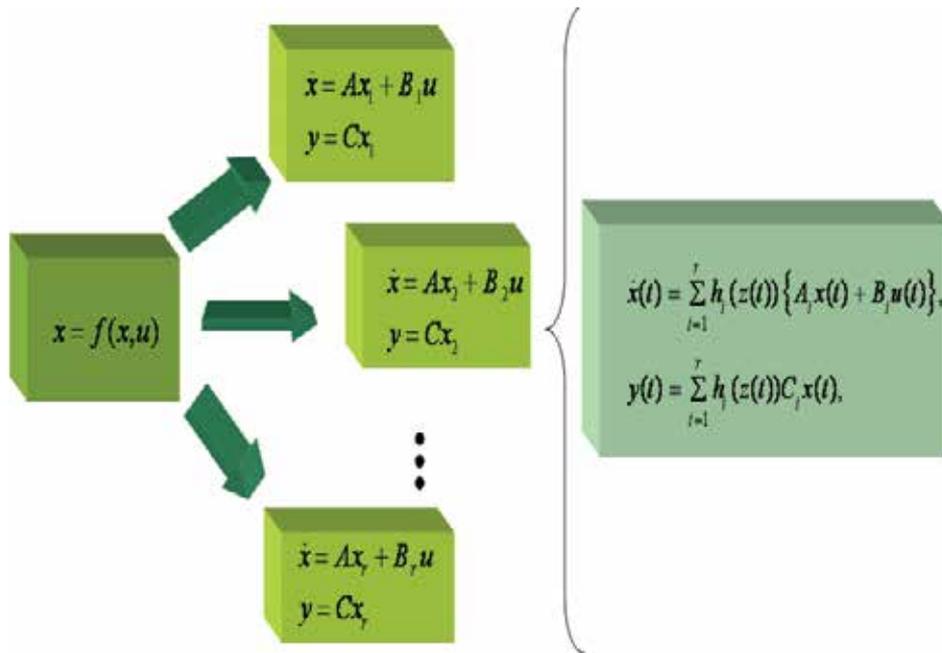


Fig. 1 Takagi-Sugeno representation for a nonlinear system

The *i*th rule of a continuous fuzzy model has the following form:

Model Rule *i*:

If $z_1(t)$ is φ_{i1} and ... and $z_p(t)$ is φ_{ip} .

$$\text{THEN } \begin{cases} \dot{x}(t) = A_i x(t) + B_i u(t) \\ y(t) = C_i x(t) \end{cases} \quad i = 1, 2, 3, \dots, r, \quad (1)$$

where φ_{ij} is a fuzzy set and r is the number of rules in the fuzzy model; $x(t) \in \mathcal{Y}^n$ is the state vector, $u(t) \in \mathcal{U}^m$ is the input vector, $y(t) \in \mathcal{Y}^p$ is the output vector, $A_i \in \mathcal{Y}^{n \times n}$, $B_i \in \mathcal{Y}^{n \times m}$, and $C_i \in \mathcal{Y}^{p \times n}$ are suitable matrices, and $z(t) = [z_1(t), \dots, z_p(t)]$ is a known vector of premise variables which may coincide or partially depend on the state $x(t)$.

Given a pair of $(x(t), u(t))$ and using a singleton fuzzifier, product inference and center of gravity defuzzifier, the aggregated Takagi-Sugeno fuzzy model can be inferred as:

$$\begin{aligned} \dot{x}(t) &= \sum_{i=1}^r h_i(z(t)) \{A_i x(t) + B_i u(t)\}, \\ y(t) &= \sum_{i=1}^r h_i(z(t)) C_i x(t), \end{aligned} \quad (2)$$

where

$$h_i(z(t)) = \frac{\prod_{j=1}^p \varphi_{ij}(z_j(t))}{\sum_{i=1}^r \prod_{j=1}^p \varphi_{ij}(z_j(t))} \quad (3)$$

for all t . The term $\varphi_{ij}(z_j(t))$ is the membership value of $z_j(t)$ in φ_{ij} . Since

$$\begin{aligned} \prod_{j=1}^p \varphi_{ij}(z_j(t)) \geq 0 \quad \text{and} \quad \sum_{i=1}^r \prod_{j=1}^p \varphi_{ij}(z_j(t)) > 0, \quad i = 1, \dots, r, \\ h_i(z(t)) \geq 0 \quad \text{and} \quad \sum_{i=1}^r h_i(z(t)) = 1, \quad i = 1, \dots, r, \end{aligned} \quad (4)$$

for all t .

2.1 Sector Nonlinearity

A nonlinear system may also be represented by sectors (Tanaka & Wang, 2001). Consider a nonlinear system given by $\dot{x}(t) = f(x(t))$ where $f(0) = 0$. A global sector is found when $\dot{x}(t) = f(x(t)) \in [s_1 \ s_2]x(t)$, where $s_1 x(t)$ and $s_2 x(t)$ are lines as shown in figure 2. A global sector guarantees an exact fuzzy representation for the nonlinear model. Some times it is difficult to find global sectors, in that case it is possible to find a local sector bounded by the region $-a < x(t) < a$, as shown in figure 3.

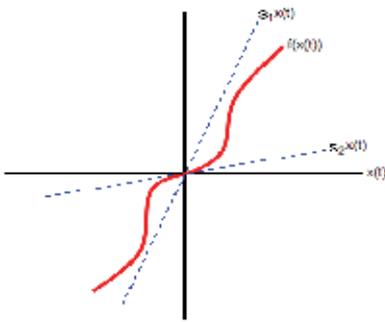


Fig. 2 Global sector

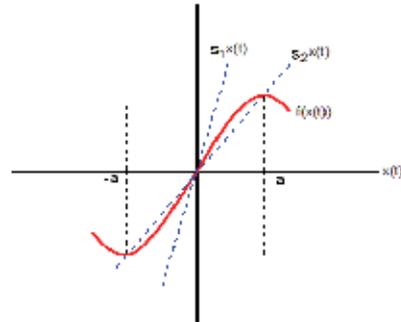


Fig. 3 Local sector

2.2 Fuzzy Observer

The state of a system is not always fully available, so it is necessary to use an observer to reconstruct, at least partially the states variables of the process. This requires to satisfy the condition

$$\lim_{t \rightarrow 0} (x(t) - \hat{x}(t)) = 0 \tag{5}$$

where $\hat{x}(t)$ denotes the state vector estimated by the fuzzy observer. There are two cases for fuzzy observers design depending on whether or not $z(t)$ depends on the state variables estimated by a fuzzy observer (Tanaka & Wang, 2001). Given the Takagi-Sugeno fuzzy model (1), the i th rule of a continuous fuzzy observer can be constructed as:

Observer Rule i

If $z_1(t)$ is φ_{i1} and ... and $z_p(t)$ is φ_{ip} .

$$\text{THEN} \begin{cases} \hat{\dot{x}} = \sum_{i=1}^r h_i(z(t)) \{ A_i \hat{x}(t) + B_i u(t) + K_i (y(t) - \hat{y}(t)) \}, \\ \hat{y}(t) = h_i(z(t)) C_i \hat{x}(t). \end{cases} \tag{6}$$

where K_i is the observer gain and $\hat{y}(t)$ is the fuzzy observer output for the i th subsystem. If $z(t)$ depends on the estimated state variables, the observer consequent part takes the following form:

$$\text{THEN} \begin{cases} \hat{\dot{x}} = \sum_{i=1}^r h_i(\hat{z}(t)) \{ A_i \hat{x}(t) + B_i u(t) + K_i (y(t) - \hat{y}(t)) \}, \\ \hat{y}(t) = h_i(\hat{z}(t)) C_i \hat{x}(t). \end{cases} \tag{7}$$

It is possible to calculate the observer gains from the solution X, N_i of the following inequalities (Tanaka & Wang, 2001).

$$\begin{aligned}
& X > 0 \\
& -A_i^T X - XA_i + C_i^T N_i^T + N_i C_i - 2\alpha X > 0, \\
& -A_i^T X - XA_i - A_j^T X - XA_j + C_j^T N_i^T + N_i C_j + C_i^T N_j^T + N_j C_i - 4\alpha X \geq 0 \\
& \qquad \qquad \qquad i < j \text{ s.t. } h_i \cap h_j \neq \emptyset
\end{aligned} \tag{8}$$

where A_i is the state matrix and C_i is the output matrix. The decay rate (α) is related with the observer speed response. The inequalities (8) can be converted to linear matrix inequalities by means of Shur's complement (Braatz & VanAntwerp, 2000). The condition $i < j$ s.t. $h_i \cap h_j \neq \emptyset$ means that inequalities (8) holds for all $i < j$ excepting $h_i z(t) \cdot h_j z(t) = 0$ for all $z(t)$. The observer gain K_i and the common positive definite matrix P can be obtained by means of

$$P = X^{-1}, \quad K_i = X^{-1} N_i. \tag{9}$$

The fermentative mathematical model will now be described.

3. Fermentation Mathematical Model

The *Saccharomyces cerevisiae* yeast may grow on glucose following three metabolic pathways (Sonnleitner & Käpelli, 1986).

1.- *Oxidative growth on glucose*, in presence of oxygen (O_2) the glucose (S) is consumed to produce biomass (X) and carbon dioxide (CO_2).



2.- *Fermentative growth on glucose*, in absence of oxygen the substrate is used to produce biomass, carbon dioxide and mainly ethanol (E).



3.- *Oxidative growth on Ethanol*, the ethanol produced by the fermentative pathway may be consumed in presence of oxygen producing biomass and carbon dioxide.



3.1 The Respiro-Fermentative and Respirative Fermentation Models

A continuous baker's yeast culture can be represented by the following set of differential equations

$$\begin{aligned}
\dot{x}_1 &= (\mu_s^o + \mu_s^r + \mu_e^o)x_1 - Dx_1 \\
\dot{x}_2 &= (-k_1\mu_s^o - k_2\mu_s^r)x_1 - Dx_2 + DS_{in} \\
\dot{x}_3 &= (k_3\mu_s^r - k_4\mu_e^o)x_1 - Dx_3 \\
\dot{x}_4 &= (-k_5\mu_s^o - k_6\mu_e^o)x_1 - Dx_4 + OTR
\end{aligned} \tag{13}$$

where the variables of model (13) are shown in table 1.

Variables	Units
x_1 = Biomass concentration	g/l
x_2 = Glucose concentration	g/l
x_3 = Ethanol concentration	g/l
x_4 = Dissolved oxygen concentration	mg/l
D = Dilution rate	1/h
S_{in} = Substrate concentration feed	g/l
$OTR = K_L a (C^{sat} - x_4)$ = Oxygen transfer rate	mg/lh
μ_s^o = Specific growth rate (oxidative growth on glucose)	1/h
μ_s^r = Specific growth rate (fermentative growth on glucose)	1/h
μ_e^o = Specific growth rate (oxidative growth on ethanol)	1/h
$k_1, k_2, k_3, k_4, k_5, k_6$ = yield coefficients	

Table 1. Variables used in the baker's yeast model (13)

The oxygen transfer rate is given by $OTR = K_L a (C^{sat} - x_4)$ which may be split in two terms, one that is constant and another one that depend on the dissolved oxygen.

$$-K_L a x_4 \tag{14}$$

$$K_L a C^{sat}, \tag{15}$$

Pormelean (1990) suggested a reformulation of model (13) using two partial models: a respiro-fermentative partial model (*RF*) with ethanol production and a respirative partial model (*R*) with ethanol consumption. With this reformulation a split process model is obtained, switching from the *RF* partial model to the *R* partial model and vice versa depending on whether the system is consuming or producing ethanol. To precise these ideas, consider a nonlinear system described by model (16-17), which can be written as

$$\dot{x}(t) = f_i(x(t)) + Bu(t) + d, \quad i = 1, 2 \tag{16}$$

$$y(t) = h(x(t)), \tag{17}$$

where $f_i(x(t))$ describe both the RF and R partial models, namely

$$f_1 = \begin{bmatrix} (\mu_{s_RF}^o + \mu_{s_RF}^r)x_1 - Dx_1 \\ (-k_1\mu_{s_RF}^o - k_2\mu_{s_RF}^r)x_1 - Dx_2 \\ k_3\mu_{s_RF}^r x_1 - Dx_3 \\ -k_5\mu_{s_RF}^o x_1 - Dx_4 - K_L ax_4 \end{bmatrix} := f_{RF} \quad (18)$$

and for the R model

$$f_2 = \begin{bmatrix} (\mu_{s_R}^o + \mu_{e_R}^o)x_1 - Dx_1 \\ -k_1\mu_{s_R}^o x_1 - Dx_2 \\ -k_4\mu_{e_R}^o x_1 - Dx_3 \\ (-k_5\mu_{s_R}^o - k_6\mu_{e_R}^o)x_1 - Dx_4 - K_L ax_4 \end{bmatrix} := f_R, \quad (19)$$

The input vector and the manipulated variable are given by

$$B = [0, D, 0, 0]^T, \quad (20)$$

$$u(t) = S_{in} \quad (21)$$

where T is the input vector transpose. As already said, OTR rate was divided in two terms, the first one $-K_L ax_4$ (14) was included in f_{RF} (18) and f_R matrices (19); the second term $K_L a C^{sat}$ (15), is taken as a known and constant perturbation (d) given by

$$d = [0, 0, 0, K_L a C^{sat}]^T \quad (22)$$

In the RF partial model, the metabolic pathways oxidative growth on glucose (10) and fermentative growth on glucose (11) are present, therefore ethanol is produced. The specific growth rates for the RF partial model are given by:

$$\begin{aligned} \mu_{s_RF}^o &= Y_{O_2} \left(q_o^{\max} \frac{x_4}{K_o + x_4} \right) \\ \mu_{s_RF}^r &= Y_r \left(q_s^{\max} \frac{x_2}{K_s + x_2} - q_c^{\max} \frac{x_4}{K_o + x_4} \frac{Y_{O_2}}{Y_o} \right) \end{aligned} \quad (23)$$

In the R partial model, the pathway oxidative growth on glucose (10) is also present; however, the specific growth rate is now given by:

$$\mu_{s_R}^o = Y_o \left(q_s^{\max} \frac{x_2}{K_s + x_2} \right) \quad (24)$$

The oxidative growth on ethanol (12) for the *R* partial model depends on,

$$\mu_{e_R}^o = \begin{cases} q_{e_1} & \text{if } q_{e_1} < q_{e_2} \\ q_{e_2} & \text{if } q_{e_1} \geq q_{e_2} \end{cases} \quad (25)$$

where

$$q_{e_1} = Y_e q_e^{\max} \frac{x_3}{K_e + x_3} \frac{K_i}{K_i + x_2} \quad (26)$$

$$q_{e_2} = Y_{O_2} e \left(q_o^{\max} \frac{x_4}{K_o + x_4} - \frac{Y_o}{Y_{O_2}} \left(q_s^{\max} \frac{x_2}{K_s + x_2} \right) \right) \quad (27)$$

The *RF* and *R* models cannot be enabled at the same time. A condition for the transition between the *RF*-*R*-*RF* partial models is given by (Ferreira, 1995)

$$\begin{aligned} RF \rightarrow R & \text{ if } \mu_s^r \leq 0 \\ R \rightarrow RF & \text{ if } \mu_e^o \leq 0 \end{aligned} \quad (28)$$

The parameters values and the initial conditions for the *RF* and *R* partial models are given in table 2; a complete description of all parameters can be found in (Ferreira, 1995). The manipulated variable $u(t)=S_{in}$ was set as a square signal as can be seen in figure 4.

Parameter	Value	Parameter	Value
q_s^{\max}	3.5 gS/gX h	k_1^{-1}	0.49 gX/gS
q_e^{\max}	0.236 gE/gX h	k_2^{-1}	0.05 gX/gS
q_o^{\max}	0.256 gO ₂ /g X h	k_3^{-1}	0.1 gX/gE
K_i	0.1 g/l	k_4^{-1}	0.72 gX/gE
K_o	0.0001 g/l	k_5^{-1}	1.2 gX/gO ₂
C^{sat}	7.0 mg/l	k_6^{-1}	0.64 gX/gO ₂
S_{in}	10 g/l	$x_1(0)$	0.1 g/l
K_e	0.1 g/l	$x_2(0)$	0.02 g/l
K_s	0.2 g/l	$x_3(0)$	0.15 g/l

Parameter	Value	Parameter	Value
$K_L a$	100 1/h	$x_4(0)$	0.0066 mg/l
D	0.2 1/h		

Table 2. Parameters values used in model (13)

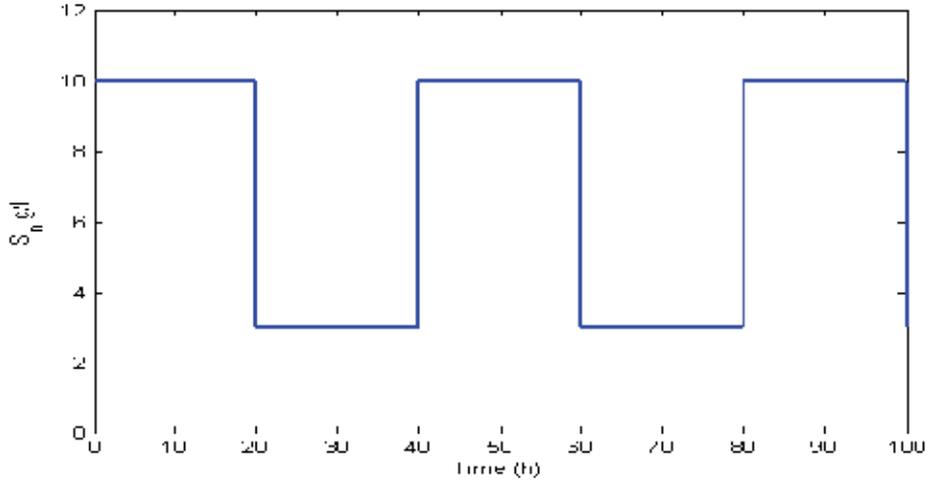


Fig. 4 Input square signal for the baker's yeast model

4. The Takagi-Sugeno Fuzzy Exact Model

When the nonlinear dynamic model for the baker's yeast is known, as well as all their parameters, a fuzzy exact model can be derived from the given nonlinear model. This requires a sector nonlinearity approach (Tanaka & Wang, 2001).

4.1 The Respiro-Fermentative Fuzzy Exact Model

To construct the *RF* exact fuzzy model we need to express the *RF* partial model as a nonlinear system (16-17).

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} \mu_{s_RF}^o + \mu_{s_RF}^r - D & 0 & 0 & 0 \\ -k_1 \mu_{s_RF}^o - k_2 \mu_{s_RF}^r & -D & 0 & 0 \\ k_3 \mu_{s_RF}^r & 0 & -D & 0 \\ -k_5 \mu_{s_RF}^o & 0 & 0 & -D - K_L a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ D \\ 0 \\ 0 \end{bmatrix} S_{in} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} K_L a C^{sat} \quad (29)$$

$$y(t) = h(x_3(t)) + h(x_4(t)) \quad (30)$$

Substituting the specific rates (23) in the $f_{RF}(x(t))$ matrix from model (29), we obtain the matrix given by (31), for convenience called scheme f_{RF-I}

scheme f_{RF_I}

$$f_{RF_I} = \begin{bmatrix} \left(Y_{O_2} q_c^{\max} - Y_{r q_c}^{\max} \frac{Y_{O_2}}{Y_o} \right) \frac{x_4}{K_o + x_4} + Y_{r q_s}^{\max} \frac{x_2}{K_s + x_2} - D & 0 & 0 & 0 \\ \left(-k_1 Y_{O_2} q_c^{\max} + k_2 Y_{r q_c}^{\max} \frac{Y_{O_2}}{Y_o} \right) \frac{x_4}{K_o + x_4} - k_2 Y_{r q_s}^{\max} \frac{x_2}{K_s + x_2} - D & 0 & 0 & 0 \\ -k_3 Y_{r q_c}^{\max} \frac{Y_{O_2}}{Y_o} \frac{x_4}{K_o + x_4} + k_3 Y_{r q_s}^{\max} \frac{x_2}{K_s + x_2} & 0 & -D & 0 \\ -k_5 Y_{O_2} q_c^{\max} \frac{x_4}{K_o + x_4} & 0 & 0 & -D - K_L a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad (31)$$

However; the matrix $f_{RF}(x(t))$ may also be written as:

Scheme f_{RF_II}

$$f_{RF_II} = \begin{bmatrix} \left(Y_{O_2} q_c^{\max} - Y_{r q_c}^{\max} \frac{Y_{O_2}}{Y_o} \right) \frac{x_4}{K_o + x_4} - D & Y_{r q_s}^{\max} \frac{x_1}{K_s + x_2} & 0 & 0 \\ \left(-k_1 Y_{O_2} q_c^{\max} + k_2 Y_{r q_c}^{\max} \frac{Y_{O_2}}{Y_o} \right) \frac{x_4}{K_o + x_4} - k_2 Y_{r q_s}^{\max} \frac{x_1}{K_s + x_2} - D & 0 & 0 & 0 \\ -k_3 Y_{r q_c}^{\max} \frac{Y_{O_2}}{Y_o} \frac{x_4}{K_o + x_4} & k_3 Y_{r q_s}^{\max} \frac{x_1}{K_s + x_2} & -D & 0 \\ -k_5 Y_{O_2} q_c^{\max} \frac{x_4}{K_o + x_4} & 0 & 0 & -D - K_L a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad (32)$$

or as

Scheme f_{RF_III}

$$f_{RF_III} = \begin{bmatrix} Y_{O_2} q_c^{\max} \frac{x_4}{K_o + x_4} - D & Y_{r q_s}^{\max} \frac{x_1}{K_s + x_2} & 0 & -Y_{r q_c}^{\max} \frac{Y_{O_2}}{Y_o} \frac{x_1}{K_o + x_4} \\ -k_1 Y_{O_2} q_c^{\max} \frac{x_4}{K_o + x_4} - k_2 Y_{r q_s}^{\max} \frac{x_1}{K_s + x_2} - D & 0 & k_2 Y_{r q_c}^{\max} \frac{Y_{O_2}}{Y_o} \frac{x_1}{K_o + x_4} & 0 \\ 0 & k_3 Y_{r q_s}^{\max} \frac{x_1}{K_s + x_2} & -D & -k_3 Y_{r q_c}^{\max} \frac{Y_{O_2}}{Y_o} \frac{x_1}{K_o + x_4} \\ -k_5 Y_{O_2} q_c^{\max} \frac{x_4}{K_o + x_4} & 0 & 0 & -D - K_L a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad (33)$$

Although there are another possible combinations to write the $f_{RF}(x(t))$ matrix, with these approaches we obtain enough information to precise our point. From scheme f_{RF_I} (31) two nonlinearities can be detected

$$NL_1 = \frac{x_4}{K_o + x_4}; \quad NL_2 = \frac{x_2}{K_s + x_2} \quad (34)$$

from scheme f_{RF_II} (32) also two different nonlinearities can be observed

$$NL_1 = \frac{x_4}{K_o + x_4}; \quad NL_3 = \frac{x_1}{K_s + x_2} \quad (35)$$

and from scheme f_{RF_III} (33) three nonlinearities are present

$$NL_1 = \frac{x_4}{K_o + x_4}; \quad NL_3 = \frac{x_1}{K_s + x_2}; \quad NL_4 = \frac{x_1}{K_o + x_4} \quad (36)$$

Although with each matrix given by (31-33) the exact fuzzy model can be built, it will take four linear subsystems (2²) for the scheme f_{RF_I} and scheme f_{RF_II} and eight linear subsystems (2³) for scheme f_{RF_III} . For convenience the nonlinearities (35) and the scheme f_{RF_II} (32) are chosen to build the RF exact fuzzy model, the reason will be evident in the next section where the fuzzy observer is constructed. The premise variable $z_1^{RF}(t)$ is defined as

$$NL_1 = z_1^{RF}(t) = \frac{x_4}{K_o + x_4}; \quad \text{for } x_4 \neq -K_o \quad (37)$$

From equation (37) the maximum and minimum values of $z_1^{RF}(t)$ can be obtained. In figure 5 the plot of (37) can be observed.

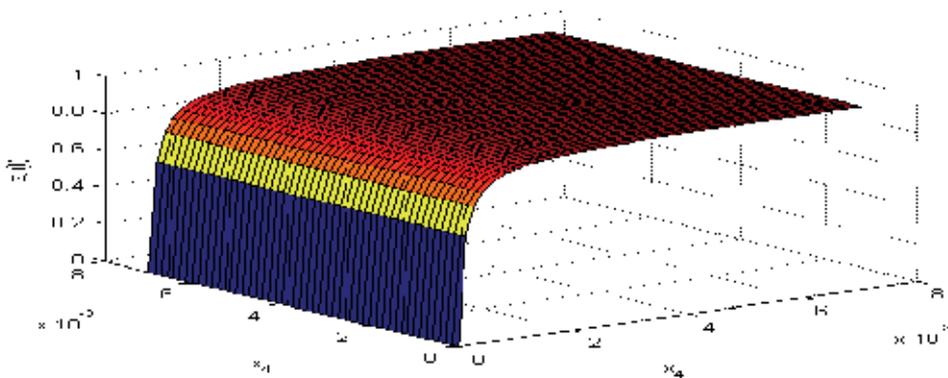


Fig. 5 Plot for the premise variable $z_1^{RF}(t)$

The maximum and minimum values of $z_1^{RF}(t)$ in the range $x_4(t) \in [0, 0.007]$ are given by

$$\max_{x_4(t)} z_1(t) = 0.9859 = a_1 \quad \min_{x_4(t)} z_1(t) = 0 = a_2 \quad (38)$$

we define the premise variable $z_2^{RF}(t)$ as:

$$NL_3 = z_2^{RF}(t) = \frac{x_1}{K_s + x_2}; \quad \text{for } x_2 \neq -K_s \quad (39)$$

From equation (39) the maximum and minimum values of $z_2^{RF}(t)$ can be obtained, as shown in figure 6,

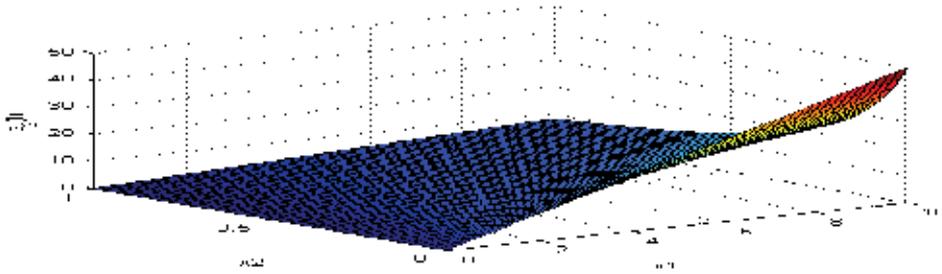


Fig. 6 Plot for the premise variable $z_2^{RF}(t)$

The maximum and minimum values of $z_2^{RF}(t)$ in the range $x_1(t) \in [0, 10]$ and $x_2(t) \in [0, 1]$ are given by

$$\max_{x_1(t)x_2(t)} z_2(t) = 50 = b_1 \quad \min_{x_1(t)x_2(t)} z_2(t) = 0 = b_2 \quad (40)$$

The membership functions are built from these equations

$$\begin{aligned} z_1^{RF}(t) &= \sum_{i=1}^2 \varphi_{1i}(z_1^{RF}(t)) a_i; \\ z_2^{RF}(t) &= \sum_{j=1}^2 \varphi_{2j}(z_2^{RF}(t)) b_j \end{aligned} \quad (41)$$

where the following properties must hold

$$\begin{aligned}\varphi_{11}(z_1^{RF}(t)) + \varphi_{12}(z_1^{RF}(t)) &= 1 \\ \varphi_{21}(z_2^{RF}(t)) + \varphi_{22}(z_2^{RF}(t)) &= 1\end{aligned}\quad (42)$$

Solving equations (41 and 42) the following membership functions are obtained

$$\begin{aligned}\varphi_{11}(z_1(t)) &= \frac{z_1(t) - a_2}{a_1 - a_2} & \varphi_{12}(z_1(t)) &= \frac{-z_1(t) + a_1}{a_1 - a_2} \\ \varphi_{21}(z_2(t)) &= \frac{z_2(t) - b_2}{b_1 - b_2} & \varphi_{22}(z_2(t)) &= \frac{-z_2(t) + b_1}{b_1 - b_2}\end{aligned}\quad (43)$$

Substituting the maximum and minimum values a_1 , a_2 , b_1 and b_2 in (32) we obtain 4 possible combinations to express the linear subsystems.

$$A_{ij}^{RF} = \begin{bmatrix} \left(Y_{O_2} q_c^{\max} - Y_r q_c^{\max} \frac{Y_{O_2}}{Y_o} \right) a_i - D & Y_r q_s^{\max} b_j & 0 & 0 \\ \left(-k_1 Y_{O_2} q_c^{\max} + k_2 Y_r q_c^{\max} \frac{Y_{O_2}}{Y_o} \right) a_i & -k_2 Y_r q_s^{\max} b_j - D & 0 & 0 \\ -k_3 Y_r q_c^{\max} \frac{Y_{O_2}}{Y_o} a_i & k_3 Y_r q_s^{\max} b_j & -D & 0 \\ -k_5 Y_{O_2} q_c^{\max} a_i & 0 & 0 & -D - Kl a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad (44)$$

$i, j = 1, 2$

The fuzzy rules for the RF partial model are stated as:

If $z_1(t)$ is " $\varphi_{11}(z_1(t))$ " and $z_2(t)$ is " $\varphi_{21}(z_2(t))$ "
THEN $\dot{x}_{11}^{RF}(t) = A_{11}^{RF} x(t) + Bu(t) + d$

If $z_1(t)$ is " $\varphi_{11}(z_1(t))$ " and $z_2(t)$ is " $\varphi_{22}(z_2(t))$ "
THEN $\dot{x}_{12}^{RF}(t) = A_{12}^{RF} x(t) + Bu(t) + d$

If $z_1(t)$ is " $\varphi_{12}(z_1(t))$ " and $z_2(t)$ is " $\varphi_{21}(z_2(t))$ "
THEN $\dot{x}_{21}^{RF}(t) = A_{21}^{RF} x(t) + Bu(t) + d$

If $z_1(t)$ is " $\varphi_{12}(z_1(t))$ " and $z_2(t)$ is " $\varphi_{22}(z_2(t))$ "
THEN $\dot{x}_{22}^{RF}(t) = A_{22}^{RF} x(t) + Bu(t) + d$

The aggregated model for the RF partial model is given by

$$\begin{aligned} \dot{x}^{RF}(t) &= \sum_{i=1}^2 \sum_{j=1}^2 \varphi_{1i}(z_1(t))\varphi_{2j}(z_2(t))\{A_{ij}^{RF}x(t) + Bu(t) + d\} \\ y^{RF}(t) &= \sum_{i=1}^2 \sum_{j=1}^2 \varphi_{1i}(z_1(t))\varphi_{2j}(z_2(t))Cx(t), \quad i, j = 1, 2. \end{aligned} \tag{45}$$

4.2 The Respirative Fuzzy Exact Model

The R partial exact model can also be built following the procedure described in section 4.1. We must be aware that the R model must be split in two models called R_{qe1} and R_{qe2} . As in schemes (31-33) several possibilities may be formulated to build the R model, therefore a possibility to express the $f_R(x(t))$ matrix (f_{Rqe1} and f_{Rqe2}) can be written as:

Scheme $f_{Rqe1,I}$

$$f_{Rqe1} = \begin{bmatrix} Y_e q_e^{\max} K_i \frac{x_3}{(K_e + x_3)(K_i + x_2)} - D & Y_o q_s^{\max} \frac{x_1}{K_s + x_2} & 0 & 0 \\ 0 & -k_1 Y_o q_s^{\max} \frac{x_1}{K_s + x_2} - D & 0 & 0 \\ -k_4 Y_e q_e^{\max} K_i \frac{x_3}{(K_e + x_3)(K_i + x_2)} & 0 & -D & 0 \\ -k_6 Y_e q_e^{\max} K_i \frac{x_3}{(K_e + x_3)(K_i + x_2)} & -k_5 Y_o q_s^{\max} \frac{x_1}{K_s + x_2} & 0 & -D - K_{La} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{46}$$

Scheme $f_{Rqe2,I}$

$$f_{Rqe2} = \begin{bmatrix} Y_{O_2} e q_c^{\max} \frac{x_4}{K_o + x_4} - D & \left(Y_o q_s^{\max} - Y_{O_2} e q_s^{\max} \frac{Y_o}{Y_{O_2}} \right) \frac{x_1}{K_s + x_2} & 0 & 0 \\ 0 & -k_1 Y_o q_s^{\max} \frac{x_1}{K_s + x_2} - D & 0 & 0 \\ -k_4 Y_{O_2} e q_c^{\max} \frac{x_4}{K_o + x_4} & k_4 Y_{O_2} e q_s^{\max} \frac{Y_o}{Y_{O_2}} \frac{x_1}{K_s + x_2} & -D & 0 \\ -k_6 Y_{O_2} e q_c^{\max} \frac{x_4}{K_o + x_4} & \left(-k_5 Y_o q_s^{\max} + k_6 Y_{O_2} e q_s^{\max} \frac{Y_o}{Y_{O_2}} \right) \frac{x_1}{K_s + x_2} & 0 & -D - K_{La} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{47}$$

To construct the exact model for the R partial model we must use the nonlinearities from models (46-47). For model 46 we have that the first nonlinearity is given by

$$NL_5 = z_3^{Rqe1}(t) = \frac{x_3}{(K_e + x_3)(K_i + x_2)}; \quad \text{for } x_3 \neq -K_e \text{ and } x_2 \neq -K_i \tag{48}$$

where the maximum and minimum values are given by

$$\max_{x_2(t)x_3(t)} z_3(t) = 9.8039 = c_1 \quad \min_{x_2(t)x_3(t)} z_3(t) = 0.1 = c_2 \quad (49)$$

The remaining nonlinearities from models (46) and (47) were the same already described by (37) and (39). The linear subsystems for the R_{qe1} model can be obtained from

$$A_{R_{qe1}} = \begin{bmatrix} Y_e q_e^{\max} K_i c_k - D & Y_o q_s^{\max} b_j & 0 & 0 \\ 0 & -k_1 Y_o q_s^{\max} b_j - D & 0 & 0 \\ -k_4 Y_e q_e^{\max} K_i c_k & 0 & -D & 0 \\ -k_6 Y_e q_e^{\max} K_i c_k & -k_5 Y_o q_s^{\max} b_j & 0 & -D \end{bmatrix} \quad (50)$$

$jk = 1, 2$

and

$$A_{R_{qe2}} = \begin{bmatrix} Y_{O_2} e q_c^{\max} a_i - D & \left(Y_o q_s^{\max} - Y_{O_2} e q_s^{\max} \frac{Y_o}{Y_{O_2}} \right) b_j & 0 & 0 \\ 0 & -k_1 Y_o q_s^{\max} b_j - D & 0 & 0 \\ -k_4 Y_{O_2} e q_c^{\max} a_i & k_4 Y_{O_2} e q_s^{\max} \frac{Y_o}{Y_{O_2}} b_j & -D & 0 \\ -k_6 Y_{O_2} e q_c^{\max} a_i & \left(-k_5 Y_o q_s^{\max} + k_6 Y_{O_2} e q_s^{\max} \frac{Y_o}{Y_{O_2}} \right) b_j & 0 & -D \end{bmatrix} \quad (51)$$

$ij = 1, 2$

A general model to obtain the rules for the R_{qe1} and R_{qe2} partial models is expressed as

for R_{qe1}

If $z_1(t)$ is " $\varphi_{1i}(z_1(t))$ " and $z_3(t)$ is " $\varphi_{3k}(z_3(t))$ ")"

$$\text{THEN } \dot{x}^{R_{qe1}}(t) = A_{jk}^{R_{qe1}} x(t) + Bu(t) + d; \quad jk = 1, 2$$

and for R_{qe2}

If $z_1(t)$ is " $\varphi_{1i}(z_1(t))$ " and $z_2(t)$ is " $\varphi_{2j}(z_2(t))$ ")"

$$\text{THEN } \dot{x}^{R_{qe2}}(t) = A_{ij}^{R_{qe2}} x(t) + Bu(t) + d; \quad ij = 1, 2$$

Finally the aggregated model for the R_{qe1} and R_{qe2} partial models is expressed as:
for R_{qe1}

$$\begin{aligned} \dot{x}^{R_{qe1}}(t) &= \sum_{j=1}^2 \sum_{k=1}^2 \varphi_{1j}(z_1(t)) \varphi_{3k}(z_3(t)) \{A_{jk}^{R_{qe1}} x(t) + Bu(t) + d\} \\ y^{R_{qe1}}(t) &= \sum_{j=1}^2 \sum_{k=1}^2 \varphi_{1j}(z_1(t)) \varphi_{3k}(z_3(t)) Cx(t), \quad j, k = 1, 2. \end{aligned} \tag{52}$$

and for R_{qe2}

$$\begin{aligned} \dot{x}^{R_{qe2}}(t) &= \sum_{i=1}^2 \sum_{j=1}^2 \varphi_{1i}(z_1(t)) \varphi_{2j}(z_2(t)) \{A_{ij}^{R_{qe2}} x(t) + Bu(t) + d\} \\ y^{R_{qe2}}(t) &= \sum_{i=1}^2 \sum_{j=1}^2 \varphi_{1i}(z_1(t)) \varphi_{2j}(z_2(t)) Cx(t), \quad i, j = 1, 2. \end{aligned} \tag{53}$$

Although the premise variables for the partial models RF and R_{qe2} were the same (37) and (39), they have different behaviors as they are multiplied by different yield coefficients. The aggregated models (45) and (52-53) represents exactly the nonlinear system (13) in the region

$$x_1(t) \in [0, 10], x_2(t) \in [0, 1], x_3(t) \in [0, 5] \text{ and } x_4(t) \in [0, 0.007]$$

A condition for the transition between the RF - R - RF partial models is given by (28).

5. Fuzzy Observer

Now that an exact fuzzy model for the nonlinear baker’s yeast partial model has been obtained, a fuzzy observer can now be designed. First of all we have to test the observability matrix for the obtained linear subsystems. A linear system is said to be observable if for any unknown initial state $x(0)$ there exist a finite $t_1 > 0$ such as the knowledge of the input u and the output y over $[0, t_1]$ suffices to determine uniquely the initial state $x(0)$. Otherwise the equation is unobservable (Chen, 1999). The pair (A,C) is observable if and only if the observability matrix

$$O = [C \ CA \ CA^2, \dots, CA^{n-1}]^T = n, \tag{54}$$

has full rank $(\rho(O) = n)$ i.e. is nonsingular.

In section 4.1 we remark that the fuzzy exact model for the RF model may be built from three schemes (among many others) namely f_{RF_I} (31), f_{RF_II} (32) and f_{RF_III} (33). If we build the fuzzy exact model for each scheme (31-33) and we test the observability matrix for these linear subsystems; for example (44), we should find that (table 3)

Schemes	observability rank for C=[0 0 1 1]
f_{RF_I}	3 3 3 3
f_{RF_II}	4 4 4 3
f_{RF_III}	4 4 4 4 4 3 4 3

Table 3. Observability matrix for schemes (31-33)

From table 3 we may notice that no full rank is achieved for f_{RF_I} , therefore a full observer cannot be built for this scheme. For schemes $f_{RF_{II}}$ and $f_{RF_{III}}$ almost full rank is achieved in every linear subsystem; however, for scheme $f_{RF_{III}}$ it will take eight linear subsystems to build the fuzzy exact model, while for scheme $f_{RF_{II}}$ only four linear subsystems will be needed. To avoid build complicated linear systems, scheme $f_{RF_{II}}$ was chosen to construct the Exact fuzzy observer. Therefore before constructing a fuzzy exact model for an observer or a controller, it will be advisable to analyze the way the premise variables are chosen to avoid lack of observability or controllability.

The following assumptions were made to build the fuzzy observer:

H1. The nominal values of the yield coefficients k_l , $-k_6$ are constant and known.

H2. The ethanol, the dissolved oxygen concentration and the OTR are known.

The procedure to build the exact fuzzy observer is the same that was followed for the fuzzy exact model, although some considerations must be taken into account. An important consideration is related with the scheme (32) where the premise variable (39) will depend on the estimated state x_1 and x_2 , therefore the premise variable must be modified to:

$$\hat{z}_2^{RF}(t) = \frac{\hat{x}_1}{K_s + \hat{x}_2}; \quad \text{for } \hat{x}_2 \neq -K_s \quad (55)$$

The same situation applies to the premise variable of model (46)

$$z_3^{Rqe1}(t) = \frac{x_3}{(K_e + x_3)(K_i + \hat{x}_2)}; \quad \text{for } x_3 \neq -K_e \text{ and } \hat{x}_2 \neq -K_i \quad (56)$$

The premise variable (37) remains unchanged. To guarantee full observability rank (table 4) the minimum values of the premise variables are modified to

$$\min_{x_4(t)} z_1(t) = a_2 = \min_{x_1(t)x_2(t)} z_2(t) = b_2 = \min_{x_2(t)x_3(t)} z_3(t) = c_2 = 0.1 \quad (57)$$

Schemes	Linear subsystems observability rank
$f_{RF_{II}}$	4 4 4 4
f_{Rqe1_I}	4 4 4 4
f_{Rqe2_I}	4 4 4 4

Table 4 Observability matrix for the linear subsystems (44, 50-51)

The membership functions are built as before; nevertheless, for (54-55) we have

$$\begin{aligned}
\varphi_{21}(\hat{z}_2(t)) &= \frac{\hat{z}_2(t) - b_2}{b_1 - b_2} & \varphi_{22}(\hat{z}_2(t)) &= \frac{-\hat{z}_2(t) + b_1}{b_1 - b_2} \\
\varphi_{31}(\hat{z}_3(t)) &= \frac{\hat{z}_3(t) - c_2}{c_1 - c_2} & \varphi_{32}(\hat{z}_3(t)) &= \frac{-\hat{z}_3(t) + c_1}{c_1 - c_2}
\end{aligned} \tag{58}$$

The linear subsystems given by (44), (50-51) are used to built the fuzzy observer. A general rule to obtain all the fuzzy rules for the RF , R_{qe1} and R_{qe2} partial models are given by:

for RF

If $\hat{z}_1(t)$ is " $\varphi_{1i}(z_1(t))$ " and $z_2(t)$ is " $\varphi_{2j}(z_2(t))$ "
THEN $\hat{x}^{RF}(t) = A_{ij}^{RF} \hat{x}(t) + Bu(t) + K_{ij}^{RF} (y(t) - \hat{y}(t)) + K_L a C^{sat}$; $j = 1, 2$

for R_{qe1}

If $\hat{z}_1(t)$ is " $\varphi_{1j}(z_1(t))$ " and $\hat{z}_3(t)$ is " $\varphi_{3k}(z_3(t))$ "
THEN $\hat{x}^{Rqe1}(t) = A_{jk}^{Rqe1} \hat{x}(t) + Bu(t) + K_{jk}^{Rqe1} (y(t) - \hat{y}(t)) + K_L a C^{sat}$; $jk = 1, 2$

for R_{qe2}

If $\hat{z}_1(t)$ is " $\varphi_{1i}(z_1(t))$ " and $z_2(t)$ is " $\varphi_{2j}(z_2(t))$ "
THEN $\hat{x}^{Rqe2}(t) = A_{ij}^{Rqe2} \hat{x}(t) + Bu(t) + K_{ij}^{Rqe2} (y(t) - \hat{y}(t)) + K_L a C^{sat}$; $ij = 1, 2$

The aggregated fuzzy observers for the RF , R_{qe1} and R_{qe2} partial models are given by

for RF

$$\begin{aligned}
\hat{x}^{RF}(t) &= \sum_{i=1}^2 \sum_{j=1}^2 \varphi_{1i}(\hat{z}_1(t)) \varphi_{2j}(z_2(t)) \left\{ A_{ij}^{RF} \hat{x}(t) + Bu(t) + K_{ij}^{RF} (y(t) - \hat{y}(t)) + K_L a C^{sat} \right\} \\
\hat{y}^{RF}(t) &= \sum_{i=1}^2 \sum_{j=1}^2 \varphi_{1i}(\hat{z}_1(t)) \varphi_{2j}(z_2(t)) C \hat{x}(t), \quad i, j = 1, 2.
\end{aligned} \tag{59}$$

for R_{qe1}

$$\begin{aligned}
\hat{x}^{Rqe1}(t) &= \sum_{j=1}^2 \sum_{k=1}^2 \varphi_{1j}(\hat{z}_1(t)) \varphi_{3k}(\hat{z}_3(t)) \left\{ A_{jk}^{Rqe1} \hat{x}(t) + Bu(t) + K_{jk}^{Rqe1} (y(t) - \hat{y}(t)) + K_L a C^{sat} \right\} \\
\hat{y}^{Rqe1}(t) &= \sum_{j=1}^2 \sum_{k=1}^2 \varphi_{1j}(\hat{z}_1(t)) \varphi_{3k}(\hat{z}_3(t)) C \hat{x}(t), \quad j, k = 1, 2.
\end{aligned} \tag{60}$$

for R_{qe2}

$$\begin{aligned}\hat{x}^{R_{qe2}}(t) &= \sum_{i=1}^2 \sum_{j=1}^2 \varphi_{1i}(\hat{z}_1(t)) \varphi_{2j}(z_2(t)) \{ A_{ij}^{R_{qe2}} \hat{x}(t) + Bu(t) + K_{ij}^{R_{qe2}} (y(t) - \hat{y}(t)) + K_L a C^{sat} \} \\ \hat{y}^{R_{qe2}}(t) &= \sum_{i=1}^2 \sum_{j=1}^2 \varphi_{1i}(\hat{z}_1(t)) \varphi_{2j}(z_2(t)) C \hat{x}(t), \quad i, j = 1, 2.\end{aligned}\quad (61)$$

5.1 Fuzzy Observer Simulation

The application of the proposed observer scheme was simulated using MATLAB™. The fuzzy observers were tested using the continuous *RF* and the *R baker's yeast* partial models given above. The inlet substrate concentration was varied between 3 g/l and 10 g/l in order to force the switching between the partial models. The partial models parameters were given in table 2. The decay rate (α) was set to zero. The estimated variables were the biomass and the substrate, each observed variable was tested with three different initial conditions 1, 3 and 4 g/l for biomass, and 0.01, 0.03 and 0.06 g/l for substrate. The behavior of the fuzzy observer for biomass estimation is shown in figure 7. The observer converges around the 20 hours of fermentation elapsed time, almost in the R_{qe2} partial model. It can be noticed the dynamics of the baker's yeast switching through the *RF*, R_{qe1} and R_{qe2} partial models. The observer substrate converges around the 15 hours of fermentation elapsed time (figure 8), therefore the substrate dynamics is faster than the biomass. The observer gains are displayed in table 5 and were calculated from the inequalities (8) through Linear Matrix Inequalities.

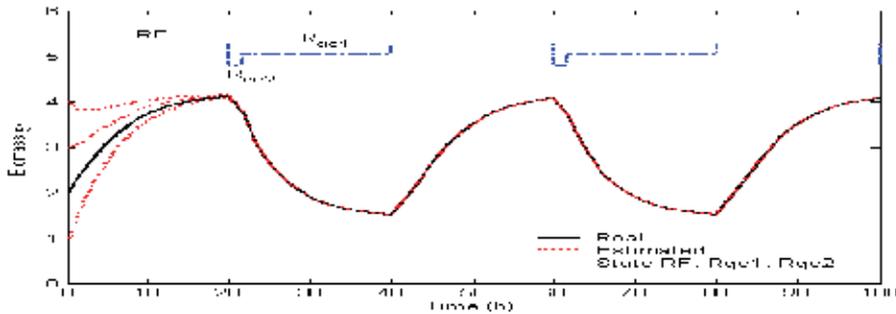


Fig. 7 Biomass observer performance with $\alpha = 0$ and $\hat{x}_1(0) = 1, 3$ and 4 g/l.

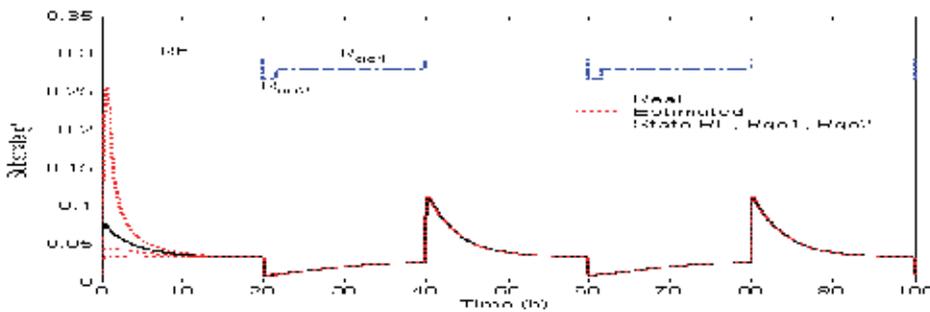


Fig. 8 Substrate observer performance with $\alpha = 0$ and $\hat{x}_2(0) = 0.01, 0.03$ and 0.06 g/l.

Gain	x_1	x_2	x_3	x_4
K_{1_RF}	-1309.8	2019.2	409.79	-498.19
K_{2_RF}	-707.7	-1299.2	1553.5	-1611.5
K_{3_RF}	-1304.4	2015.3	405.83	-494.33
K_{4_RF}	-702.28	-1303.1	1549.6	-1607.6
K_{1_Rqe1}	1937.1	-4136.6	113.86	-213.31
K_{2_Rqe1}	1945.6	-4153.3	113.56	-213.03
K_{3_Rqe1}	-58.432	-11.52	94.846	-192.06
K_{4_Rqe1}	-49.883	-28.246	94.539	-191.77
K_{1_Rqe2}	-845.59	1626.8	109	-205.81
K_{2_Rqe2}	-12.963	-1569.3	1112.7	-1185.8
K_{3_Rqe2}	-842.39	1620.4	108.95	-205.76
K_{4_Rqe2}	-9.761	-1575.6	1112.7	-1185.8

Table 5. Observer gains, with $\alpha = 0$.

Common positive definite matrices that guarantees global asymptotic stability (Tanaka & Wang, 2001), were found for each partial model, namely

$$P_{RF} = \begin{bmatrix} 2.4375 \times 10^{-4} & -1.7425 \times 10^{-4} & -1.7842 \times 10^{-4} & 1.7363 \times 10^{-4} \\ -1.7425 \times 10^{-4} & 9.6041 \times 10^{-4} & -3.3103 \times 10^{-4} & 3.2221 \times 10^{-4} \\ -1.7842 \times 10^{-4} & -3.3103 \times 10^{-4} & 3.9367 \times 10^{-4} & -3.8324 \times 10^{-4} \\ 1.7363 \times 10^{-4} & 3.2221 \times 10^{-4} & -3.8324 \times 10^{-4} & 3.9792 \times 10^{-4} \end{bmatrix},$$

$$P_{Rqe1} = \begin{bmatrix} 5.7188 \times 10^{-5} & -1.1189 \times 10^{-4} & -2.0528 \times 10^{-6} & 1.9285 \times 10^{-6} \\ -1.1189 \times 10^{-4} & 2.313 \times 10^{-4} & -1.0663 \times 10^{-6} & 1.192 \times 10^{-6} \\ -2.0528 \times 10^{-6} & -1.0663 \times 10^{-6} & 3.8217 \times 10^{-6} & -3.7426 \times 10^{-6} \\ 1.9285 \times 10^{-6} & 1.192 \times 10^{-6} & -3.7426 \times 10^{-6} & 7.6266 \times 10^{-6} \end{bmatrix},$$

$$P_{Rqe2} = \begin{bmatrix} 1.0333 \times 10^{-4} & -2.0491 \times 10^{-4} & -1.5395 \times 10^{-6} & 1.4853 \times 10^{-6} \\ -2.0491 \times 10^{-4} & 7.8655 \times 10^{-4} & -2.4702 \times 10^{-4} & 2.4119 \times 10^{-4} \\ -1.5395 \times 10^{-6} & -2.4702 \times 10^{-4} & 1.7446 \times 10^{-4} & -1.7037 \times 10^{-4} \\ 1.4853 \times 10^{-6} & 2.4119 \times 10^{-4} & -1.7037 \times 10^{-4} & 1.8174 \times 10^{-4} \end{bmatrix}$$

To improve the observer performance the decay rate ratio (α) was set to 0.3. The behavior of the fuzzy observer for biomass estimation is shown in figure 9. The observer converges in about 6 hours of fermentation elapsed time, now within the *RF* state. The observer substrate converges around the 5 hours of fermentation elapsed time (figure 10). The observer gains for $\alpha = 0.3$ are displayed in table 6 and were calculated using the inequalities given by (8).

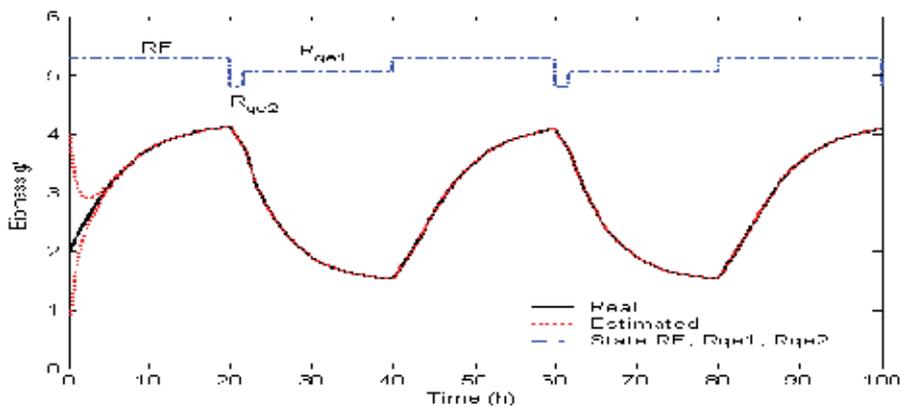


Fig. 9 Biomass observer performance with $\alpha = 0.3$ and $\hat{x}_1(0) = 1, 3$ and 4 g/l.

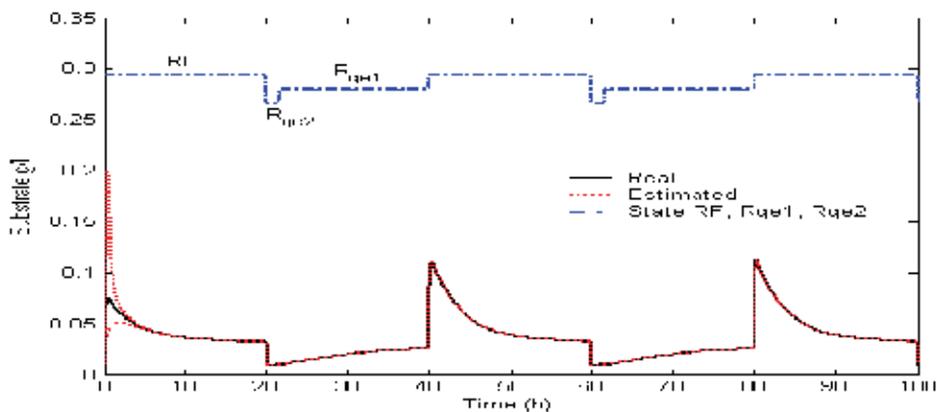


Fig. 10 Substrate observer performance with $\alpha = 0$ and $\hat{x}_2(0) = 0.01, 0.03$ and 0.06 g/l.

Gain	x_1	x_2	x_3	x_4
K_{1_RF}	-45325	10803	1898.6	-1944.6
K_{2_RF}	-15713	-270.57	2674.9	-2707.9
K_{3_RF}	-43990	10667	1784.5	-1833.4
K_{4_RF}	-14378	-406.29	2560.8	-2596.7
K_{1_Rqe1}	-84248	-5319.8	1619.2	-1647.6
K_{2_Rqe1}	-66998	-5161.4	1320.5	-1362.6
K_{3_Rqe1}	-65489	-714.01	1220	-1262.5
K_{4_Rqe1}	-48239	-555.65	921.3	-977.46
K_{1_Rqe2}	-41344	5226.6	3786.8	-3783.1
K_{2_Rqe2}	-24341	1700.4	2765.6	-2790.6
K_{3_Rqe2}	-40157	5110.9	3684.8	-3683.8
K_{4_Rqe2}	-23155	1584.6	2663.5	-2691.3

Table 6. Observer gains, with $\alpha = 0.3$.

Common positive definite matrices that guarantees global asymptotic stability (Tanaka & Wang, 2001), were found for each partial model, namely

$$\begin{aligned}
 P_{RF} &= \begin{bmatrix} 1.6199 \times 10^{-4} & -3.8358 \times 10^{-5} & -3.5281 \times 10^{-5} & 3.44 \times 10^{-5} \\ -3.8358 \times 10^{-5} & 1.936 \times 10^{-5} & 4.2418 \times 10^{-6} & -4.1297 \times 10^{-6} \\ -3.5281 \times 10^{-5} & 4.2418 \times 10^{-6} & 1.094 \times 10^{-5} & -1.067 \times 10^{-5} \\ 3.44 \times 10^{-5} & -4.1297 \times 10^{-6} & -1.067 \times 10^{-5} & 1.0781 \times 10^{-5} \end{bmatrix}; \\
 P_{Rqe1} &= \begin{bmatrix} 1.0525 \times 10^{-3} & 1.6867 \times 10^{-5} & -4.4263 \times 10^{-5} & 4.2217 \times 10^{-5} \\ 1.6867 \times 10^{-5} & 2.4884 \times 10^{-5} & -1.3321 \times 10^{-6} & 1.3043 \times 10^{-6} \\ -4.4263 \times 10^{-5} & -1.3321 \times 10^{-6} & 2.0592 \times 10^{-6} & -1.9649 \times 10^{-6} \\ 4.2217 \times 10^{-5} & 1.3043 \times 10^{-6} & -1.9649 \times 10^{-6} & 2.3272 \times 10^{-6} \end{bmatrix}; \\
 P_{Rqe2} &= \begin{bmatrix} 1.7858 \times 10^{-3} & -3.6232 \times 10^{-4} & -3.4184 \times 10^{-4} & 3.3329 \times 10^{-4} \\ -3.6232 \times 10^{-4} & 0.00019792 \times 10^{-4} & 2.6424 \times 10^{-5} & -2.5682 \times 10^{-5} \\ -3.4184 \times 10^{-4} & 2.6424 \times 10^{-5} & 9.4507 \times 10^{-5} & -9.218 \times 10^{-5} \\ 3.3329 \times 10^{-4} & -2.5682 \times 10^{-5} & -9.218 \times 10^{-5} & 9.4605 \times 10^{-5} \end{bmatrix}
 \end{aligned}$$

From (58, 59 and 60) an exact fuzzy observer for a nonlinear baker's yeast model was designed. The fuzzy estimator had a satisfactory behavior. A different approach to construct a fuzzy observer using the whole term $OTR=K_L a(C^{sat}-x_4)$ as a known and constant perturbation was reported in (Herrera, 2007a). In this case a partial observer was constructed due that full rank in the observability matrix could not be achieved.

6. The Fuzzy Exact Model, ($u(t)=D$).

The construction of the fuzzy exact model for a continuous baker's yeast fermentation can become quite complex when the output of the system is given by $u(t)=D$, for example for the *RF* partial model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} \mu_{s_RF}^o + \mu_{s_RF}^r & 0 & 0 & 0 \\ -k_1 \mu_{s_RF}^o - k_2 \mu_{s_RF}^r & 0 & 0 & 0 \\ k_3 \mu_{s_RF}^r & 0 & 0 & 0 \\ -k_5 \mu_{s_RF}^o & 0 & 0 & -K_L a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -x_1 \\ -x_2 + S_{in} \\ -x_3 \\ -x_4 \end{bmatrix} D + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} K_L a C^{sat} \quad (62)$$

In this case the input matrix is not constant anymore, depending now on the variables x_1, x_2, x_3, x_4 . So we define the new premise variable as

$$z_{x_1}(t) = x_1, \quad z_{x_2}(t) = x_2, \quad z_{x_3}(t) = x_3, \quad z_{x_4}(t) = x_4, \quad (63)$$

The new premise variables may be written as

$$\begin{aligned} \varphi_{41}(z_{x_1}(t)) &= \frac{z_{x_1}(t) - d_2}{d_1 - d_2} & \varphi_{42}(z_{x_1}(t)) &= \frac{-z_{x_1}(t) + d_1}{d_1 - d_2} \\ \varphi_{51}(z_{x_2}(t)) &= \frac{z_{x_2}(t) - e_2}{e_1 - e_2} & \varphi_{52}(z_{x_2}(t)) &= \frac{-z_{x_2}(t) + e_1}{e_1 - e_2} \\ \varphi_{61}(z_{x_3}(t)) &= \frac{z_{x_3}(t) - f_2}{f_1 - f_2} & \varphi_{62}(z_{x_3}(t)) &= \frac{-z_{x_3}(t) + f_1}{f_1 - f_2} \\ \varphi_{71}(z_{x_4}(t)) &= \frac{z_{x_4}(t) - g_2}{g_1 - g_2} & \varphi_{72}(z_{x_4}(t)) &= \frac{-z_{x_4}(t) + b_1}{b_1 - b_2} \end{aligned} \quad (64)$$

where the maximum and minimum values of (63) are displayed in table 7.

Premise variable	maximum	minimum
$z_{x_1}(t)$	$d_1 = 10$	$d_2 = 0$
$z_{x_2}(t)$	$e_1 = 1$	$e_2 = 0$
$z_{x_3}(t)$	$f_1 = 5$	$f_2 = 0$
$z_{x_4}(t)$	$g_1 = 0.007$	$g_2 = 0$

Table 7. Maximum and minimum values for $z_{x_1}(t)$, $z_{x_2}(t)$, $z_{x_3}(t)$ and $z_{x_4}(t)$

The input matrix can now be written as

$$B_{lmno} = [-d_1, -e_m + S_{in}, -f_n, -g_o]^T \quad (65)$$

The other premise variables are still given by (37) and (39). A general rule to construct all the fuzzy rules can be stated as

If $z_1(t)$ is " $\varphi_{1i}(z_1(t))$ " and $z_2(t)$ is " $\varphi_{2j}(z_2(t))$ " and $z_{x_1}(t)$ is " $\varphi_{4i}(z_2(t))$ " and $z_{x_2}(t)$ is " $\varphi_{5m}(z_2(t))$ " and $z_{x_3}(t)$ is " $\varphi_{6n}(z_2(t))$ " and $z_{x_4}(t)$ is " $\varphi_{7o}(z_2(t))$ "

$$\text{THEN } \dot{x}^{RF}(t) = A_{ijlmno}^{RF} x(t) + B_{ijlmno} u(t) + d; \quad ijlmno = 1, 2$$

It must be remarked that 64 subsystems would be needed to construct the RF partial model. Finally the aggregated fuzzy system for the RF partial model is given by

$$\begin{aligned}\dot{x}^{RF}(t) &= \sum_1^{64} h_{\psi_{RF}}(z(t)) \left\{ A_{ijlmno}^{RF} x(t) + B_{ijlmno}^{RF} u(t) + d \right\} \\ y^{RF}(t) &= \sum_1^{64} h_{\psi_{RF}}(z(t)) Cx(t), \quad ijlmno = 1, 2.\end{aligned}\tag{66}$$

where

$$\begin{aligned}\psi_{RF} &= o + 2(n-1) + 4(m-1) + 8(l-1) + 16(j-1) + 32(i-1), \\ h_{\psi_{RF}}(z(t)) &= h_{\psi_{RF}}(z(t)) = \varphi_{1i}(z_1(t))\varphi_{2j}(z_2(t))\varphi_{4l}(z_{x_1}(t)) \cdot \\ &\quad \varphi_{5m}(z_{x_2}(t))\varphi_{6n}(z_{x_3}(t))\varphi_{7o}(z_{x_4}(t))\end{aligned}\tag{67}$$

The R_{qe1} and R_{qe2} fuzzy exact model were constructed following the same rules and also 64 subsystems were obtained for each partial model. As stated before now the exact fuzzy model gets quite complex because it will be necessary 192 subsystems to represent the RF, R_{qe1} and R_{qe2} partial models. From the fuzzy exact model built for the case explained a fuzzy observer can also be built, more details are reported in (Herrera, 2007b). A multiple Takagi-Sugeno multiple controller was designed to force the switching between the RF and the R baking yeast partial models (Herrera, 2007c; Herrera, 2007d). The substrate fuzzy controller tracked a square reference signal varied between 0.01 g/l and 0.07 g/l. S_{in} was set to 5 g/l. It is worth noting that the controller was capable to force the switching along the partial models.

7. Conclusion

Based on the idea of splitting the baker's yeast model, a novel TS fuzzy model was proposed using the sector nonlinearities method, giving an exact representation of the original nonlinear plant. Moreover, an observer for each partial model was constructed. It is worth noting that the observer was capable of switching along the partial models, without performance degradation. Therefore, the approach presented here may be considered a valid method to design an observer. Future work will include the experimental validation of the fuzzy observer and optimal controllers for fed-batch fermentation cultures.

8. Acknowledgments

This work has been supported by the Mexican Consejo Nacional de Ciencia y Tecnología (CONACyT), under grants 46538 and 41148.

9. References

Ascencio, P.; Sbarbaro, D. & Azevedo, S. (2004). An adaptive fuzzy hybrid state observer for bioprocesses. *IEEE Transactions on Fuzzy Systems*, Vol. 12, No. 5, pp. 641-651, ISSN 1063-6706

- Bastin, G. & Dochain, D. (1990). *On Line Estimation and Adaptive Control of Bioreactors*. Elsevier, ISBN 10-0-444-88430-0, Amsterdam
- Braatz, R.; & VanAntwerp (2000). A tutorial on linear and bilinear matrix inequalities. *Journal of Process Control*, Vol. 10, pp. 363-385, ISSN 0959-1524
- Cazzador, L. & Luvenova, V. (1995). Nonlinear estimation of specific growth rate for aerobic fermentation processes. *Biotechnology and Bioengineering*, Vol. 47, pp. 626-632, ISSN 1097-0290
- Chen, C. (1999). *Linear Systems Theory and Design*, Oxford University Press, ISBN 0030716918, United States of America
- Dochain, D. (2003). State and parameter estimation in chemical and biochemical processes: a tutorial. *Journal of Process Control*, Vol. 13, pp. 801-818, ISSN 0959-1524
- Farza, M.; Nadri, M. & Hammouri, H. (2000). Nonlinear observation of specific growth rate in aerobic fermentation processes. *Bioprocess Engineering*, Vol. 23, pp. 359-366, ISSN 1615-7591
- Ferreira, E.C. (1995). Identificação e controlo adaptivo de processos biotecnológicos, *PhD Dissertation* (in Portuguese), Universidade Do Porto
- Genovesi, A.; Harmand, J. & Steyer, J.P. (1999). A fuzzy logic based diagnosis system for the on-line supervision of an anaerobic digester pilot-plant. *Biochemical Engineering Journal*, Vol. 3, No. 3, pp. 171-183, ISSN 1369-703X
- Guay, M. & Zhang, T (2002). Adaptive nonlinear observer for microbial growth processes. *Journal of process control*, Vol. 12, pp.633-643, ISSN 0959-1524
- Herrera, L.; Castillo, T.; Ramírez, J. & Ferreira, E.C. (2007a). Exact fuzzy observer for a baker's yeast fermentation process, *10th Computer Applications in Biotechnology*, pp. 309-314, Cancún México, June 6-8, International Federation of Automatic Control, preprints, to be published at the IFAC-PapersOnLine website
- Herrera, L.; Castillo, T.; Ramírez, J. & Ferreira, E.C. (2007b). Exact fuzzy observer for a fed-batch baker's yeast fermentation process, *IEEE International Conference on Fuzzy Systems*, pp. 1-6. London England July 23-26, ISBN 1-4244-1210-2/07, IEEEExplore ISSN 1098-7584
- Herrera, L.; Castillo, T.; Ramírez, J. & Ferreira, E.C. (2007c). Takagi-Sugeno multiple model controller for a continuous baking yeast fermentation process, *4th International Conference on Informatics in Control, Automation and Robotics*, pp. 436-439, Angers France, May 9-12, ISBN 978-972-8865-82-5
- Herrera, L. (2007d). Sobre el problema de la observación y control de un modelo difuso para un proceso fermentativo conmutado, *Ph. D. Thesis (In spanish)*, Centro de Investigación y de Estudios Avanzados del I.P.N, Unidad Guadalajara, México
- Horiuchi, J. & Kishimoto, M. (1998). Fuzzy-aided estimation of biological parameters based on material balances. *Journal of Fermentation and Bioengineering*, Vol. 86, pp. 111-117, ISSN 0922-338X
- Karakazu, C.; Türker, M. & Öztürk, S. (2006). Modeling, on-line state estimation and fuzzy control of production scale fed-batch baker's yeast fermentation. *Control Engineering Practice*, Vol. 14, No. 1, pp. 959-974, ISSN 0967-0661
- Lubenova, V.; Rocha, I. & Ferreira, E.C. (2003). Estimation of multiple biomass growth rates and biomass concentrations in a class of bioprocesses. *Bioprocess Biosyst Eng*, Vol. 25, pp. 395-406, ISSN: 1615-7591

- Oliveira, R.; Ferreira, E.C. & Feyo de Azevedo, S. (2002). Stability dynamics of convergence and tuning of observer based kinetic estimators. *Journal of Process Control*, Vol. 12, pp. 311-323, ISSN 0959-1524
- Pormeileau, P. (1990). Modelisation et controle d'un procédé fed-batch de culture des levures á pain (*Saccharomyces cerevisiae*). *Ph. D. dissertation*. Ecole Polytechnique de Montréal, Canada
- Soh, Y.C. & Cao, W. (1999). Multi-rate nonlinear state and parameter estimation in bioreactors. *Biotechnology and Bioengineering*, Vol. 63, No. 1, pp. 22-32, ISSN 1097-0290
- Sonnleitner, B. & Käppeli, O. (1986). Growth of *Saccharomyces cerevisiae* is controlled by its limited respiratory capacity: formulation and verification of a hypothesis. *Biotechnology and Bioengineering*, Vol. 28, pp. 927-937, ISSN 1097-0290
- Takagi, T. & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE trans. Sys. Man. Cyber.*, Vol. 15, pp. 116-132, ISSN 0018-9472
- Tanaka, K. & Wang, H. (2001). *Fuzzy Control Systems Design and Analysis a Linear Matrix Inequality Approach*, John Wiley & Sons, ISBN 0-471-32324, United States of America
- Veloso, C.; Rocha, I & Ferreira, E.C. (2007) Estimation of biomass concentration using interval observer in an E. coli fed-batch fermentation, *10th Computer Applications in Biotechnology*, pp. 99-104, Cancún México, June 6-8, International Federation of Automatic Control, preprints
- Yamuna, R. & Ramachandra, R. (1999). Control of fermenters - a review. *Bioprocess Engineering*, Vol. 21, pp. 77-88, ISSN 1615-7591

An Intelligent Marshalling Plan Using a New Reinforcement Learning System for Container Yard Terminals

Yoichi Hirashima
Osaka Institute of Technology
Japan

1. Introduction

In recent years, the number of shipping containers grows rapidly, and in many container yard terminals, increasing throughput of material handling operation becomes important issue as well as decreasing the turnaround times of vessels. Material handling operations for loading containers into a vessel is highly complex, and the complexity grows at an exponential rate according to the growth of the number of containers, the operation occupy a large part of the total run time of shipping at container terminals. A challenge of this chapter is focused on improving throughput of the material handling operations for loading container on a vessel by using reinforcement learning. Commonly, materials are packed into containers and each container in a vessel has its own position determined by the destination, weight, owner, and so on (Siberholz et al., 1991; Günther & Kim, 2005). Thus, containers have to be loaded into a ship in a certain desired order because they cannot be rearranged in the ship. Therefore, containers must be rearranged before loading if the initial layout is different from the desired layout. Containers carried into the terminal are stacked randomly in a certain area called bay and a set of bays are called yard. The rearrangement process conducted within a bay is called marshalling.

In the problem, the number of stacks in each bay is predetermined and the maximum number of containers in a stack is limited. Containers are moved by a transfer crane and the destination stack for the container in a bay is selected from the stacks being in the same bay. In this case, a long series of container movements is often required to achieve a desired layout, and results that are derived from similar initial layouts can be quite different. Problems of this type have been solved by using techniques of optimization, such as genetic algorithm (GA) and multi agent method (Koza, 1992; Minagawa & Kakazu, 1997). These methods can successfully yield some solutions for block stacking problems. However, they adopt the environmental model different from the marshalling process, and do not assure to obtain the desired layout of containers.

Another candidate for solving the problem is the reinforcement learning (Watkins & Dayan, 1992), which is known to be effective for learning under unknown environment that has the

Markov Property. The Q-learning, one of the realization algorithm for the reinforcement learning can be applied to generate marshalling plan, when all the estimates of evaluation-values for pairs of the layout and container movement are obtained. These values are called "Q-value". The optimal series of container movements can be obtained by selecting the movement that has the best evaluation for each layout. However, conventional Q-learning has to store evaluation-values for all the layout-movement pairs. Therefore, the conventional Q-learning has great difficulties for solving the marshalling problem, due to its huge number of learning iterations required to obtain admissible plan (Baum, 1999). Recently, a Q-learning method that can generate marshalling plan has been proposed (Motoyama et al., 2001). Although these methods were effective for several cases, the desired layout was not achievable for every trial so that the early-phase performances of learning process can be spoiled. This chapter introduces a new Q-learning method for marshalling plan, and some additional methods to improve learning performances. The learning process in the proposed method is consisted of two stages: 1. determination of rearrangement order, 2. selection of destination for removal containers. Each stage has a corresponding learning algorithm, and Q-values in one stage are referred from the learning algorithm in the other stage. Stages are repeated sequentially in accordance with container movements and Q-values are discounted according to the number of container movements, so that Q-values reflect the total number of container movements. Consequently, selecting the best Q-values leads the best series of container movements required to obtain a desired layout. Moreover, each rearranged container is placed into the desired position so that every trial can achieve one of desired layouts. In addition, in the proposed method, each container has several desired positions in the final layout, and the feature is considered in the learning algorithm. Thus, the early-phase performances of the learning process can be improved.

The remainder of the chapter is organized as follows. The marshalling process in container yard terminals is elaborated in section 2, following the problem description. In section 3, a learning algorithm of the proposed method is detailed, and a data storage structure for storing Q-values is explained in this section. Computer simulations are conducted for several cases and proposed method is compared to conventional ones in section 4. Finally, concluding remarks are given in section 5.

2. Problem description

Fig.1 shows an example of container yard terminal. The terminal consists of containers, yard areas, yard transfer cranes, auto-guided vehicles, and port crane. Containers are carried by trucks and each container is stacked in a corresponding area called bay and a set of bays constitutes a yard area. Each bay has n_y stacks that m_y containers can be laden, the number of containers in a bay is k , and the number of bays depends on the number of containers. Each container is recognized by an unique name c_i ($i=1, \dots, k$). A position of each container is discriminated by using discrete position numbers, $1, \dots, n_y m_y$. Then, the position of the container c_i is described by x_i ($1 \leq i \leq k, 1 \leq x_i \leq n_y m_y$), and the state of a bay is determined by the vector, $x = [x_1, \dots, x_k]$.

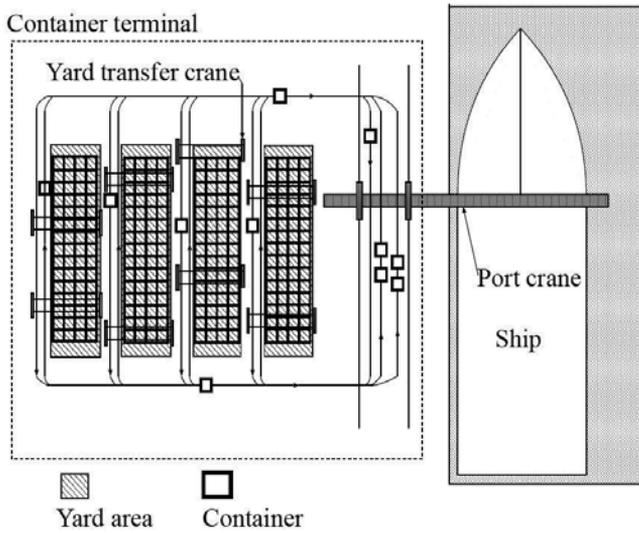


Fig. 1. Container terminal

2.1 Grouping

The desired layout in a bay is generated based on the loading order of containers that are moved from the bay to a ship. In this case, the container to be loaded into the ship can be anywhere in the bay if it is on top of a stack. This feature yields several desired layouts for the bay. Thus, in the addressed problem, when containers on different stacks are placed at the same height in a desired layout, it is assumed that the desired positions of such containers can be exchanged. Fig.2 shows an example of desired layouts, where $m_y = n_y = 3, k = 9$. In this example, containers are loaded in the ship in the descendent order. Then, containers c_7, c_8, c_9 are in the same group (Group1), and their positions are exchanged because the loading order can be kept unchanged after the exchange of positions. In the same way, c_5, c_5, c_6 are in the Group2, and c_1, c_2, c_3 are in the Group3 where positions of containers can be exchanged. Consequently several candidates for desired layout of the bay are generated from the original desired-layout.

In addition to the grouping explained above, a "heap shaped group" for n_y containers at the top of stacks in original the desired layout (group 1) is generated as follows:

1. n_y containers in group 1 can be placed at any stacks if their height is same as the original one.
2. Each of them can be stacked on other $n_y - 1$ containers when both of followings are satisfied:
 - (a) They are placed at the top of each stack in the original desired-layout,
 - (b) The container to be stacked is loaded into the ship before other containers being under the container.

Other groups are the same as ones in the original grouping, so that the grouping with heap contains all the desired layout in the original grouping.

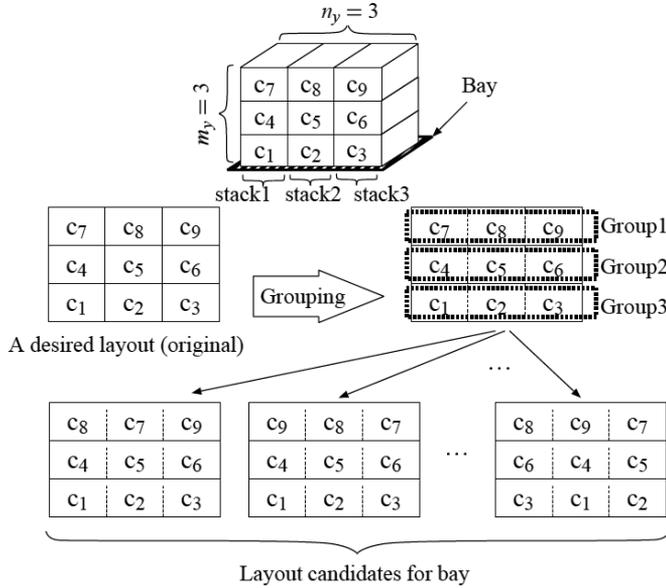


Fig. 2. Layouts for bay

2.2 Marshalling process

The marshalling process consists of 2 stages: ① selection of a container to be rearranged, and ② removal of the containers on the selected container in ①. After these stages, rearrangement of the selected container is conducted. In the stage ②, the removed container is placed on the destination stack selected from stacks being in the same bay. When a container is rearranged, n_y positions that are at the same height in a bay can be candidates for the destination. In addition, n_y containers can be placed for each candidate of the destination. Then, defining t as the time step, $c_a(t)$ denotes the container to be rearranged at t in the stage ①. $c_a(t)$ is selected from candidates $c_{y_{i_1}}$ ($i_1 = 1, \dots, n_y^2$) that are at the same height in a desired layout. A candidate of destination exists at a bottom position that has undesired container in each corresponding stack. The maximum number of such stacks is n_y , and they can have n_y containers as candidates, since the proposed method considers groups in the desired position. The number of candidates of $c_a(t)$ is thus $n_y \times n_y$. In the stage ②, the container to be removed at t is $c_b(t)$ and is selected from two containers $c_{y_{i_2}}$ ($i_2 = 1, 2$) on the top of stacks. In this stage, C_{y_1} is on the $c_a(t)$ and C_{y_2} is

on the destination of $c_a(t)$. Then, in the stage ②, $c_b(t)$ is removed to one of the other stacks in the same bay, and the destination stack $u(t)$ at time t is selected from the candidates u_j ($j = 1, \dots, n_y - 2$). $c_a(t)$ is rearranged to its desired position after all the c_{y_2} s are removed. Thus, a state transition of the bay is described as follows:

$$x_{t+1} = \begin{cases} f(x_t, c_a(t)), & \text{(stage ①)} \\ f(x_t, c_b(t), u(t)), & \text{(stage ②)} \end{cases} \quad (1)$$

where $f(\cdot)$ denotes that removal is processed and x_{t+1} is the state determined only by $c_a(t)$, $c_b(t)$ and $u(t)$ at the previous state x_t . Therefore, the marshalling plan can be treated as the Markov Decision Process.

Additional assumptions are listed below:

- a. The bay is 2-dimensional.
- b. Each container has the same size.
- c. The goal position of the target container must be located where all containers under the target container are placed at their own goal positions.
- d. $k \leq m_y n_y - 2m_y + 1$

The maximum number of containers that must removed before rearrangement of $c_a(t)$ is $2m_y - 1$ because the height of each stack is limited to m_y . Thus, assumption d. assures the existence of space for removing all the $c_b(t)$, and $c_a(t)$ can be placed at the desired position from any state x_t .

Fig.3 shows 3 examples of marshalling process, where $m_y = 3$, $n_y = 5$, $k = 8$. Positions of containers are discriminated by integers 1, ..., 15. The first container to be loaded is c_8 and containers must be loaded by descendent order until c_1 is loaded. In the figure, a container marked with a \square denotes $c_a(t)$, a container marked with a \bigcirc is removed one, and an arrowed line links source and destination positions of removed container. Cases (a),(b) have the same order of rearrangement, c_2, c_7, c_6 , and the removal destinations are different. Whereas, case (c) has the different order of rearrangement, c_8, c_2, c_7 . When no groups are considered in desired arrangement, case (b) requires 5 steps to complete the marshalling process, and other cases require one more step. Thus, the total number of movements of container can be changed by the destination of the container to be removed as well as the rearrangement order of containers.

If groups are considered in desired arrangement, case (b) achieves a goal layout at step2, case (a) achieves at step3, case (c) achieves at step4. If extended groups are considered, cases (a),(b) achieve goal layouts at step2 and case (c) achieves at step4. Since extended goal

layouts include the non-extended goal layouts, and since non-extended goal layouts include a non-grouping goal layout, equivalent or better marshalling plan can be generated by using the extended goal notion as compared to plans generated by other goal notions. The objective of the problem is to find the best series of movements which transfers every container from an initial position to the goal position. The goal state is generated from the shipping order that is predetermined according to destinations of containers. A series of movements that leads a initial state into the goal state is defined as an episode. The best episode is the series of movements having the smallest number of movements of containers to achieve the goal state.

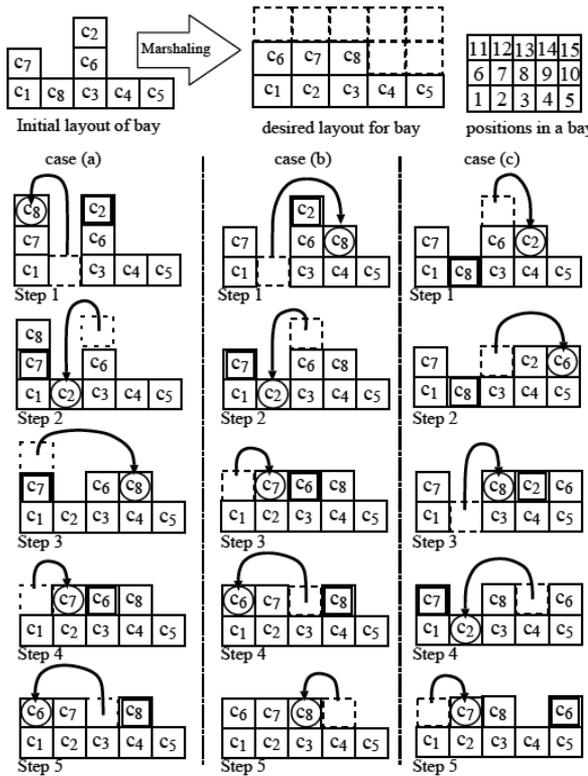


Fig. 3. Marshalling process

3. Reinforcement Learning for Marshalling Plan

3.1 Update rule of Q-values

In the selection of $c_a(t)$, the container to be rearranged, an evaluation value is used for each candidate $c_{y_{i_1}}$ ($i_1 = 1 \dots n_y - 2$). In the same way, evaluation values are used in the selection of the container to be removed $c_b(t)$ and its destination u_j ($j = 1 \dots n_y - 2$). Candidates of $c_b(t)$ is $c_{y_{i_2}}$ ($i_2 = 1, 2$). The evaluation value for the selection of $c_{y_{i_1}}$, $c_{y_{i_2}}$ and u_j at the state x are called Q-values, and a set of Q-values is called Q-table. At the l th episode, the Q-

value for selecting $c_{y_{i_1}}$ is defined as $Q_1(l, x, c_{y_{i_1}})$, the Q-value for selecting $c_{y_{i_2}}$ is defined as $Q_2(l, x, c_{y_{i_1}}, c_{y_{i_2}})$ and the Q-value for selecting u_j is defined as $Q_3(l, x, c_{y_{i_1}}, c_{y_{i_2}}, u_j)$. The initial value for Q_1, Q_2, Q_3 is assumed to be 0.

In this method, a large amount of memory space is required to store all the Q-values referred in every episode. In order to reduce the required memory size, the length of episode that corresponding Q-values are stored should be limited, since long episode often includes ineffective movements of container. In the following, update rule of Q_3 is described. When a series of n movements of container achieves the goal state x_n from an initial state x_0 , all the referred Q-values from x_0 to x_n are updated. Then, defining L as the total counts of container-movements for the corresponding episode, L_{\min} as the smallest value of L found in the past episodes, and s as the parameter determining the threshold, Q_3 is updated by the following equation when $L < L_{\min} + s$ ($s > 0$) is satisfied:

$$Q_3(l+1, x_t, c_a(t), c_b(t), u_t) = (1-\alpha)Q_3(l, x_t, c_a(t), c_b(t), u_t) + \alpha[R + V_{t+1}], \quad (2)$$

$$V_t = \begin{cases} \gamma \max_{y_{i_1}} Q_1(l, x, c_{y_{i_1}}) & \text{(stage ①)}, \\ \gamma \max_{y_{i_2}} Q_2(l, x, c_a(t), c_{y_{i_2}}) & \text{(stage ②)}. \end{cases}$$

where γ denotes the discount factor and α is the learning rate. Reward R is given only when the desired layout has been achieved. L_{\min} is assumed to be infinity at the initial state, and updated by the following equation when $L < L_{\min}$:

$$L = L_{\min}.$$

In the selection of $c_b(t)$, the evaluation value $Q_3(l, x, c_a(t), c_b(t), u_j)$ can be referred for all the u_j ($j = 1 \dots n_y - 2$), and the state x does not change. Thus, the maximum value of $Q_3(l, x, c_a(t), c_b(t), u_j)$ is copied to $Q_2(l, x, c_a(t), c_b(t))$, that is,

$$Q_2(l+1, x, c_a(t), c_b(t)) = \max_j Q_3(l, x, c_a(t), c_b(t), u_j). \quad (3)$$

In the selection of $c_a(t)$, the evaluation value $Q_1(l, x, c_a(t))$ is updated by the following equations:

$$Q_1(l+1, x, c_a(t)) = \begin{cases} \max_{y_{i_1}} Q_1(l, x, c_{y_{i_1}}) + R & \text{(stage ①)}, \\ \max_{y_{i_2}} Q_2(l, x, c_a(t), c_{y_{i_2}}) & \text{(stage ②)}. \end{cases} \quad (4)$$

In order to select actions, the " ϵ -greedy" method is used. In the " ϵ -greedy" method, $c_a(t)$, $c_b(t)$ and a movement that have the largest $Q_1(l, x, c_a(t))$, $Q_2(l, x, c_a(t), c_b(t))$ and $Q_3(l, x, c_a(t), c_b(t), u_j)$ are selected with probability $1-\epsilon$ ($0 < \epsilon < 1$), and they are selected randomly with probability ϵ .

3.2 Learning algorithm

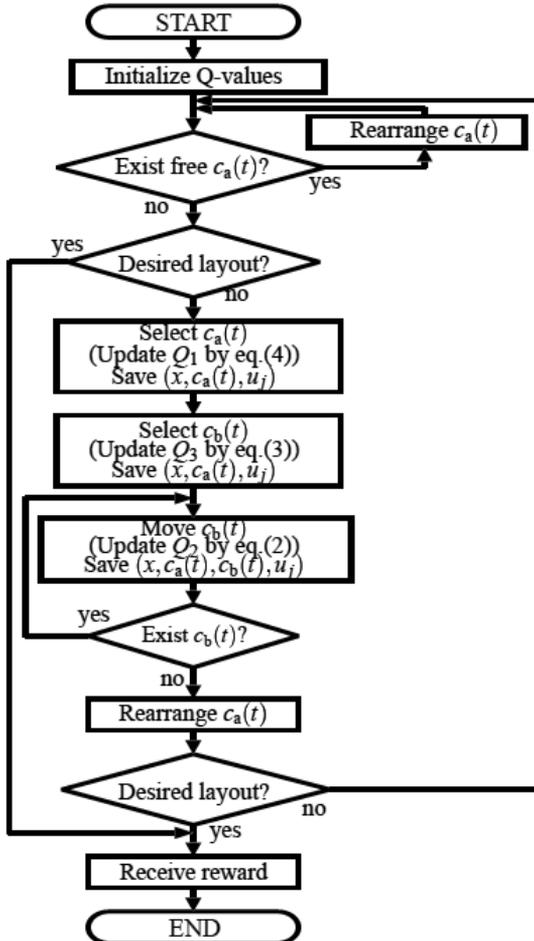


Fig. 4. Flowchart of the learning algorithm

By using the update rule, restricted movements and goal states explained above, the learning process is described as follows:

- I. Count the number of containers being in the goal positions and store it as n
- II. If $n = k$, go to X.
- III. Select $c_a(t)$ to be rearranged
- IV. Store $(x, c_a(t))$

- V. Select $c_b(t)$ to be removed
- VI. Store $(x, c_a(t), c_b(t))$
- VII. Select destination position u_j for $c_b(t)$
- VIII. Store $(x, c_a(t), c_b(t), u_j)$
- IX. Remove $c_b(t)$ and go to V. if another $c_b(t)$ exists, otherwise go to I.
- X. Update all the Q-values referred from the initial state to the goal state according to eqs. (2), (3)

A flow chart of the learning algorithm is depicted in Fig.4.

3.3 Data storage structure for storing Q-values

In the addressed problem, the state of a bay $x = [x_1, \dots, x_k]$ is described by the positions of all the containers. In this case, the number of states of the bay increases by the exponential rate with increase of container counts. Also, evaluation values have to be stored for each state in order to compare candidates of, $c_a(t)$, $c_b(t)$, or u_j . In realistic situations the number of containers is often large, then required memory size to store information for all the state of the bay also becomes large.

Therefore, in the proposed method, binary trees for storing Q-values are constructed dynamically during the course of the learning, so that only Q-values corresponding states that are referred in learning process are stored (Hirashima et al., 1999). This feature can effectively reduce the required memory size for solving a marshalling problem and improve the solution. In the following, data storage structure of a lookup table for storing Q-values are explained.

A set of Q-values stored in a lookup table is called Q-table. In order to construct Q-table by using binary tree, the binary description of x_i ($i = 1 \dots k$) is defined as $b_i = \beta_{i1} \dots \beta_{iI}$ ($\beta_{ij} = 0, 1$; $j = 1, \dots, I$), where I is the order of binary description of x_i . Then, the binary description of x can be described by $B = b_1 \dots b_k$ of order kI , and a binary tree of depth $kI+1$ is used to represent x . At each node of the binary tree, 0 is assigned to left descendant of the node and 1 is assigned to right descendant, and β_{ij} denotes the descendant at the node of depth $I(i-1)+j$. Each leaf of the tree stores state and corresponding Q-value. Given an input to the Q-table, the leaf corresponding to the input is specified by single search by using B . When the input corresponds to the value stored by the leaf, the Q-table outputs the Q-value stored by the leaf. Otherwise, the Q-table outputs 0. Fig.5 depicts a Q-table constructed by a binary tree in the case of $k = m_y = n_y = 2, I=3$. In the figure, inputs $x_w = [1,3]$, $x_e = [4,4]$ are given to the Q-table. Since b_i of x_w is 001, descendants are specified by the order left, left and right from the root. Then, the leaf stores the same state as the input, and the Q-table outputs stored Q-value. While, b_i of x_e is 100, descendants are specified by the order right and left from the root. Then, the state that leaf has is different from the input, and the Q-table outputs 0.

Initially, the tree has only root that has pointer to a leaf having data of state and Q-value. When the referred state has an updated Q-value, 2 consecutive memory units are newly allocated storing pointers to leafs storing data of state and Q-value. The Q-value and corresponding input are stored in another memory unit that is newly allocated for storing data according to β_{ij} . When the next updated Q-value appears, the input and the value pointed by the leaf are compared. When they have the same value, the stored Q-value is update. Otherwise 3 memory units are newly allocated in the memory space, one for data and others for pointers. The algorithm for Q-table construction is described below, and Fig.6 is the flowchart of the algorithm.

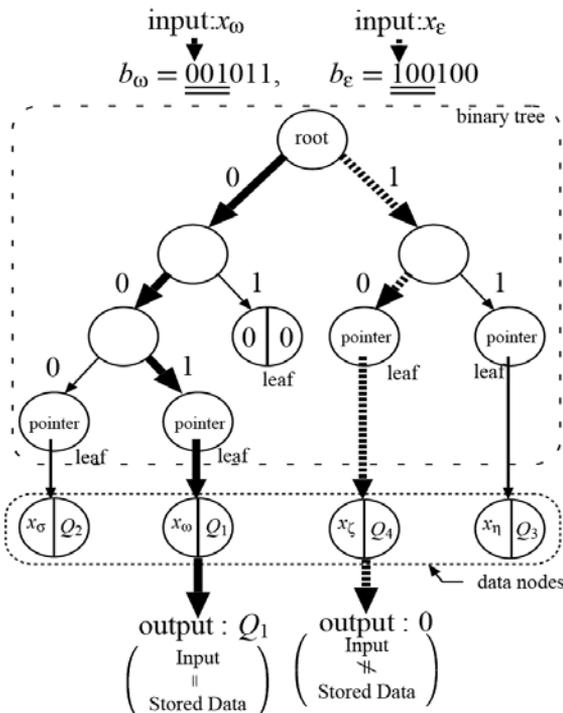


Fig. 5. Structure of Q-table

- (1) Calculate \mathbf{B} from \mathbf{x} and initialize $i=j=1$
- (2) If a memory unit corresponding to \mathbf{B} is a leaf then go to 3, and if it is node then go to (4)
- (3) update i_j by eq.(5)

$$\begin{cases} j \leftarrow j+1, i \leftarrow i & (j < I) \\ j \leftarrow 1, i \leftarrow i+1 & (j = I) \end{cases} \tag{5}$$
 and go to (2).
- (4) Conduct eq.5 again, allocate 2 nodes for expanding a tree, and 1 leaf for storing state and Q-value. Then, copy data from original leaf into corresponding leaf, and store the pointers indicating a new leaf and nodes into original nodes.

- (5) If β_{ij} has the same value as the state stored in the leaf, go to (4). Otherwise, store the new input and Q-value into the corresponding leaf.

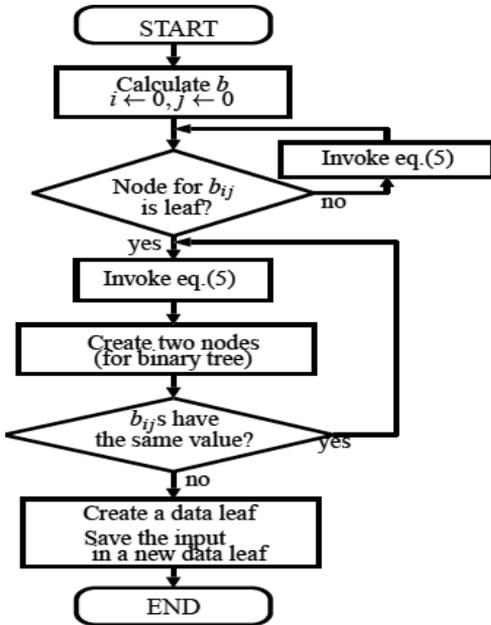


Fig. 6. Flowchart of the Q-table construction

4. Simulations

Computer simulations are conducted for 2 cases, and learning performances are compared for following 5 methods:

- (A) proposed method considering grouping with heap,
- (B) proposed method considering original grouping,
- (C) a learning method using eqs. (2)-(4) as the update rule without grouping (Hirashima et al., 2005),
- (D) method (E) considering original grouping.
- (E) a learning method using eqs. (2),(3) as the update rule, which has no selection of the desired position of $c_a(t)$ (Motoyama et al., 2001).

In methods (D),(E), although the stage ② has the same process as the stage in the method (A), the container to be rearranged, $c_a(t)$, is simply selected from containers being on top of stacks. The learning process used in methods (D),(E) is as follows:

- (i) The number of containers being on the desired positions is defined as k_B and count k_B
- (ii) If $k_B = k$, go to (vi) else go to (iii),
- (iii) Select $c_a(t)$ by using ϵ -greedy method,

- (iv) Select a destination of $c_a(t)$ from the top of stacks by using ϵ -greedy method,
- (v) Store the state and go to (i),
- (vi) Update all the Q-values referred in the episode by eqs. (2),(3).

Since methods (D),(E) do not search explicitly the desired position for each container, each episode is not assured to achieve the desired layout in the early-phase of learning. The flowchart of the learning process in methods (D),(E) is described in Fig.7.

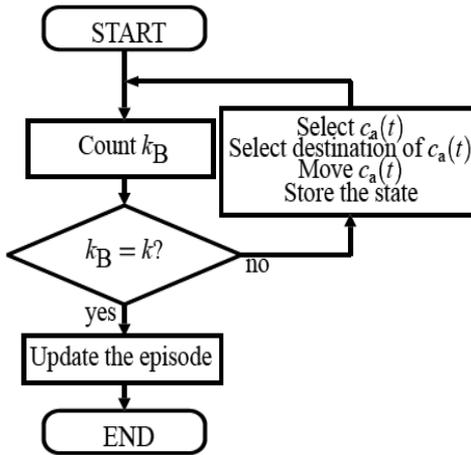


Fig. 7 Flowchart of learning process in methods (D),(E)

In methods (A)-(E), parameters in the yard are set as $k=18$, $m_y = n_y = 6$ that are typical values of marshalling environment in real container terminals. Containers are assumed to be loaded in a ship in descendant order from c_{18} to c_1 . Fig.8 shows an original desired layout for the two cases, and Fig.9 shows corresponding initial layout for each case. Other parameters are put as $\alpha = 0.8$, $\gamma = 0.8$, $R=1.0$, $\epsilon = 0.8$, $s=15$.

Results for case 2 are shown in Fig.10. In the figure, horizontal axis shows the number of trials, and vertical axis shows the minimum number of movements of containers found in the past trials. Each result is averaged over 20 independent simulations. In both cases, solutions that is obtained by methods (A),(B) and (C) is much better as compared to methods (D),(E) in the early-phase of learning, because methods (A),(B),(C) can achieve the desired layout in every trial, whereas methods (D),(E) cannot. Also, methods (A),(B) successfully reduces the number of trials in order to achieve the specific count of container-movements as compared to method (C), since methods (A),(B) considers grouping and finds desirable layouts that can easily diminish the number of movements of container in the early-phase learning. Moreover, at 10000th trail, the number of movements of containers in method (A) is smaller as compared to that in method (B) because, among the extended layouts, method (A) obtained better desired layouts for improving the marshalling process as compared to the layout generated by method (B). Desired layouts generated by methods (A),(B) are depicted in the Fig.11 for case 2.

Method	Case 1		Case 2	
	min. counts	ave. value	min. counts	ave. value
(A)	18	19.10	23	24.40
(B)	20	20.40	25	26.20
(C)	34	35.05	35	38.85
(D)	38	46.90	50	64.00
(E)	148	206.4	203	254.0

Table 1. The best solution of each method for cases 1, 2

The container-movement counts of the best solution and its averaged value for each method are described in Table1. Averaged values are calculated over 20 independent simulations. Among the methods, method (A) derives the best solution with the smallest container-movements. Therefore method (A) can improve the solution for marshalling as well as learning performance to solve the problem.

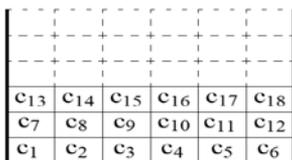


Fig. 8. A desired layout for cases 1,2

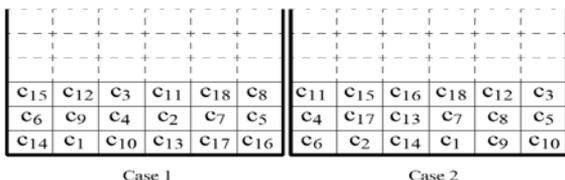


Fig. 9. Initial layouts for cases 1,2

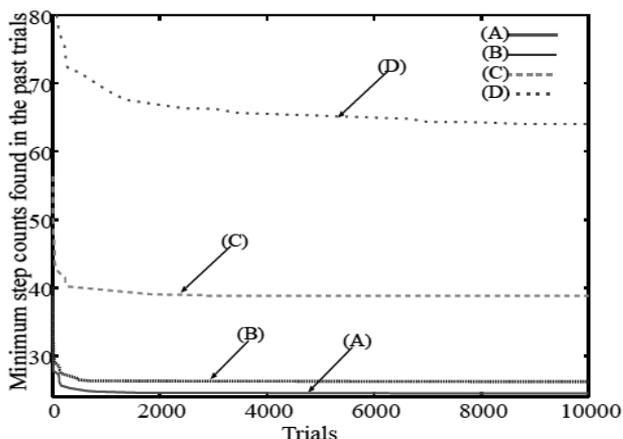


Fig. 10. Performance comparison for case 2

5. Conclusions

A new reinforcement learning system for marshalling plan at container terminals has been proposed. Each container has several desired positions that are in the same group, and the learning algorithm is designed to considering the feature.

In computer simulations, the proposed method could find solutions that had smaller number of movements of containers as compared to conventional methods. Moreover, since the proposed method achieves the desired layout in each trial as well as learns extended desirable layouts, the method can generate solutions with the smaller number of trials as compared to conventional methods.

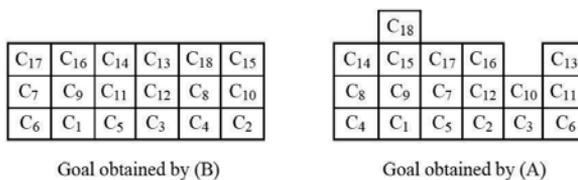


Fig. 11. Final layouts of the best solutions for case 2

6. References

- Baum, E. B. (1999). Toward a model of intelligence as an economy of agents, *Machine Learning*, Vol. 35, 155–185.
- Günther, H.-O. & Kim, K. H. (2005). *Container Terminals and Automated Transport Systems*, pp. 184–206, Springer.
- Hirashima, Y., Iiguni, Y., Inoue, A., & Masuda, S. (1999). Q-learning algorithm using an adaptive-sized Q-table, *Proc. IEEE Conf. Decision and Control*, 1599–1604.
- Hirashima, Y., Takeda, K., Furuya, O., Inoue, A., & Deng, M. (2005). A new method for marshaling plan using a reinforcement learning considering desired layout of containers in terminals, *Preprint of 16th IFAC World Congress*, paperID We-E16-TO/2.
- Koza, J. R. (1992). *Genetic Programming : On Programming Computers by means of Natural Selection and Genetics*, MIT Press.
- Minagawa, M. and Kakazu, Y. (1997). An approach to the block stacking problem by multi agent cooperation, *Trans. Jpn. Soc. Mech. Eng. (in Japanese)*, C-63(608):231–240.
- Motoyama, S., Hirashima, Y., Takeda, K., and Inoue, A. (2001). A marshalling plan for container terminals based on reinforcement learning, *Proc. of Inter. Sympo. on Advanced Control of Industrial Processes*, pages 631–636.
- Siberholz, M. B., Golden, B. L., and Baker, K. (1991). Using simulation to study the impact of work rules on productivity at marine container terminals, *Computers Oper. Res.*, 18(5):433–452.
- Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning, *Machine Learning*, 8:279–292.

Chaotic Neural Network with Time Delay Term for Sequential Patterns

Kazuki HIROZAWA and Yuko OSANA
Tokyo University of Technology
Japan

1. Introduction

Recently, chaos is drawing much attention as a method to realize flexible information processing as well as neural networks, fuzzy logic and genetic algorithms. Chaos is a phenomenon which is observed in deterministic nonlinear systems and the behavior is unpredictable. The chaotic behavior exists in many fields such as hydrodynamics, electric circuits and biological systems. In particular, it is considered that chaos plays an important role in the memory and learning of a human brain (Skarda & Freeman, 1987; Yao & Freeman, 1990; Freeman, 1991).

In order to mimic the real neurons, a chaotic neuron model has been proposed by Aihara *et al.* (Aihara *et al.*, 1990). In this model, chaos is introduced by considering the following properties of the real neurons; (1) spatio-temporal summation, (2) refractoriness and (3) continuous output function. It is known that the dynamic (chaotic) association is realized in the associative memories composed of the chaotic neurons (Aihara *et al.*, 1990; Osana & Hagiwara, 1998a; Osana & Hagiwara, 1998b). These models composed of chaotic neurons can generate chaotic response by setting appropriate parameters.

On the other hand, in nervous system, there are two types of synapses; (1) chemical synapse and (2) electrical synapse. The synaptic delay for a chemical synapse is typically about 2 ms, while the synaptic delay for an electrical synapse may be about 0.2 ms. Although the synaptic delay are constant in the most of the artificial neural network models, we can expect that the associative memory which has two types of weights can realize more complex associations.

In this research, we propose a Chaotic Neural Network with Time Delay term for Sequential Patterns (CNNTDSP). It has two types of connection weights; (1) normal weights and (2) weights with time delay, and realize associations of the sequential pattern in short term, and dynamic associations between sequential patterns in long term.

2. Chaotic Neural Network

Here we briefly review a chaotic neuron model and a chaotic neural network (Aihara *et al.*, 1990).

2.1 Chaotic Neuron Model

A chaotic neuron model based on Caianiello-Nagumo-Sato model(Caianiello, 1961; Nagumo & Sato, 1972) has been proposed by Aihara *et al.* (Aihara et al., 1990). In this model, chaos is introduced by considering spatio-temporal summation, refractoriness and continuous output function; they are observed in real neurons. The mathematical expression of this model is given by

$$x(t) = f \left[A(t) - \alpha \sum_{d=0}^t k^d x(t-d) - \theta \right] \quad (1)$$

$$f(u) = \frac{1}{1 + \exp(-u/\varepsilon)} \quad (2)$$

where $x(t)$ shows the output of the neuron at the time t , $A(t)$ is the strength of the externally applied input at the time t , α is the scaling factor of the refractoriness ($\alpha \geq 0$), k is the damping factor of the refractoriness ($0 \leq k < 1$) and θ is the threshold of the neuron. $f(\cdot)$ is the output function. ε is the steepness parameter. The chaotic neuron model can generate chaotic response by setting appropriate parameters.

2.2 Chaotic Neural Network

A neural network composed of the chaotic neurons described in 2.1 is called a chaotic neural network.

The dynamics of the chaotic neuron i in a neural network composed of N chaotic neurons is represented by the following equation(Aihara et al., 1990):

$$x_i(t+1) = f \left[\sum_{j=1}^M v_{ij} \sum_{d=0}^t k_s^d A_j(t-d) + \sum_{j=1}^N w_{ij} \sum_{d=0}^t k_m^d x_j(t-d) - \alpha \sum_{d=0}^t k_r^d x_i(t-d) - \theta_i \right] \quad (3)$$

where $x_i(t)$ shows the output of the neuron i at the time t , M is the number of the external input, v_{ij} is the connection weight from the external input j to the neuron i , $A_j(t)$ is the strength of the external input j at the time t , w_{ij} is the connection weight between the neuron i and the neuron j , α is the scaling factor of the refractoriness, k_s , k_m and k_r are the damping factors and θ_i is the threshold of the neuron i .

2.3 Quick Learning

The Quick Learning(Hattori et al., 1994) has been proposed in order to improve the storage capacity of the Bidirectional Associative Memory(Kosko, 1988). The Quick Learning has two phases; (1) Hebbian Learning and (2) PRLAB (Pseudo-Relaxation Learning Algorithm for BAM)(Oh & Kothari, 1994).

Consider the training set $\{(X^{(1)}, Y^{(1)}) \dots (X^{(p)}, Y^{(p)}) \dots (X^{(P)}, Y^{(P)})\}$ ($X^{(p)} \in \{1, -1\}^N$, $Y^{(p)} \in \{1, -1\}^M$) is stored in the Bidirectional Associative Memory which has N neurons in the X Layer and M neurons in the Y Layer.

2.4 Hebbian Learning

First, the connection weight between the neuron i in the X Layer and the neuron j in the Y Layer w_{ij} is calculated based on the Hebbian learning.

$$w_{ij} = \sum_{p=1}^P X_i^{(p)} Y_j^{(p)} \quad (4)$$

2.5 PRLAB

Next, the connection weights are update based on the PRLAB algorithm. In the PRLAB, the training pair $(X^{(p)}, Y^{(p)})$ is given to the network, and if the conditions shown in Eqs.(5) and (6) are not satisfied, the connection weights and the threshold of the neuron are updated.

$$\left(\sum_{j=1}^N W_{ij} X_i^{(p)} - \theta_{Y_j} \right) Y_j^{(p)} > 0 \quad j = 1 \dots M \quad (5)$$

$$\left(\sum_{j=1}^M W_{ij} Y_j^{(p)} - \theta_{X_i} \right) X_i^{(p)} > 0 \quad i = 1 \dots N \quad (6)$$

The connection weights and the threshold are updated as follows:

For the neuron in the X Layer, if $S_{X_i}^{(p)} \leq 0$,

$$\Delta W_{ij} = -\frac{\lambda}{M+1} \left(S_{X_i}^{(p)} - \xi X_i^{(p)} \right) Y_j^{(p)} \quad (7)$$

$$\Delta \theta_{X_i} = \frac{\lambda}{M+1} \left(S_{X_i}^{(p)} - \xi X_i^{(p)} \right) \quad (8)$$

For the neuron in the Y Layer, if $S_{Y_j}^{(p)} \leq 0$,

$$\Delta W_{ij} = -\frac{\lambda}{N+1} \left(S_{Y_j}^{(p)} - \xi Y_j^{(p)} \right) X_i^{(p)} \quad (9)$$

$$\Delta \theta_{Y_j} = \frac{\lambda}{N+1} \left(S_{Y_j}^{(p)} - \xi Y_j^{(p)} \right) \quad (10)$$

where $\xi (\xi > 0)$ is the relaxation factor and $\lambda \in (0,2)$ is the constant. S_{X_i} and S_{Y_j} are given by

$$S_{X_i} = \sum_{j=1}^M W_{ij} Y_j^{(p)} - \theta_{X_i} \quad (11)$$

$$S_{Y_j} = \sum_{i=1}^N W_{ij} X_i^{(p)} - \theta_{Y_j}. \tag{12}$$

3. Chaotic Neural Network with Time Delay Term for Sequential Patterns

3.1 Outline

The proposed Chaotic Neural Network with Time Delay term for Sequential Patterns (CNNTDSP) has two types of connection weights; (1) normal weights and (2) weights with time delay. The proposed model realizes associations of the sequential pattern in short term and dynamic associations between sequential patterns in long term.

Let consider the case where 2 sequential patterns composed of 3 patterns shown in Fig.1 are memorized in the network. In the network, we can expect that the association of the sequential pattern 1 ($A \rightarrow B \rightarrow C$) is realized for a certain period, and the association of the sequential pattern 2 ($D \rightarrow E \rightarrow F$) is realized on the other period as follows.

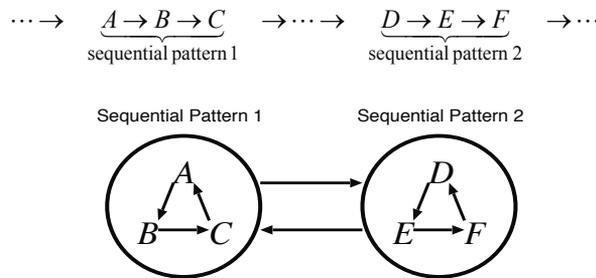


Fig. 1. 2 Sequential Patterns.

3.2 Structure

Figure 2 shows the structure of the proposed model. As shown in Fig.2, the proposed model has two types of weights w and v , and all neurons are connected mutually. N neurons in the network are chaotic neuron.

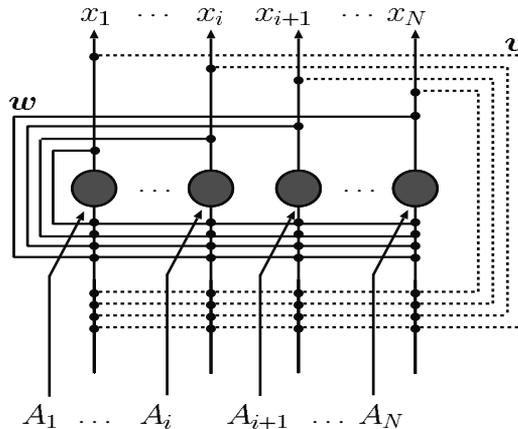


Fig. 2. Structure of Proposed Model.

3.3 Dynamics

In the proposed model, the dynamics of the neuron i is given by

$$x_i(t+1) = g \left[\sum_{d=0}^t k_s^d A_i(t-d) + \sum_{j=1}^N w_{ij} \sum_{d=0}^t k_{m_f}^d x_j(t-d) + \sum_{j=1}^N v_{ij} \sum_{d=0}^t k_{m_s}^d x_j(t-T-d) - \alpha \sum_{d=0}^t k_r^d x_i(t-d) \right] \quad (13)$$

$$g(u) = \tanh(u / \varepsilon) \quad (14)$$

where $x_i(t)$ is the output of the neuron i at the time t , $A_i(t)$ is the external input i at the time t , T is the time delay constant, k_s , k_{m_f} , k_{m_s} and k_r are the damping factors, and α is the scaling factor of refractoriness. w_{ij} and v_{ij} are the connection weights from the neuron i to the neuron j . $g(\cdot)$ is the output function, ε is the steepness parameter.

3.4 Learning Process

In the proposed model, the connection weights w and v are trained by the Quick Learning(Hattori et al., 1994). In order to realize the dynamic associations between sequential patterns in long term, the learning patterns are embedded in the connection weights v as the auto associative weights. In order to realize the associations of the sequential pattern in short term, the sequential patterns are memorized in the connection weights w as the hetero associative weights.

Let consider the case where the training set composed of S sequential patterns $(X^{(1,1)} \rightarrow X^{(1,2)} \rightarrow \dots \rightarrow X^{(1,P(1))} \rightarrow \dots \rightarrow (X^{(s,1)} \rightarrow \dots \rightarrow X^{(s,P(s))} \rightarrow \dots \rightarrow (X^{(S,1)} \rightarrow \dots \rightarrow X^{(S,P(S))})$.

3.4.1 Connection Weights for Auto Association : v

The connection weights for auto association v memorizes all patterns in the training set. Although the Quick Learning was proposed as the learning algorithm for BAM (Kosko, 1988), it can be used to learn the training set for auto association in the single layer network such as the proposed model.

(1) Hebbian Learning

The connection weight v_{ij} is trained as follows:

$$v_{ij} = \begin{cases} \sum_{s=1}^S \sum_{p=1}^{P(s)} X_i^{(s,p)} X_j^{(s,p)}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \quad (15)$$

where S is the number of the sequential patterns, $P(s)$ is the number of the patterns included in the s th sequential pattern and $X_i^{(s,p)}$ is the i th element of training pattern p in the sequence s .

(2) PRLAB

In the PRLAB, all patterns to be stored are given to the network, and if

$$\left(\sum_{j=1}^N v_{ij} X_j^{(s,p)} \right) X_i^{(s,p)} \leq 0 \quad (16)$$

is satisfied, the connection weight v_{ij} is updated as follows:

$$\Delta v_{ij} = -\frac{\lambda}{N+1} \left(\sum_{k=1}^N v_{ik} X_k^{(s,p)} - \xi X_i^{(s,p)} \right) X_j^{(s,p)} \quad (17)$$

where ξ ($\xi > 0$) is the pseudo-relaxation constant and λ ($\lambda \in (0,2)$) is the constant. In this learning, the connection weights are updated by Eq.(17) under the constraint that $v_{ij} = v_{ji}$ and $v_{ii} = 0$.

3.4.2 Connection Weights for Hetero Association : w

The connection weights for hetero association w memorizes the sequential patterns as a limit cycle.

Although the Quick Learning was proposed as the learning algorithm for BAM(Kosko, 1988), it can be used to learn hetero associative training set in the single layer network such as the proposed model.

(1) Hebbian Learning

The initial value of connection weight w_{ij} is calculated as follows:

$$w_{ij} = \sum_{s=1}^S \sum_{p=1}^{P(s)} X_i^{(s,p+1)} X_j^{(s,p)} \quad (18)$$

$$\left(X_i^{(s,P(s)+1)} = X_i^{(s,1)} \right) \square$$

(2) PRLAB

In the PRLAB, all patterns to be stored are given to the network, and if

$$\left(\sum_{j=1}^N w_{ij} X_j^{(s,p)} \right) X_i^{(s,p+1)} \leq 0 \quad (19)$$

is satisfied, the connection weight w_{ij} is updated by

$$\Delta w_{ij} = -\frac{\lambda}{N+1} \left(\sum_{k=1}^N w_{ik} X_k^{(s,p)} - \xi X_i^{(s,p+1)} \right) X_j^{(s,p)} \square \quad (20)$$

4. Computer Experiment Results

In this section, we show the computer experiment results to demonstrate the effectiveness of the proposed model.

4.1 Association Result

Here, the training set which has 2 sequential patterns composed of 3 patterns shown in Fig.3 were memorized.

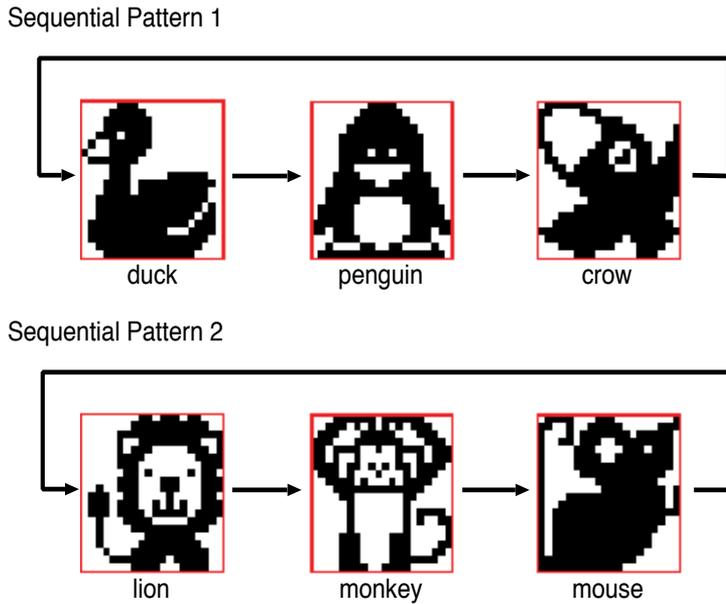


Fig. 3. Stored Patterns.

Figure 4 shows the association result when "mouse" in sequential pattern 2 was given to the network as an initial input in the experiment condition of Fig.4.

As shown in Fig.4, from $t = 1$, the patterns in the sequential pattern 2 were recalled, from $t = 96$, the patterns in the sequential pattern 1 were recalled, and from $t = 246$, the patterns in the sequential pattern 2 were recalled again.

Figure 5 shows the association result of the proposed model when "mouse" in sequential pattern 2 was given to the network as an initial input by the direction cosine. Here, the direction cosine between the output at the time t and the most similar pattern in the sequential pattern s , $\cos \theta^s(t)$ is defined as follows:

$$\cos \theta^s(t) = \cos \theta^{s(c)}(t) \tag{21}$$

$$c = \arg \max_i |\cos \theta^{s(i)}(t)| \tag{22}$$

$$\cos \theta^{s(i)}(t) = \frac{(X^{s(i)} \cdot x(t))}{|X^{s(i)}||x(t)|} \tag{23}$$

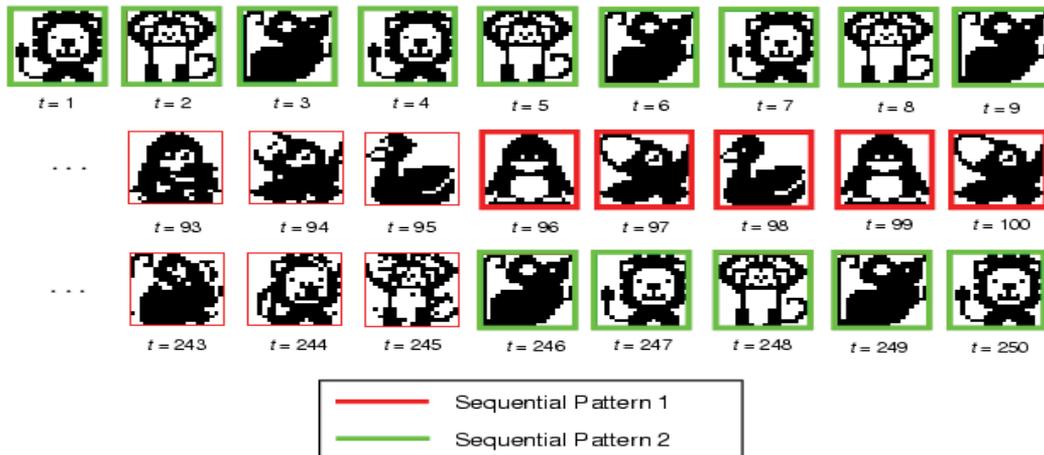


Fig. 4. Association Result when “mouse” in Sequential Pattern 2 was Given.

where $X^{s(i)}$ is the i th stored pattern in the sequential pattern s , $X(t)$ is the output at the time t , $\cos \theta^{s(i)}(t)$ is the direction cosine between the output at the time t , $x(t)$ and the i th pattern in the sequential pattern s , $X^{s(i)}$, and c shows the most similar pattern to the output in the sequential pattern.

As shown in Fig.5, during $t=1 \sim 66$, $t=246 \sim 330$, $t=615 \sim 699$ and $t=984 \sim 1000$, the sequential pattern 2 were recalled, and during $t=96 \sim 218$, $t=353 \sim 429$, $t=458 \sim 587$, $t=722 \sim 798$ and $t=827 \sim 956$, the sequential pattern 1 was recalled. From these results, we confirmed that the proposed model can realize associations of the sequential pattern in short term and dynamic associations between sequential patterns in long term.

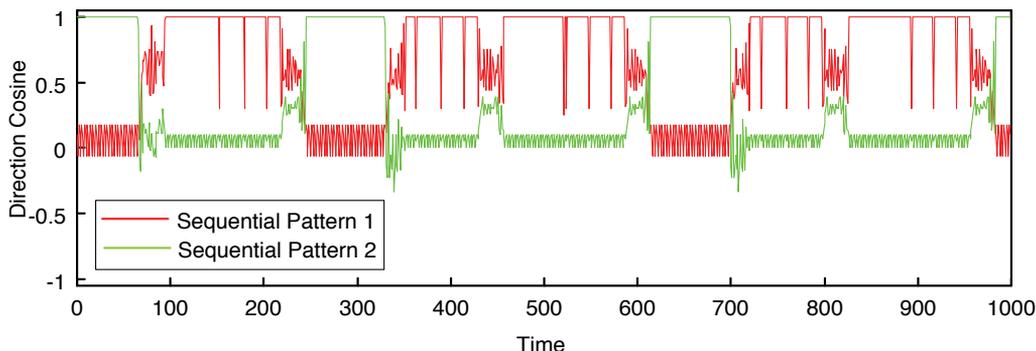


Fig. 5. Association Result when “mouse” in Sequential pattern 2 was Given (Direction Cosine).

4.2 Pattern Dependency of Recall Ability

In the chaotic neural network(Aihara et al., 1990), some patterns are recalled frequently and the other patterns are not recalled or recalled only a few times. However, it is not clear what makes such difference. Here, we examined the pattern dependency of the recall ability.

In this experiment, we used 40 training sets which have 6 patterns (2 sequential patterns composed of 3 patterns). We used 6 patterns shown in Fig.3; (a)duck, (b)penguin, (c)crow, (d)lion, (e)monkey and (f)mouse. When the training set shown in Fig.3 is described as

$$[abc] [def] ,$$

40 training sets can be described as follows:

1: [abc] [def]	2: [abc] [dfe]	3: [acb] [def]
4: [acb] [dfe]	5: [abd] [cef]	6: [abd] [cfe]
7: [adb] [cef]	8: [adb] [cfe]	9: [abe] [cdf]
10: [abe] [cdf]	11: [aeb] [cdf]	12: [aeb] [cdf]
□		
32: [aed] [bfc]	33: [adf] [bce]	34: [adf] [bec]
35: [afd] [bce]	36: [afd] [bec]	37: [aef] [bcd]
38: [aef] [bdc]	39: [afe] [bcd]	40: [afe] [bcd]

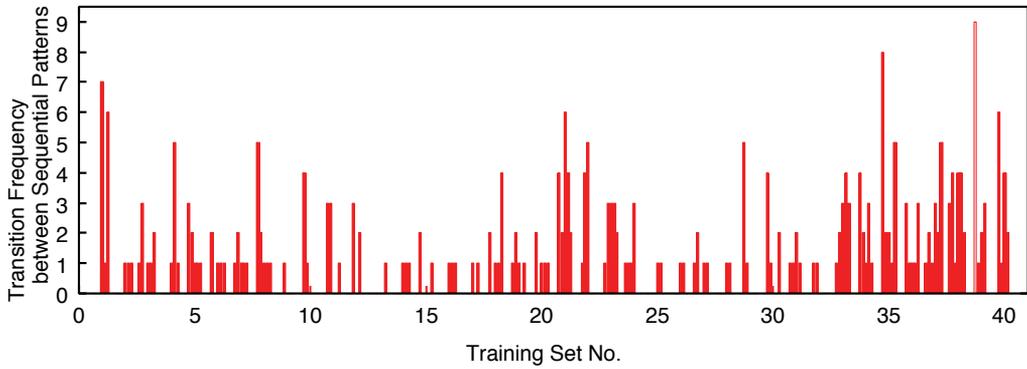
In this experiment, each training set was memorized and the proposed model recalled patterns during $t = 1 \sim 1000$ under the condition shown in Table 1.

Network Parameters		
The Number of Neurons	N	400
Damping Factor	k_s	0.0
Damping Factor	k_m	0.3
Damping Factor	k_r	0.97
Scaling Factor of Refractoriness	α	82
Time Delay Constant	T	14
External Input	$A_i(t)$	0
Quick Learning Parameters		
Relaxation Constant	ξ	0.1
Pseudo Relaxation Constant	λ	1.8

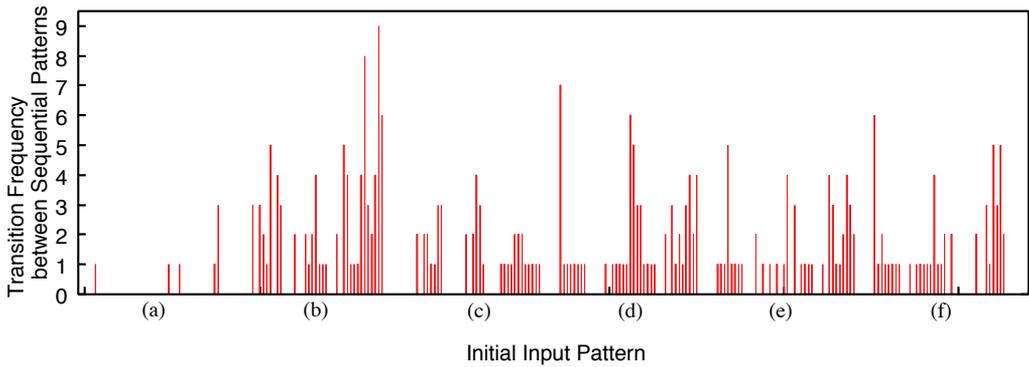
Table 1. Experiment Conditions.

Figures 6~8 show the pattern dependency of the recall ability. In Figs.6~8, (a) shows the result sorted by the training set number, and (b) shows the result sorted by the initial input pattern. Figure 6 shows the transition frequency between sequential patterns. As shown in Fig.6, the transition frequency is large when the pattern (b) was given as an initial input. Figure 7 shows the number of unique stored patterns which were recalled. As shown in Fig.7, when the pattern (a) was given as an initial input, only 3 or 4 patterns were recalled. In contrast, when the patterns (b)~(f) were given, all patterns were often recalled.

Figure 8 shows the average recall frequency of stored patterns. As shown in Fig.8, the patterns (d) ~ (f) were given as an initial input, the average recall frequency of stored patterns is larger than that for the patterns (a) ~ (c). From these results, we confirmed that the initial input pattern influences the recall ability in the proposed model.

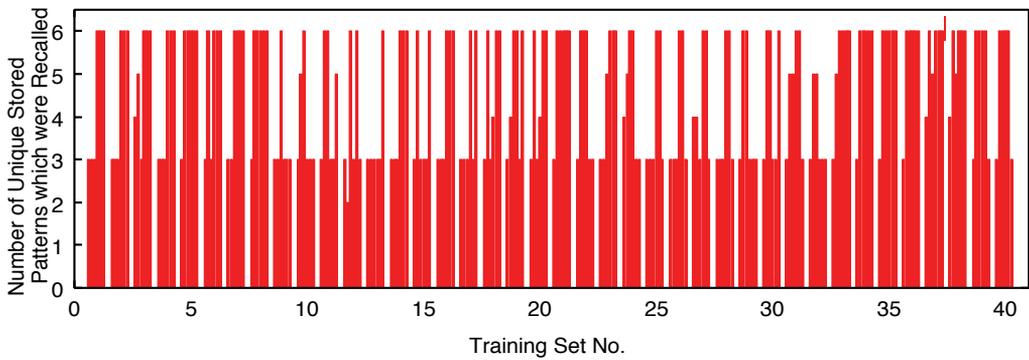


(a) Sorted by Training Set.

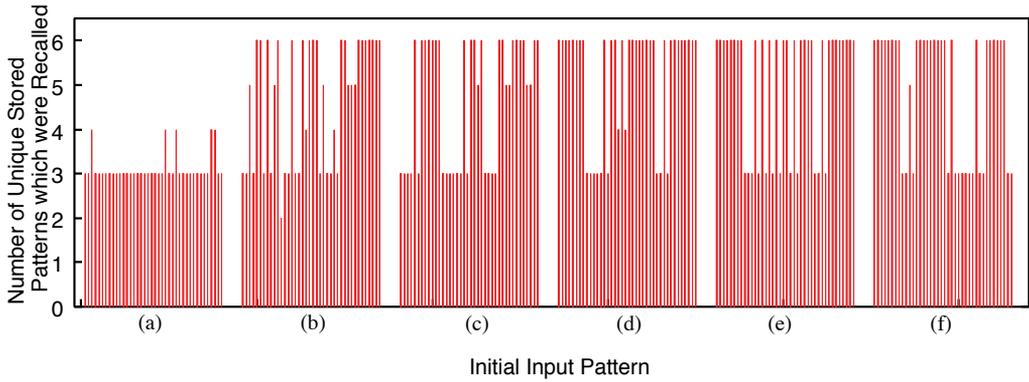


(b) Sorted by Initial Patterns.

Fig. 6. Transition Frequency between Sequential Patterns.

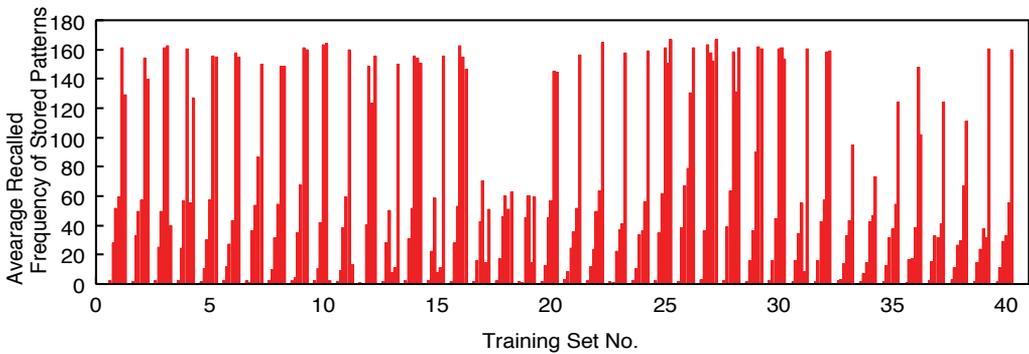


(a) Sorted by Training Set.

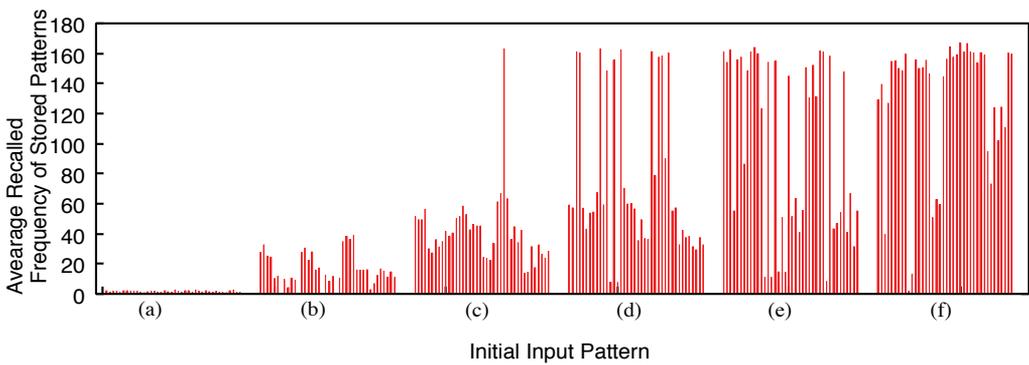


(b) Sorted by Initial Patterns.

Fig. 7. Number of Unique Stored Patterns which were Recalled.



(a) Sorted by Training Set.



(b) Sorted by Initial Patterns.

Fig. 8. Average Recalled Frequency of Stored Patterns.

5. Conclusion

In this research, we have proposed the Chaotic Neural Network with Time Delay term for Sequential Patterns (CNNTDSP). The proposed model is based on the conventional chaotic

neural network and has two types of connection weights; (1) normal weights for hetero associations and (2) weights with time delay for auto associations. The proposed model deal with the sequential patterns. In the proposed model, associations of the sequential pattern in short term and dynamic associations between sequential patterns in long term are realized. We carried out a series of computer experiments and confirmed the proposed model can realize associations of the sequential pattern in short term and dynamic associations between sequential patterns in long term.

6. References

- Aihara, K.; Takabe, T. & Toyoda, M. (1990). Chaotic neural networks. *Physics Letter A*, Vol.144, No.6, 7, pp.333-340
- Caianiello, E. R. (1961). Outline of a Theory of Thought-Processes and Thinking Machines. *Journal Theoretical Biology*, Vol.2, pp.204-235
- Freeman, W. J. (1991). The physiology of perception. *Scientific American*, Vol.264, pp.78-85
- Hattori, M.; Hagiwara, M. & Nakagawa, M. (1994). Quick learning for bidirectional associative memory. *IEICE Japan*, Vol.E77-D, No.4, pp.385-392
- Kosko, B. (1988). Bidirectional associative memories. *IEEE Transactions on Neural Networks*, Vol.18, No.1, pp.49-60
- Nagumo, J. & Sato, S. (1972). On a response characteristic of a mathematical neuron model. *Kybernetik*, pp.155-164
- Oh, H. & Kothari, S. C. (1994). Adaption of the relaxation method for learning in bidirectional associative memory. *IEEE Transactions on Neural Networks*, Vol.5, No.4, pp.576-583
- Osana, Y. & Hagiwara, M. (1998). Improved chaotic associative memory for various associations. *Proceedings of International Symposium on Nonlinear Theory and Its Applications*, pp.467-470, Crans-Montana
- Osana, Y. & Hagiwara, M. (1998). Separation of superimposed pattern and many-to-many associations by chaotic neural networks. *Proceedings of IEEE and INNS International Joint Conference on Neural Networks*, Vol.1, pp.514-519, Anchorage
- Skarda, C. A. & Freeman, W. J. (1987). How brains make chaos in order to make sense of the world. *Behavioral and Brain Science*, Vol.10, pp.161-195
- Yao, Y. & Freeman, W. J. (1990). Model of biological pattern recognition with spatially chaotic dynamics. *Neural Networks*, Vol.3, No.2, pp.153-170

PDE based approach for segmentation of oriented patterns

Aymeric Histace¹, Michel Ménard², Christine Cavarro-Ménard³

¹ETIS UMR CNRS 8051, ENSEA, univ cergyponoise, F-95000 Cergy

²L3i, Université de la Rochelle, F-17000 La Rochelle

³LISA, Université d'Angers, F-49000

France

1. Introduction

Image data restoration by diffusion equation is now a well established approach since the pioneering work of Perona and Malik (Perona & Malik, 1990). Originally, image diffusion consists in a convolution by a Gaussian kernel which introduces a scale dimension related to

the standard deviation of the Gaussian kernel $\sigma = \sqrt{2t}$. This convolution is equivalent to solve the following linear diffusion equation, known in physics as the heat equation:

$$\frac{\partial \psi(x, y, t)}{\partial t} = \Delta \psi(x, y, t) \quad (1)$$

with $\psi(x, y, 0) = \psi_0$

where ψ is the restored image at scale t , and ψ_0 denotes original noisy image.

Nevertheless, Eq. (1) leads to an isotropic smoothing of the original image without edge preservation. To solve this drawback, Perona and Malik (Perona & Malik, 1990) have proposed a non linear adaptive diffusion process. In order to reduce the smoothing effect near edges due to the diffusion process, the authors introduce within classical diffusion process of Eq. (1) a parameter function of the local gradient intensity. Then, they obtained the following Partial Differential Equation (PDE):

$$\frac{\partial \psi(x, y, t)}{\partial t} = \text{div} \left(g(\|\nabla \psi(x, y, t)\|) \nabla \psi(x, y, t) \right) \quad (2)$$

with $\psi(x, y, 0) = \psi_0$

where g is a decreasing function of the gradient. Practical implementation of the Perona-Malik equation gives generally interesting results; Noise is almost suppressed and edges

remain stable on many scales. Nevertheless, it appears in (Catté et al, 1992) that noise can also be enhanced introducing strong oscillations, due to the non convergence of the process. Solutions were proposed in (Alvarez & Morel, 1992), (Catté et al, 1992) and (Komprobst et al, 1997) but, as the authors noticed, when diffusion is processed many times sharp edges are smoothed.

A global coherent based method was proposed by Weickert (Weickert, 1995). Author builds up a tensor driven diffusion D , imposing the way to diffuse along the smallest contrast direction and the orthogonal one and weighting the diffusion according to a measure of coherence:

$$\frac{\partial \psi(x, y, t)}{\partial t} = \text{div}(D \cdot \nabla \psi(x, y, t)) \quad (3)$$

with $\psi(x, y, 0) = \psi_0$

Many others works deal with PDE's based image restoration in corresponding literature: (Geman & Reynolds, 1992; Nitzberg & Shiota, 1992; Whitaker & Pizer, 1993; Weickert, 1995; Weickert, 1998; Terebes et al., 2002; Tschumperle & Deriche, 2002; Tschumperle and Deriche, 2005).

In (Deriche & Faugeras, 1996), authors propose a unique PDE to express the whole principle of PDE's based restoration approach:

$$\frac{\partial \psi}{\partial t} = \frac{\phi'(\|\nabla \psi\|)}{\|\nabla \psi\|} \psi_{\xi\xi} + \phi''(\|\nabla \psi\|) \psi_{\eta\eta} \quad (4)$$

with $\psi(x, y, 0) = \psi_0$

where $\eta = \nabla \psi / \|\nabla \psi\|$, $\xi \perp \eta$ (Fig. 1) and Φ is a decreasing function.

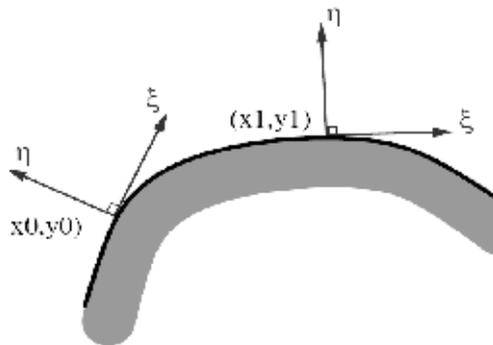


Fig. 1. An image contour and its moving vector basis (ξ, η) . Taken from (Tschumperle & Deriche, 2002).

This PDE is characterized by an anisotropic diffusive effect in the privileged directions ξ and η allowing a denoising of scalar image.

A limitation of this diffusion process (Eq. (4)) is its high dependance to the intrinsic quality of the original image and the impossibility to integrate prior information on the pattern to be restored if it can be characterized by particular data (orientation for example). Moreover, no characterization of the uncertainty/inaccuracy compromise can be made on the studied pixel, since the scale parameter is not directly integrated in the minimisation problem in which relies the common diffusion equations (Nordström, 1990). In this article we propose an original PDE directly integrating the scale parameter and allowing the taking into account of a priori knowledge on pattern to restore. We propose more particularly, to derive this PDE, to use a recent theory known as Extreme Physical Information (EPI) recently developed by Frieden (Frieden, 1998) and applied to image processing by Courboulay et al (Courboulay et al., 2002).

The second section of this article is dealing with the presentation of EPI and with the obtaining of the particular PDE. The third one presents a direct application to the presented diffusion process which may find applicability in robotics and automation. Section 4 shows the possible application of the developed method in a particular medical application: Enhancement of tagged cardiac MRI (Magnetic Resonance Imaging). Last part is dedicated to discussion.

2. EPI and Image Diffusion

2.1 EPI

Developed by Frieden, the principle of Extreme Physical Information (EPI) is aimed at defining a new theory of measurement. The key element of this new theory is that it takes into account the effect of an observer on a measurement scenario. As stated by Frieden (Frieden, 1996; Frieden, 1998), "EPI is an observer-based theory of physics". By observing, the observer is both a collector of data and an interference that affects the physical phenomenon which produces the data. Although the EPI principle brings new concepts, it still has to rest on the definition of information. Fisher information was chosen for its ability to effectively represent the quality of a measurement. Fisher information measure was introduced by Fisher in 1922 (Fisher, 1922) in the context of statistical estimation. In the last ten years, a growing interest for this information measure has arisen in theoretical physics. In his recent book (Frieden, 1998), Frieden has characterized Fisher information measure as a versatile tool to describe the evolution laws of physical systems; one of his major results is that the classical evolution equations as the Schrödinger wave equation, the Klein-Gordon equation, the Helmholtz wave equation, or the diffusion equation, can be derived from the minimization of Fisher information measure under proper constraint. Practically speaking, EPI principle can be seen as an optimization of the information transfer from the system under measurement to the observer, each one being characterized by a Fisher Information measure denoted respectively I and J . The first one is representative of the quality of the estimation of the data, and the second one allows to take into account the effect of the subjectivity of the observer on the measure. The existence of this transfer leads to create

fluctuations on the acquired data compared to the real ones. In fact, this information channel leads to the loss of accuracy on the measure whereas the certainty is increased.

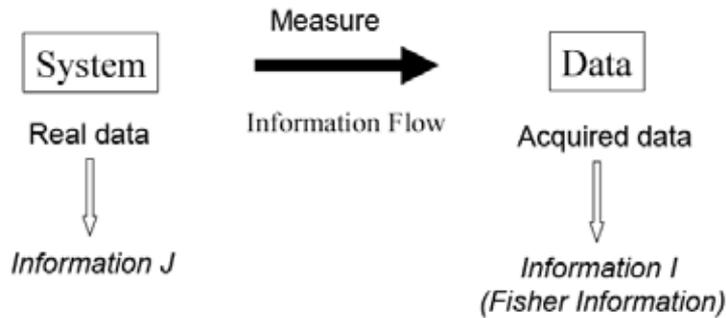


Fig.2. Fisher Information and measure

The goal of EPI is then to extremize the difference $I-J$ (i.e. the uncertainty/inaccuracy compromise) denoted K , called Physical Information of the system, in order to optimized information flow.

2.2 Application to image diffusion

Application to image diffusion can be illustrated by Fig. 3.

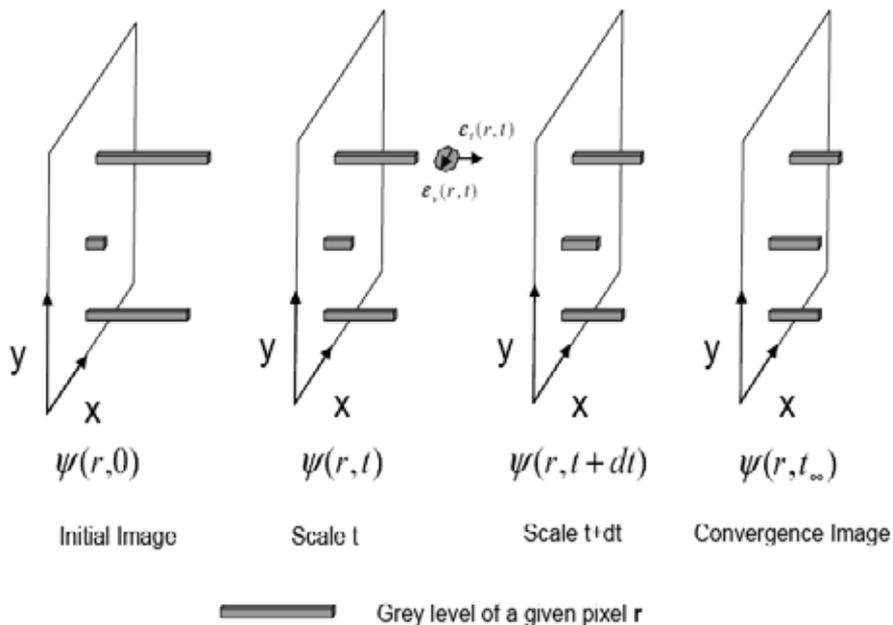


Fig. 3. Uncertainty/inaccuracy compromise and isotropic image diffusion. When parameter t tends to infinity, luminance of all pixels ($r=(x,y)$) of the corresponding image is the same and equal to the spatial average of the initial image.

As far as isotropic image diffusion is concerned, the uncertainty deals with the fluctuations of the grey level of a given pixel compared with its real value, whereas the inaccuracy deals with the fluctuations of the spatial localisation of a given pixel compared with the real one. The two different errors ($\varepsilon_r(t)$ and $\varepsilon_o(t)$) of Fig. 3 which are introduced all along the diffusion process are characterized by a measure of Fisher information. Intrinsic Fisher information J will be an integration of the diffusion constrained we impose on the processing.

Then, we can apply EPI to image diffusion process by considering an image as a measure of characteristics (as luminance, brightness, contrast) of a particular scene, and diffusion as the observer of this measure at a given scale. Extreme Physical Information K is then defined as follows (Frieden, 1998):

$$K(\psi) = \iint d\Omega dt \times \left[(\nabla - A)(\nabla - A)\psi^2 + \left(\frac{\partial \psi}{\partial t} \right)^2 - \psi^2 \right] \quad (5)$$

where \mathbf{A} is a potential vector representing the parameterizable constrain integrated within diffusion process. Extremizing K by Lagrangian approach leads to a particular diffusion equation given by :

$$\frac{\partial \psi}{\partial t} = (\nabla - A)(\nabla - A)\psi \quad (6)$$

with $\psi(x, y, 0) = \psi_0$

As a consequence, thanks to the possible parameterization of \mathbf{A} , it is possible to take into account particular characterized patterns to preserve from the diffusion process.

2.3 About \mathbf{A}

The \mathbf{A} potential allows to control the diffusion process and introduce some prior constrains during image evolution. For instance, if no constrain are to be taken into account, we set \mathbf{A} as vector null and then Eq. (6) becomes:

$$\frac{\partial \psi}{\partial t} = \nabla \cdot \nabla \psi = \Delta \psi \quad (7)$$

with $\psi(x, y, 0) = \psi_0$

which is the well known heat equation characterized by an isotropic smoothing of the data processed. In order to enlarge the possibility given by Eq. (6), the choice we make for \mathbf{g} is based on the fact that Eq. (6) allows a weighting of the diffusion process with the difference of orientation between the local calculated gradient and \mathbf{A} . More precisely, to explain the way \mathbf{A} is practically implemented, let consider Fig. 4.

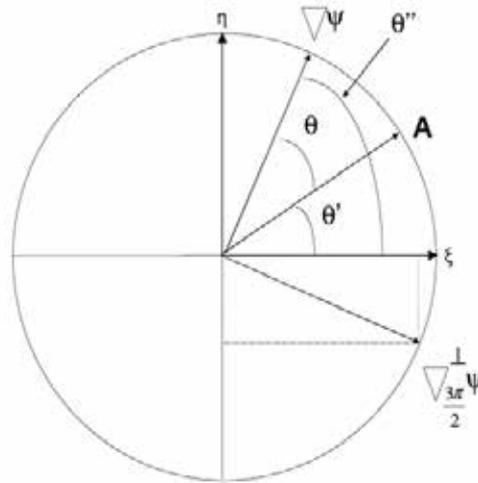


Fig. 4. Local geometrical implementation of \mathbf{A} in terms of the local gradient.

The expression of the local gradient in terms of θ'' is, considering Fig. 4 :

$$\nabla \psi = \begin{pmatrix} \|\nabla \psi\| \cos \theta'' \\ \|\nabla \psi\| \sin \theta'' \end{pmatrix} \quad (8)$$

and an expression of \mathbf{A} in terms of θ' is :

$$\mathbf{A} = \begin{pmatrix} \|\nabla \psi\| \cos \theta' \\ \|\nabla \psi\| \sin \theta' \end{pmatrix} \quad (9)$$

Norm of \mathbf{A} is imposed in order to make it possible the comparison with the gradient. To this point, the most interesting expression of \mathbf{A} would be the one in terms of θ , which represents the difference angle between \mathbf{A} and the local gradient. If we made so, using trigonometrical properties and noticing that $\theta = \theta'' - \theta'$, we obtain a new expression for \mathbf{A} :

$$\mathbf{A} = \begin{pmatrix} \|\nabla \psi\| (\cos \theta'' \cos \theta + \sin \theta'' \sin \theta) \\ \|\nabla \psi\| (\sin \theta'' \cos \theta - \cos \theta'' \sin \theta) \end{pmatrix} \quad (10)$$

Eq. (10) could be simplified by integrating the vectorial expression of the local gradient (Eq. (8)) :

$$A = \nabla \psi \cos \theta + \nabla^{\perp}_{\frac{3\pi}{2}} \psi \sin \theta \quad (11)$$

From Eq. (11), we could then derive a general expression for \mathbf{A} considering it as a vectorial operator :

$$A = \nabla \cdot \cos \theta + \nabla^{\perp}_{\frac{3\pi}{2}} \cdot \sin \theta \quad (12)$$

with θ the relative angle between \mathbf{A} and $\nabla \psi$ for a given pixel and ∇^{\perp} the local vector orthogonal to ∇ (Fig. 4). This expression only represents the way it is possible to reexpress \mathbf{A} by an orthogonal projection in the local base (ξ, η) (Fig. 1) . Considering it, Eq. (6) becomes:

$$\frac{\partial \psi}{\partial t} = \frac{\partial^2 \psi}{\partial \eta^2} (1 - \cos \theta) + \frac{\partial^2 \psi}{\partial \xi^2} (1 - \cos \theta) \quad (13)$$

One can notice on Eq. (13) that when angle $\theta = 0$ (i.e. \mathbf{A} and $\nabla \psi$ are colinear), the studied pixel will not be diffused for $\frac{\partial \psi}{\partial t} = 0$. On the contrary, a nonzero value of θ will lead to a weighted diffusion of the considered neighbourhood of the pixel (Eq. (13)). As a consequence, by imposing local θ values, it is possible to preserve particular patterns from the diffusive effect within the processed image.

3. Application to oriented pattern extraction

In this section, we present results obtained on simple images in order to show the restoration and denoising potential of the method. For practical numerical implementation, the process of Eq. (13) is discretized with a time step τ . The images $\psi(t_n)$ are calculated, with Eq. (13), at discrete instant $t_n = n\tau$ with n the number of iterations of the process.

Let first consider an image showing vertical, horizontal, and $45^\circ \pm$ -oriented dark stripes on a uniform background (Fig. 5).

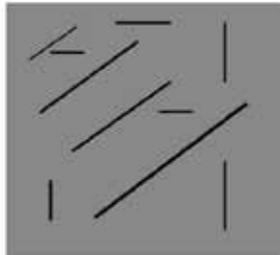


Fig. 5. Image 1: Dark stripes with various orientations on a uniform background.

Considering Eq. (13), by imposing two possible orientations for \mathbf{A} (135° , 325°) which corresponds to the gradient orientations of the diagonal stripes, one could expect to preserve them from isotropic diffusion. Diffusion results are presented Fig. 6.

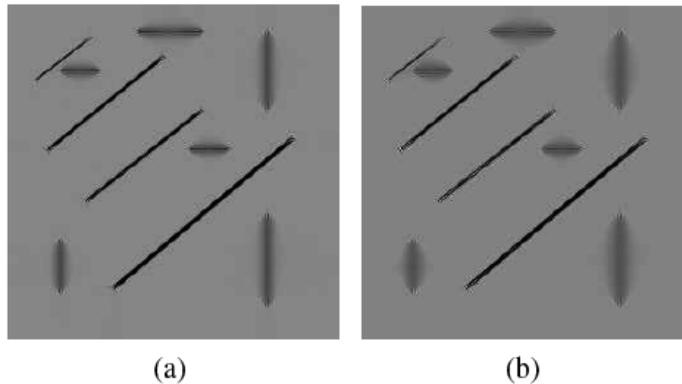


Fig. 6. Diffusion of "Image 1" (Fig. 5) for (a) $n=100$ and (b) $n=200$. \mathbf{A} is chosen in order to preserve only diagonal stripes from isotropic diffusion process. Time step τ is fixed to 0.2.

As one was expected it, the vertical and horizontal dark stripes in diffused images tend to disappear whereas the diagonal stripes are preserved all along the diffusion process.

Let now consider a noisy simple grid diagonally oriented corrupted by a Gaussian noise of standard deviation set to 0.3.

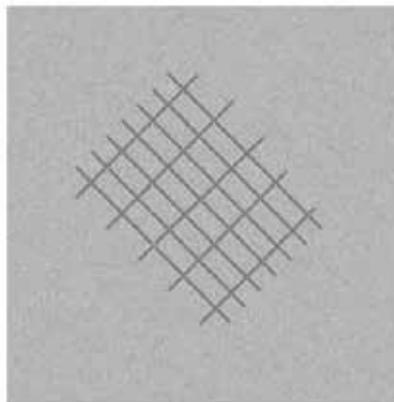


Fig. 7. Image 2: Noisy diagonally oriented grid (Gaussian noise). PSNR (calculated with the non corrupted version of the grid as reference) is equal to 68 dB.

If we apply the same diffusion process of Eq. (10) to this noisy simple grid imposing this time four possible orientations for \mathbf{A} corresponding to the four possible gradient orientations of the grid, it is then possible to show the denoising effect of the diffusion process (Fig. 8).

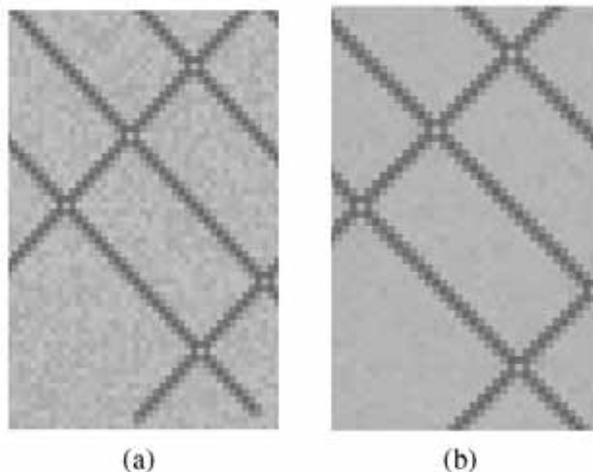


Fig. 8. Diffusion of "Image 2" (a) (Fig. 7) for (b) $n=50$. As one can notice, the grid itself is preserved from the diffusive effect of Eq. (3) whereas noise is iteration after iteration removed. Time step τ is fixed to 0.2.

As intended, the grid itself is not diffused at all and the increase of the Peak Signal to Noise Ratio (PSNR) from 68 dB to 84 dB, shows that the added Gaussian noise is removed iteration after iteration.

After these first experimental results obtained on ad hoc images, we now propose a practical medical application which main objective is automation of detection and tracking of a particular pattern within image sequences.

4. Application to enhancement of tagged cardiac MRI

4.1 Presentation of the problematic

The non invasive assessment of the cardiac function is of major interest for the diagnosis and the treatment of cardiovascular pathologies. Whereas classical cardiac MRI only enables radiologists to measure anatomical and functional parameters of the myocardium (mass, volume...), tagged cardiac MRI makes it possible to evaluate local intra-myocardial displacements. For instance, this type of information can lead to a precise characterization of the myocardium viability after an infarction. Moreover, data concerning myocardium viability makes it possible to decide of the therapeutic : medical treatment, angiopathy, or coronary surgery and following of the amelioration of the ventricular function after reperfusion.

The SPAMM (Space Modulation of Magnetization) acquisition protocol (Zerhouni et al,1988) we used for the tagging of MRI data, displays a deformable 45-degrees oriented dark grid which describes the contraction of myocardium (Fig. 9) on the images of temporal Short-Axis (SA) sequences. The temporal tracking of this grid can enable radiologists to evaluate the local intramyocardial displacement. Nevertheless, tagged cardiac images present peculiar characteristics which make it difficult to analyze. More precisely, images

are of low contrast compared with classical MRI, and their resolution is only of approximately one centimeter. Numerous studies were carried out concerning the analysis of the deformations of the grid of tag on SA sequences (see (Petitjean et al, 2005; Axel et al, 2007) for a complete overview) but all have in common the necessary enhancement of tagged cardiac images. As far as no technic as allowed to develop a "Gold Standard" method, we then propose to use the presented selective diffusion scheme to enhance the oriented grid of tags (Fig. 9) in order to make easier its temporal segmentation.



Fig. 9. SA tagged MRI of the Left Ventricle extracted from a sequence acquired between end diastole and end systole.

4.2 Processing and results

Considering Fig. 9 and Eq. (10), a solution to the problem of enhancement of the grid of tags is to impose particular orientations for \mathbf{A} , considering the fact that the gradients to be preserved within tagged cardiac MRI are well known and correspond to the orientations of the grid-of-tag ones (i.e. 45° , 135° , 225° , 315°). Others solutions making a better computation of \mathbf{A} can be imagined and are at the moment under development. Nevertheless, whereas this rough solution is not the best, it is sufficient for what we want to demonstrate in this chapter.

Considering this parameterization of Eq. (10), Fig. 10 shows results obtained on the first image of a classical tagged cardiac MRI sequences (Fig. 9). For a better appreciation of the proposed results, only one direction (45°) of the grid has been taken into account during the selective diffusion process of Eq. (10)

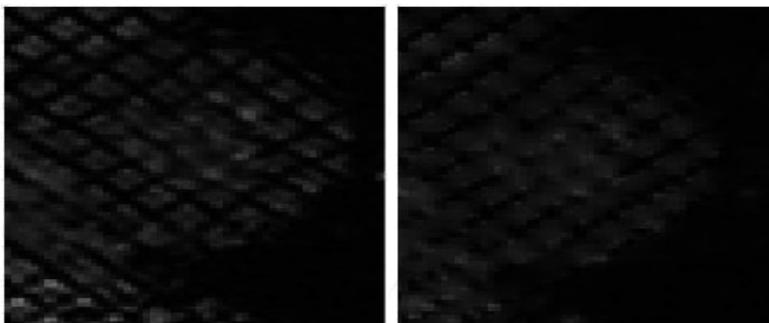


Fig. 10. Preservation of the 45° -oriented tag (right) on the initial image (left) of a tagged sequence.

As we can see in Fig.10, the diffusion process makes possible the fading of noisy artifacts, and non-45°-oriented lines. This result can be generalized to other direction of the grid.

At last, the enhancement effect of the selective diffusive process can only be seen on the evolution of a particular intensity profile extracted from original and enhanced tagged cardiac image. Indeed, Fig. 11 shows that selectiveness of Eq. (10) makes it possible the smoothing of all data except tag profiles.

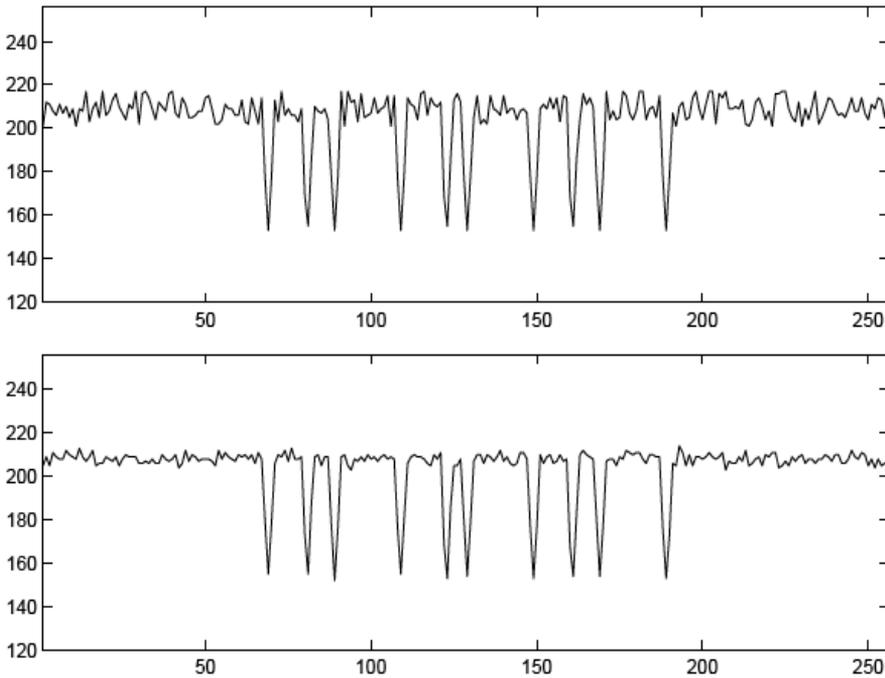


Fig. 11. Up: Original intensity profile extracted from original tagged cardiac MRI (orthogonally to orientation of the grid of tags). Bottom: Enhanced image thanks process of Eq. (10) with parametrization presented in previous subsection. As one can notice, selectiveness of Eq. (7) makes it possible the smoothing of all data excepttag profiles.

5. Conclusion

In this chapter an original diffusion method, based on the use of a particular PDE (Eq. (10)) derived from EPI theory, has been presented as an alternative to classical PDE's based approach. It has been shown that the integration of the potential vector \mathbf{A} within the formulation of this PDE makes the integration, within the diffusion scheme, of particular constrains possible. This has been assimilated to integration of selectiveness within classical isotropic diffusion process. Examples on ad hoc images have been presented to show the potential of the presented method in the areas of denoising and extraction of oriented patterns. A particular application has also been proposed in the case of enhancement of tagged cardiac MRI. This method may find applicability in others areas like vision in robotics for instance.

6. References

- Alvarez, L. & Morel, J. (1992) *Image selective smoothing and edge detection by nonlinear diffusion. SIAM Journal of Numerical Analysis*, 29(3):845–866, 1992.
- Axel, L., Chung, S., and Chen, T. (2007). Tagged MRI analysis using gabor filters. In *Biomedical Imaging: From Nano to Macro, 4th IEEE International Symposium on*, 684–687.
- Catte, F.; Coll, T.; Lions, P. & Morel, J (1992). *Image selective smoothing and edge detection by nonlinear diffusion. SIAM Journal of Applied Mathematics*, 29(1):182–193.
- Courboulay, V., Ménard, M., Eboueya, M., & Courtellemont, P. (2002). Une nouvelle approche du filtrage linéaire optimal dans le cadre de l'information physique extrême. In *RFIA 2002*, pages 897–905.
- Deriche, R. & Faugeras, O. (1996). Les EDP en traitements des images et visions par ordinateur, *Traitement du Signal*, 13(6) 551-578.
- Fisher, R. (1922). *Philosophical Transactions of the Royal Society of London*, 222:309.
- Frieden, B. (1996). Fisher information as a measure of time. *Astrophysics and Space Sciences*, 244:387–391.
- Frieden, B. (1998). *Physics from Fisher Information*. Cambridge University Press.
- Geman, S. & Reynolds, G. (1992). Constrained restoration and the recovery of discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):367–383.
- Komprobst, P.; Deriche, R. & Aubert, G. (1997). Image coupling, restoration and enhancement via PDE's. *Proceedings of the International Conference on Image Processing*, volume 2, pages 458–461, October.
- Nordstrom, N. (1990). Biased anisotropic diffusion—a unified regularization and diffusion approach to edge detection. *Image and Vision Computing*, 8(4):318–327.
- Nitzberg, M. & Shiota, T. (1992). Nonlinear image filtering with edge and corner enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):826–833.
- Perona, P. & Malick, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.
- Petitjean, C., Rougon, N., and Cluzel, P. (2005). Assessment of myocardial function: A review of quantification methods and results using tagged MRI. *Journal of Cardiovascular Magnetic Resonance*, 7(2):501–516.
- Terebes, R., Laviaille, O., Baylou, P., & Borda, M. (2002). Mixed anisotropic diffusion. In *Proceedings of the 16th International Conference on Pattern Recognition*, volume 3, pages 1051–1061.
- Tschumperle, D. & Deriche, R. (2002). Diffusion PDEs on vector-valued images. *Signal Processing Magazine, IEEE*, 19:16–25.
- Tschumperle, D. & Deriche, R. (2005). Vector-valued image regularization with PDE's: A common framework for different applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:506–517.
- Weickert, J. (1995). Multiscale texture enhancement. In *Computer Analysis of Images and Patterns*, 230–237.
- Weickert, J. (1998). *Anisotropic Diffusion in image processing*. Teubner-Verlag, Stuttgart.
- Whitaker, R. & Pizer, S. (1993). A multi-scale approach to nonuniform diffusion. *CVGIP:Image Understanding*, 57(1):99–110.
- Zerhouni, E., Parish, D., Rogers, W., Yang, A., & Shapiro, E. (1988). Human heart : tagging with MR imaging - a method for noninvasive assessment of myocardial motion. *Radiology*, 169(1):59–63.

The robot voice-control system with interactive learning

Miroslav Holada, Martin Pelc
*Technical University of Liberec
Czech Republic*

1. Introduction

Nowadays, robots are penetrating human lives and carry out many tasks that would have been impossible for machines just few decades ago. One can usually get an “end user” narrow- or single-purpose robotic solution for a specific task with a user interface providing control within the scope of the given task. Another possibility is to get a more general-purpose robot and design a custom control system for peculiar tasks. The latter usually involves programming using the manufacturer's rudimentary robot movement interface and requires deeper knowledge of robotics, computer sense, control systems etc.

Providing a general-purpose control system for general-purpose robots which wouldn't require most of the aforementioned knowledge poses a challenge. It would make robotics appealing to wider audience and possibly speed up development and application of robots. Such system needs to have some “learning” capabilities as well as a friendly user interface for task “teaching”.

The goal of our work described in this capture is a PC-software-based interactive system for general-purpose robot voice-control. This paper describes the designed prototype, its structure and the dialogue strategy in particular. The interactive control of robots could be used in special situations, when a robot is working in dangerous areas and no programming beforehand is possible. It could also be used in a situation when supervised learning for robot's later autonomous operation has to be done, without knowledge about the robot's programming language.

Generally, the robots are actuated by sets of control commands, sometimes by a manual control interface (such as touchpad or joystick). The operator has to know the control commands, syntax rules and other properties necessary for successful robot control. The proposed system tries to simplify this robot programming and make it more user-friendly and easy to use. The system offers commands like “move left” or “elevate arm” that are translated and sent into the corresponding device (robot).

2. Project features

The project is based on a former research. The research involved a voice-control dialog system, speech recognition, vocabulary design and speech synthesis feedback for user

command confirmation. Together with a scene manager and a digital image processing module, it forms the core of the control system as shown in figure 1.

The key feature of the system is that it can “learn” a series of commands to autonomously perform certain tasks using the robot. Digital cameras will be used to navigate in the robot's working space. Supported by computer vision algorithms, the system should be able to find objects of interest (and to keep track of them) on the scene including the robot itself and allow the objects to be referred to by user's commands. The main feature is that a user is not required to have knowledge about robot programming, computer vision, etc.

The system should also offer a straightforward robot movement control via either verbal commands or graphical user interface (GUI).

The system is being developed as a pure-software solution hosted on the Windows platform. The components of the system are described below.

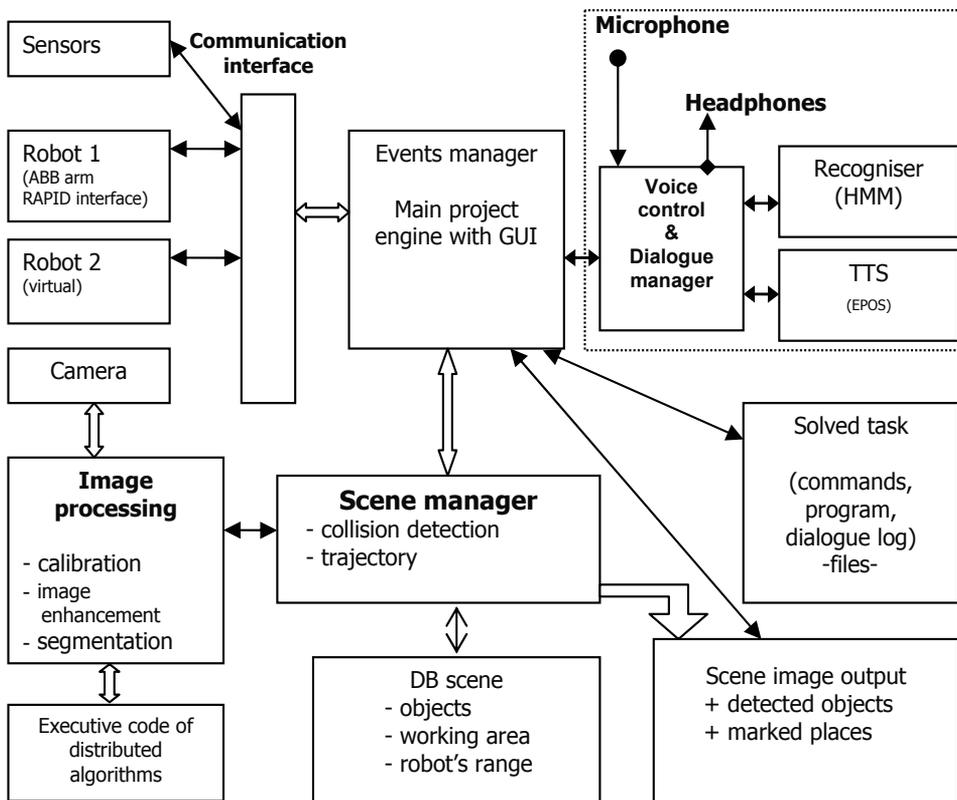


Fig. 1. Functional layout of the system.

2.1 Scene manager

The scene manager forms a connection between the main program (engine) and the image processing part. It actually controls the image processing module and initiates image acquisition and processing. Using the processed image data, it updates the scene database, keeps track of objects found on the scene and provides the scene object and image data to

the main engine. It is also aware of the robot's coordinate system and plans the robot's movement when requested by the engine.

The database itself consists of two types of data. It contains the list of parametrized objects detected on the scene as well as the robot calibration data. The latter allows mutual image-space to robot-space coordinate translation which is used in robot navigation. Each object detected on the scene is internally represented as a data object (class instance), all the objects are stored in a dynamic list. Some of the attributes are: a unique object identifier, object's shape descriptor, central point coordinates, bounding rectangle etc. Such data allows smooth object manipulation and serves as a base for object collision avoidance along the manipulation trajectory.

The scene manager also combines unprocessed camera image with scene data to highlight detected objects and to present them to the user via a GUI as shown in figure 2. The user has a view of the computer's scene understanding and may correctly designate objects of interest in his or her commands.

Being in its early stages, the project currently works only with 2D data and relies on the user's z-axis navigation aid. The system is expected to incorporate a second camera and 3D computer vision in the future to become fully 3D aware.

2.2 Speech processing

The voice interface between an operator and the controlled process is provided by a speech recogniser and a text-to-speech synthesis (TTS) system (both for Czech language). The TTS synthesis system named EPOS was developed by URE AV Prague. It allows various male or female voices with many options of setting (Hanika & Horak, 1999).

The speech recognition is based on a proprietary isolated word engine that was developed in previous projects (Nouza, 2000). The recogniser is speaker independent, noise robust, phoneme based with 3-state HMM (Hidden Markov Models) and 32 Gaussians. It is suitable for large vocabularies (up 10k words or short phrases) and allows us to apply various commands and their synonyms (Nouza & Nouza, 2004).

Both voice components are built into a distributed system named DUNDIS (Holada, 2004). The advantage of this solution is the fact that the designed system needs to incorporate only a relatively simple software client. This client sends speech data to the recognition server where speech recognition is executed. The TTS engine can work separately but in this case it is linked to recognizer due to echo cancellation problem. The unwanted acoustic feedback (meaning that the computer "hears" and recognizes what it "speaks") is eliminated by half-duplexing the communication (it either "speaks" or "listens" but not both at the same time).

2.3 Image processing

The robot's working area is captured by a colour high-resolution digital camera (AVT Marlin F-146C, 1/2" CCD sensor). The camera is placed directly above the scene in a fixed position. We implemented a simple interactive method to synchronize the robot's coordinate system (XY) and the camera's one using pixel units and prepare modifications to compensate geometric distortions introduced by camera lens.

The picture 2 shows the overall view of the test workplace. The camera is placed above the scene and is partially visible on the top of the picture. The working scene is composed of the

robot's surrounding, most notably the white desk with disks that are placed on ribbons to prevent robot's tool damage (crashing directly into the desk).

Digital image processing methods are placed in a library which is served by the scene manager with the object database. The figure 3 shows the circular object detection using the reliable Hough transform (HT). HT is commonly used for line or circle detection but could be extended to identify positions of arbitrary parametrizable shapes. Such edge-based object detection is not too sensitive to imperfect input data or noise. Using a touch-display or verbal commands it is possible to focus the robot onto a chosen object (differentiated by its color or numbering) and then tell the robot what to do.

So far the system supports detection of basic geometric shapes (circle, rectangle) and basic colors.

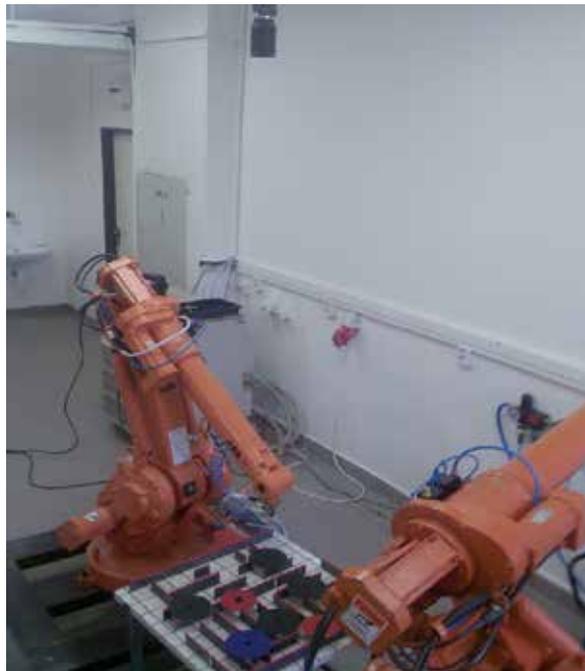


Fig. 2. Overall view of the test workplace with robot, working scene and the camera.

2.4 Robots description

For the purpose of debugging the system, a virtual robot device was designed which behaved like the real one but worked only as a graphical computer simulation. For field tests a real robot had to be used and we chose an industrial robot typically used in robotics lessons which was available to us.

The prototype system uses a compact industrial general-purpose robotic arm (ABB IRB 140). The robot is a 6-axes machine with fast acceleration, wide working area and high payload. It is driven by a high performance industrial motion control unit (S4Cplus) which employs the RAPID programming language. The control unit offers extensive communication capabilities - FieldBus, two Ethernet channels and two RS-232 channels. The serial channel

was chosen for communication between the robot and the developed control system running on a PC.

The robotic control software module simplifies the robot use from the main engine's point of view. It abstracts from the aspects of physical communication and robot's programming interface. It either accepts or refuses movement commands issued by the core engine (depending on command's feasibility). When a command is accepted, it is carried out asynchronously, only notifying the engine once the command is completed.

Industrial robots have their own sophisticated control systems which allow arbitrary task programming. In our case, the RAPID-based ABB control system proved to be not very suitable for applications which require direct movement control by the computer program, not just the RAPID program stored in the control system.

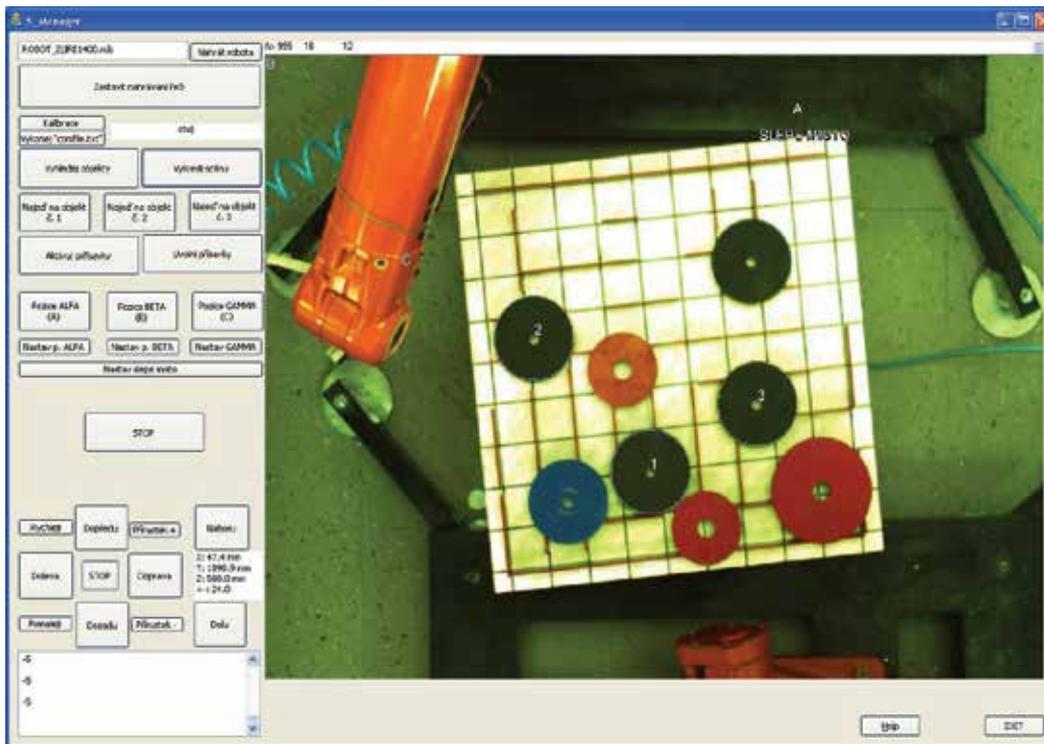


Fig. 3. The system's GUI with the scene view and highlighted objects.

2.5 Distributed computing

Most of the system's modules are developed and run on a standard PC to which the robot is connected. Since some of the software modules require significant computational power, the system's response time was far from satisfactory when the whole system ran on a single computer. Therefore, the most demanding computations (namely the object recognition and the voice recognition) were distributed to other (high performance) computers via network (TCP connections).

The solution with distributed components is advantageous especially for research and debugging. If any part of the system crashes then after its restart the other parts are quickly

reconnected without need to reload and initialize them. Today's local networks are fast enough so any introduced transfer delays are insignificant.

3. Dialogue strategy

The dialogue scenario contains four vocabularies. The first is composed of simple basic control commands like "move up", "stop" or "take it". They are necessary for basic robot control and many synonyms may be defined for each action. The second group contains unused words and short phrases. It is the biggest group (vocabulary) with tens of thousands items and we can define names of new actions from this group. The names of learned tasks defined by user are the third group of words in the dialogue scenario. The fourth group contains titles for built-in activities like robot calibration, learning initialization or defining a new name for the most recent operation. Items from this group cannot be used in newly defined commands, though.

The process of learning a new function starts when the operator says the built-in command "beginning of learning". Any known commands issued afterwards are memorized until "end of learning" command is given. A newly defined task has to be given a name. The new name should consist of one previously unused vocabulary word or unused combination of several words (for example "take it + and + move up"). This simple strategy allows to define new robot's tasks just using voice, without keyboard or mouse. It is possible to use any previously defined tasks to compose a new and more complex task.

The picture 3 shows a captured screen of the designed system's GUI. There are buttons representing the basic voice commands on the left side. The majority of the screen is taken up by camera view showing highlighted significant points and detected objects.

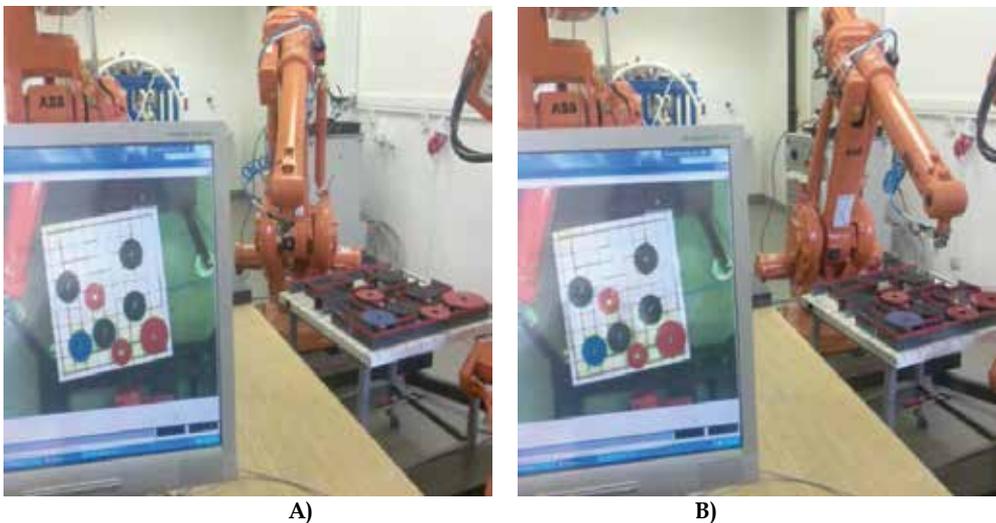


Fig. 4. Scene capture and object detection: A) initial shot with arm outside of view, B) arm carrying out a task

During the dialog some basic logic rules have to be respected. When the objects on the working scene are being analysed, the robot's arm is moved out of scene first (fig. 4) to

avoid object confusion and occlusion. The system stores the last detection result in the scene database.

The whole dialog system is event-driven. We can categorize the events into three fundamental branches: operator events, scene manager events and device events.

3.1 Operator events

Operator events usually occur in response to operator's requests. For example, commands which are supposed to cause robot's movement, object detection, new command definition or detection of a new object. This kind of event can occur at any time, but the dialog manager has to decide if it was a relevant and feasible request or if it was just a random speech recognition error.

Although the acoustic conditions in robotic applications usually involve high background noise (servos, air-pump), the speech recognizer works usually with over 90% recognition score. If the operator says a wrong command or a command out of context (for example, the operator says "drop" but the robot doesn't hold anything) then the event manager asks him or her for a feasible command in the stead of the nonsensical one.

3.2 Scene manager events

This sort of event occurs when the scene manager detects a discrepancy in the scene. For example when the operator says "move up" and the robot's arm moves all the way up until the maximum range is reached. When this happen a scene event is generated and the system indicates that the top position was reached.

Other scene event occurs when the operator wants to take up an object, but the system does not know which one because of multiple objects detected on the scene. This event generates a query to the operator for proper object specification.

3.3 Device events

These events are produced by external sensors and other components and devices connected to the system. They are processed in the event manager where corresponding action is taken. The response manifests itself in the form of a request for the operator, or more often causes a change in robot's behaviour.

The difference between scene manager events and device events is that scene events are generated by the system itself (based on a known scenario, robot geometry, object shape and position). They are computed and predictable. On the other hand, device events' time cannot be exactly predicted before they actually happen.

3.4 Examples of dialog

For a simpler robot orientation and navigation the positions on the scene are virtualized. They are named after the Greek letters like "Position α " (alpha) or "Position β " (beta). These virtual positions may be redefined to suit the operator's needs. A blind-area may also be defined and it is completely omitted from any image processing and anything in this area is completely ignored.

As an example (see figure 3.) the robot can grab all the black disks and move them to some other place on the scene. This place is defined as "Position alpha" and the "blind area" is set up on the same coordinates. After this the operator starts an example dialog:

"Start recording new command." ...this is operator's command (italic text)...

"I'm recording" ...the system says (bold text)...

"Search black disks"

"I'm searching ... Four disks were found"

"Move on first"

"I'm moving ... Done"

"Take it."

"Ok"

"Move on position alpha."

"I'm moving ... Done"

"Put it"

"Ok"

"Stop recording."

"I stop the recording. Please, say new command"

"Search" "Disks" "Done"

"New command is entered and named: Search disks. Is it right?"

"Yes"

...now, the newly defined command may be used...

"Repeat command"

"Enter command"

"Search disks"

"OK"

...now the system repeats the command until no more disks are found...

"No object found. Repeating done."

...now all the disks on the scene are transported into position alpha...



Fig. 5. a) The initial scene.



b) The robot grabbing a target disc.

The robot finds the remaining three disks and puts them into the selected area. If no disk is found the robot interrupts the execution of the given command and waits for a new command. This is shown in figures 5 and 6.

The figure 7 shows an operation where the system finds a “small red disk”, grabs it and puts it onto a chosen “black disk”. The navigation of the arm relies only on the image processing results.

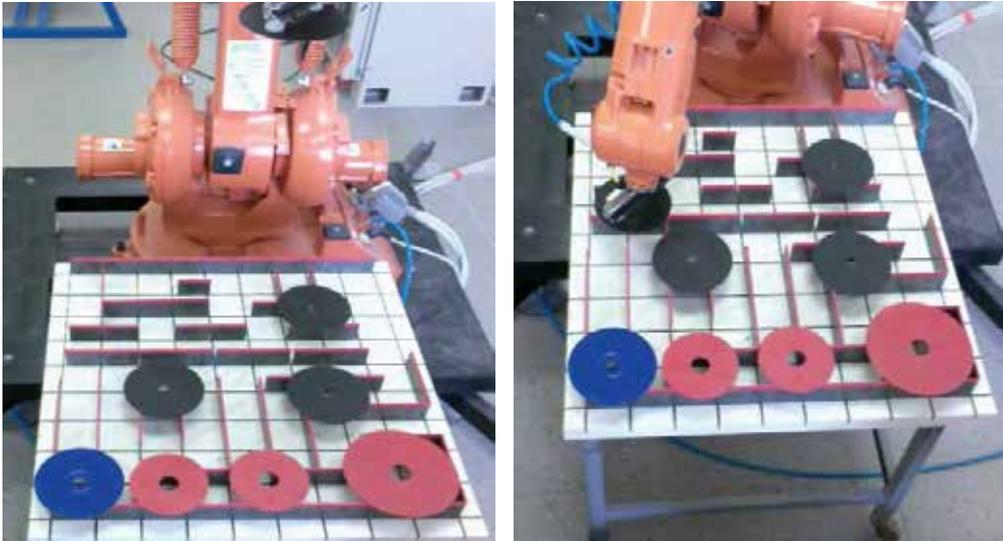


Fig. 6. The robot lifting a disk, moving it around and placing it in a desired position.

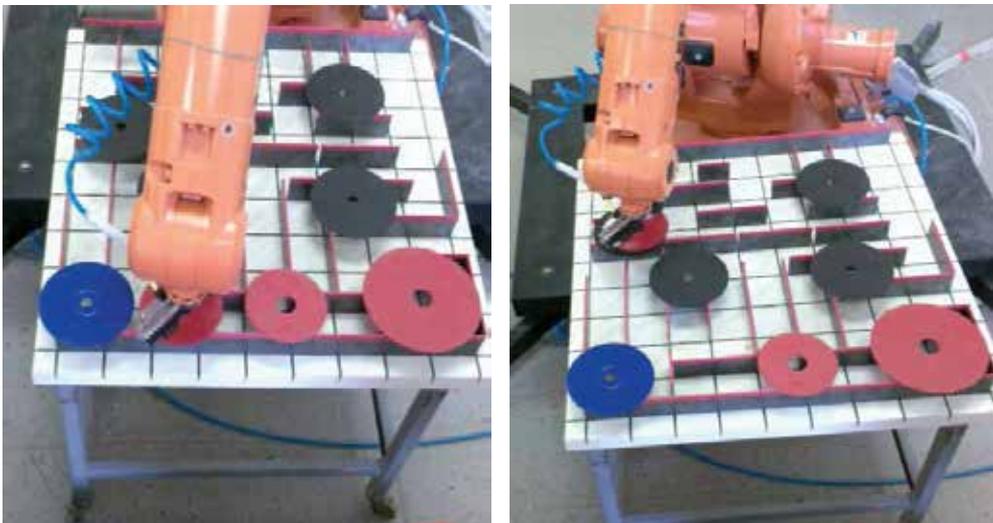


Fig. 7. The robot grabbing another disc and stacking up a pile.

4. Conclusion

The system is especially usable as an accessory robot control interface for assistant and second-rate operations. The designed prototype cooperates with only one specific industry

robot (ABB) so far but the robotic control module may easily be extended to support other robots (Katana, mobile robots, etc.) as well.

The system offers robot control and robot task programming even to people without explicit programming knowledge. It is sufficient for the operator to know the Czech voice interface of the presented system.

The system is able to "memorize" issued commands and reproduce tasks. The designed dialogue strategy was verified using a real robot in real conditions.

The fusion of computer vision, voice recognition and robot control is quite challenging but it looks promising. The development itself is rather complicated as it requires knowledge from many different areas of science.

Employed computer vision greatly simplifies robot navigation as the user actually sees the system's "understanding" of the scene. This also allows for a much better utilization of voice control.

Contemporary computer hardware seems to be adequate for the demanding operations involved, but the system may still require distributed computing (to achieve reasonable response times and user comfort).

The presented prototype serves as a base for further development. The system is planned to use 3D vision as well as arbitrary object detection and description to become fully 3D-aware and needing as little user aid as possible.

5. Acknowledgement

This work has been supported by the Grant Agency of the Czech Republic (grant no. 102/07/P455) and the internal grant IG FM TUL 2007/002.

6. References

- Nouza J. (2000). A Czech Large Vocabulary Recognition System for Real-Time Applications. In *Text, Speech and Dialogue* (eds. Sojka, Kopecek, Pala) Springer-Verlag, Heidelberg
- Nouza, J., Nouza, T. (2004). A Voice Dictation System for a Million-Word Czech Vocabulary. In: *Proc. of ICCCT 2004, Austin, USA*
- Holada, M. (2004). The experiences and usability of distributed speech recognition system DUNDIS. In: *Proc. of 14th Czech-German Workshop "Speech Processing", Prague, Czech Republic*, pp. 159-162,
- Hanika, J, Horak, P. (1999). Text to Speech Control Protocol. In: *Proc of the Int. Conf. Eurospeech'99, Budapest, Hungary*
- Sonka, M., Hlaváč, V., Boyle, R. D. (1998). *Image Processing, Analysis and Machine Vision*. PWS, Boston, USA
- Cerva, P., Nouza, J. (2007). Design and Development of Voice Controlled Aids for Motor-Handicapped Persons, In: *Conference of the International Speech Communication Association (Interspeech 2007), ISSN: 1990-9772*

Intelligent Detection of Bad Credit Card Accounts

Yo-Ping Huang¹, Frode Eika Sandnes², Tsun-Wei Chang³ & Chun-Chieh Lu⁴

¹*Dept. of Electrical Engineering, National Taipei University of Technology, Taiwan*

²*Faculty of Engineering, Oslo University College, Norway*

³*Dept. of Computer Science and Information Eng., De Lin Institute of Technology, Taiwan*

⁴*Department of Computer Science and Engineering, Tatung University, Taiwan*

1. Overview

The effective management of credit risk is a critical component of risk management and essential to the long-term success of any banking organization. Credit-risk management practices vary considerably among organizations. Bad accounts affect the credit reputation of the cardholder and cause issuers to lose billions of dollars each year. However, bad debts problems are complex, full of uncertainty and can only be resolved through manual intervention. This study proposes an intelligent model for the early and accurate detection of bad accounts based on fuzzy logic (FL) and back propagation neural networks (BPN). Experimental results verify the effectiveness of the proposed strategy.

2. Introduction

The number of credit cards issued in Taiwan has increased rapidly since 1992. In 2004 the number of cards totalled approximately 44,180,000 cards, and the amount of revolving credit had reached NT\$457,900 million (new Taiwan dollars) (Central Bank of China, 2008). Due to the over-issue credit cards, the amount of bad debt has reached NT\$21,400 million. Cardholders with financial difficulties may be unable to keep up with payments. The card issuing banks subsequently may risk financial losses. According to the 2004 annual report of bad debt amount of revolving credit from Taiwan Financial Supervisory Commission (FSC) (Taiwan Financial Supervisory Commission, 2008) reveals that many banks suffer large and the losses are characterized by a year-on-year growth of 10.2% (Lin, 2005).

The FSC began to pay attention to the problem to prevent the Taiwanese credit card market from repeating the South Korea financial crisis (The International Commercial Bank of China, 2008). Bad credit card accounts are classified into two types: One type covers bad debt accounts that have bad payment records. It has a potential risk to regain credit amount. The other is called a charge-off account, where customers show no willingness to pay the debt, and the issuer will have to write off the debt as a loss.

Bad accounts not only result in losses for the issuers but also affect the reputation of cardholders which is maintained by the Taiwan Joint Credit Information Center (JCIC) (Taiwan Joint Credit Information Center, 2008). The JCIC will store information about charge-off accounts for seven years. Card issuers are responsible for reporting bad accounts to the JCIC and once a customer has a bad credit entry in the JCIC it may be more difficult, or impossible, to borrow money.

The process of detecting bad accounts is fraught with difficulties. The problem of bad account detection is complicated, uncertain, and is dependent on the operation environment. Many bad account records are not caught by current detection systems.

The primary objective of this study is to discover important credit card patterns that can be used for detection. This is achieved by soliciting good debt, bad debt, and charge-off records from a credit card company database. An intelligent fuzzy-neural inference system (Chakraborty, 2001; Huang et. al., 2007) is utilized to construct a better prediction model.

3. Background

A credit card allows a cardholder with a credit limit to purchase goods and services to obtain cash disbursements on credit offered by a third party, for which a cardholder is subsequently billed (National Credit Card Center of R.O.C, 2008). There are six parties involved in a credit card transaction (ShopSite, 2008), namely *cardholders*, *issuers*, *merchants*, *acquirers*, i.e., a bank or other financial institution that approves a merchant for accepting credit cards, and then collects the merchant's online payments, *associations*, e.g., Visa and MasterCard are associations of member banks and financial institutions (American Express, Discover, and Diner's Club are single corporations that issue their own cards), *payment gateways*, i.e., a company that provides an interface between the Internet and the secure banking networks. A payment gateway authenticates the parties involved and acts as a channel for moving credit card transactions from a store to a payment processor.

First, the customer applies for a credit card from the issuer (Cardservice International, 2008; Indiana University Office of the Treasurer, 2008). A bank or other financial institution issues the credit card according to the applicant's credit status, income and reimbursement ability. Then, the customer selects the desired products or services and pays a merchant using the card, whereupon the merchant's credit card terminal sends the request to the merchant's payment gateway. The information includes the store's identification information, the customer's credit card information, and the amount. The payment gateway checks its database to see which acquirer the store uses. The gateway sends the transaction information to the acquirer's payment processor. Then, the payment processor examines the customer's credit card number to determine the issuer and forwards the customer information and amount. Next, the issuer checks if the information is valid and if there is enough credit to cover the transaction. If ok, the issuer sends an authorization code or declined transaction message back to the payment processor which again sends the authorization message back to the payment gateway. Next the store issues a receipt to the customer and the issuing banks settle funds to the acquiring bank. The acquirer deposits the

money in the merchant's local account, minus the discount rate and transaction fee. The issuer's billing system computes cardholder's transactions, fee, and interest on cycle day and sends a bill to notify cardholder to pay within the due-date.

3.1 Credit Card Risk Management

Some banks are improving the credit decision-making process by means of credit scoring technologies that often are based on expert systems that can adapt to changes in the economy or within specific customer segments. If a customer scores low on the scoring system their application will be rejected or accepted with only a low credit line. The use of credit-scoring varies from those that are using only credit bureau data and JCIC data, to those that blend bureau data with other information based on the firm's own experience, to the most advanced applications using adaptive algorithms (Burns & Stanley, 2001).

Faced with an exploding number of transactions, experienced personnel are no longer able to identify fraud caused by stolen cards, false merchants or disloyal employees. Automatic fraud detection has therefore become an important topic (Kuo et. al., 2004).

3.2 Data Mining

Data mining aims to discover hidden knowledge, unknown patterns, and new rules from large databases that are potentially useful and ultimately understandable for making crucial decisions (Zhang & Zhou, 2004).

The knowledge to be mined is closely related to a target application and the original data. Therefore, data mining should be considered along with several other issues rather than an isolated task. First, data mining needs to take ultimate applications into account. For example, credit card fraud detection and stock market prediction may require different data mining techniques.

3.3 Credit Card Data Mining

In recent years data mining approaches (Zhang & Zhou, 2004) have been proposed for solving credit card related problems. For instance, Yu (2003) addressed the detection of bad debt and inactive accounts from the customer profile and delinquency. Chiang (2003) addressed the relationship between credit-evaluation factors and credit card default risk. Lee (2002) analysed the credit risk of cardholders based on their profile. Lin (2003) addressed the risk improvement of cardholders based on their payment record. Aleskerov et. al. (1997) and Brause et. al. (1999) both addressed credit card fraud detection using transaction category, amount and time.

3.4 Fuzzy Logic

Fuzzy logic systems (FLS) usually contain four major components (Huang et. al., 2006): *Fuzzification*, i.e., the relationships between input variables and linguistic variables is defined by using membership functions (MFs). *Knowledge Base*, i.e., application domain expert knowledge. It uses the linguistic rules to represent an expert's tactic. *Decision Making Logic*, i.e., the inference engine has the capacity of inferring rules using fuzzy rules and fuzzy implication, and so simulates the human decision-making ability. *Defuzzification*, i.e.,

the reverse process of fuzzification. It produces a crisp output value from the fuzzy outputs of the inference engine.

3.5 Back-Propagation Neural Network

Neural networks (NNs) have been widely applied to various tasks in hybrid combinations with data mining approaches (Huang et. al., 2008; Huang et. al., 2007; Kijirikul & Chongkasemwongse, 2001). NNs are computer models built to mimic human pattern recognition. NNs are able to learn the unknown non-linear relationships between causes and prospective outcomes based on examples.

3.6 BPN Model

The back-propagation neural network (BPN) is one of the most popular artificial supervised learning neural networks. It is very powerful in terms of assessment and prediction. A neural network consists of a set of fundamental processing elements (also called neurons) that are distributed in hierarchical layers. Most multi-layer feed-forward neural networks contain three types of layers, namely the *input layer*, the *hidden layer* and the *output layer*. The input layer receives features of input data and distributes them to the hidden layer. After each neuron in a hidden layer receives the inputs from all of the neurons in a layer ahead of input layer, the values are added through applied weights and converted to an output value by an activation function. Then, the output is passed to all of the neurons in the next layer, providing a feed-forward path to the output layer. The weights between two neurons in two adjacent layers are adjusted through an iterative training process while training samples are presented to the network. The output layer yields the outputs of the network.

3.7 BPN Algorithm

The BPN algorithm applies the basic principle of the gradient steepest descent method to minimize the error function. It compares the outputs of the processing units in the output layer with desired outputs to adjust the connecting weights. The weights between two neurons in two adjacent layers are adjusted through an iterative training process while training samples are presented to the network. A close approximation of the transformation function which is compared the outputs of the processing units in the output layer with desired outputs can be acquired (Huang & Hsieh, 2003). After training, the network can be used to classify unseen data. The unseen data are then fed into the network, and the output with highest value will be taken as the prediction.

4. Method

4.1 Training and test material

Experimental data were collected randomly from a local bank in Taiwan on June 30, 2004. In order to reduce noise and handle missing values, the stop card accounts and unreasonable accounts were discarded from the dataset. The original dataset consisted of 744,477 accounts which were reduced to 449,211 records after a cleanup. The validation samples comprised 446,211 accounts including 432,916 (97.02%) normal accounts, 9,105 (2.04%) bad accounts and 4,190 (0.94%) charge-off accounts. The data set was divided to a training sample and a test sample. The training samples consisted of 3,000 accounts upon which 1,000 (33.33%)

were bad accounts, 1,000 (33.33%) were charge-off accounts, and 1,000 (33.33%) were normal accounts.

Input variables that deal with cardholder's accounts were divided into two groups: (1) socio-economic data and (2) financial data. Basic socio-economic characteristics that are used in our raw database are: (1) gender, (2) marital status, (3) education, (4) age and (5) occupation. Basic financial characteristics that are used in our raw database are: (1) credit limit, (2) current balance, (3) payment amount, (4) transaction amount, (5) revolving credit amount, (6) late charge fee, (7) credit cash amount, (8) delinquency flag, (9) cycle, (10) client's account age and (11) zip code.

Clients' accounts are classified as being either *normal*, *bad debt*, or *charge-off*. Clients are in bad debt if they exceed a contracted overdraft for more than 30 days during a period of 6 months. Clients are charge-off if they exceed a contracted overdraft for more than 180 days during a period of 6 months. Otherwise, a client is considered normal.

4.2 Data Selection

The scheme employed herein incorporates the following features: (1) *Credit limit*, the maximum amount a person is allowed to borrow on a credit card (see Table 1). It includes purchases, cash advances, and any finance charges or fees. Some issuers increase cardholder's credit limit to promote their consumption. Most of the bad account's credit limit is below NT\$100,000. The normal account's credit limit is between NT\$100,000 and NT\$300,000. (2) *Gender*, Table 2 shows the credit status related to gender. Female clients have a higher rate of normal accounts than male clients. Males therefore have higher risk of bad accounts than females. (3) *Education*, Table 3 shows the cardholder's education. Clients with good education have higher rate of normal accounts than other groups. Accounts with just a high-school diploma are at higher risk than others. (4) *Marital status*, the data revealed that the credit status has no apparent relationship to marital status. (5) *Cycle*, a monthly billing date from a creditor which summarizes the activity and expenses on an account between the last billing date and the current billing date. The effect of cycle on credit status shows no apparent difference in the entire classes as given in Table 3. (6) *Age*, there is a group of high-risk cardholders between 20-40 years of age as shown in Table 4. Note that workers younger than 20 years old or elder than 65 years old who are unemployed are discarded (see Table 4). (7) *Client account age*, clients who have had accounts for about one year make up a group of high-risk cardholders. Normal account holders continue using their credit cards without problems beyond the first year as shown in Table 5. (8) *Current balance*, the total amount of money owed on a credit line. It includes any unpaid balance from the previous months, new purchases, cash advances and any charges at present. There are 91.78% normal accounts owed below NT\$100,000 dollars as given in Table 6. (9) *Payment amount paid before the next billing date*, the bad accounts and charge-off accounts have low payment amounts. They have no ability to pay off their credit amount as shown in Table 6. (10) *Transaction amount*, the amount that a person charges and owes on a credit card between the last billing date and the current billing date. It includes purchases, cash advances, and any finance charges or fees. The account of poor credit status will be limited their purchase as shown in Table 7. (11) *Delinquent flag*, a credit line or loan account where the late payments have been received or the payments have not been made according to the

respective terms and conditions in a current month. The charge-off account has the current delinquent flag of long term as demonstrated in Table 8. (12) *Balance to credit line ratio (B/C)*, is used to record the cardholder usage of the credit line. The normal accounts use the credit card in a good manner. The charge-off accounts have a high B/C ratio with over purchase as shown in Table 9.

Credit limit	Normal account		Bad debt account		Charge-off account		Total account	
	No.	%	No.	%	No.	%	No.	%
1 - 100000	66,090	15.2%	6,217	61.5%	3,176	61.2%	75,483	16.8%
100,001-200,000	117,227	27.0%	2,179	21.6%	1,186	22.9%	120,592	26.9%
200,001- 300,000	135,965	31.3%	1,230	12.2%	682	13.1%	137,877	30.7%
300,001- 400,000	70,572	16.3%	328	3.3%	119	2.3%	71,019	15.8%
400,001- 500,000	26,495	6.1%	124	1.2%	20	0.4%	26,639	5.9%
500,001and over	17,567	4.1%	27	0.3%	7	0.1%	17,601	3.9%
Total	433,916	100.0%	10,105	100.0%	5,190	100.0%	449,211	100.0%

Table 1. Risk related to credit limit.

Gender	Normal account		Bad debt account		Charge-off account		Total account	
	No.	%	No.	%	No.	%	No.	%
Female	291,118	67.1%	4,841	47.9%	2259	43.5%	298,218	66.4%
Male	142,798	32.9%	5,264	52.1%	2931	56.5%	150,993	33.6%
Total	433,916	100.0%	10,105	100.0%	5190	100.0%	449,211	100.0%

Table 2. Risk related to gender.

Education	Normal account		Bad debt account		Charge-off account		Total account	
	No.	%	No.	%	No.	%	No.	%
Master	19,725	4.6%	135	1.3%	27	0.5%	19,887	4.4%
College	193,883	44.7%	2,250	22.3%	843	16.2%	196,976	43.9%
High school	143,807	33.1%	5,302	52.5%	2,953	56.9%	152,062	33.9%
Unknown	76,501	17.6%	2,418	23.9%	1,367	26.3%	80,286	17.9%
Total	433,916	100.0%	10,105	100.0%	5,190	100.0%	449,211	100.0%

Table 3. Risk related to education.

Age	Normal account		Bad debt account		Charge-off account		Total account	
	No.	%	No.	%	No.	%	No.	%
20-30	86,977	20.0%	3,071	30.4%	1,267	24.4%	91,315	20.3%
31-40	156,391	36.0%	2,976	29.5%	1,617	31.2%	160,984	35.8%
41-50	119,563	27.6%	2,550	25.2%	1,506	29.0%	123,619	27.5%
51-60	55,934	12.9%	1,275	12.6%	678	13.1%	57,887	12.9%

Age	Normal account		Bad debt account		Charge-off account		Total account	
	No.	%	No.	%	No.	%	No.	%
61-70	12,302	2.8%	219	2.2%	112	2.2%	12,633	2.8%
71-80	2,713	0.6%	14	0.1%	10	0.2%	2,737	0.6%
81-90	33	0.0%	0	0.0%	0	0.0%	33	0.0%
90 and over	3	0.0%	0	0.0%	0	0.0%	3	0.0%
Total	433,916	100.0%	10,105	100.0%	5,190	100.0%	449,211	100.0%

Table 4. Comparison of credit status by age.

Account age	Normal account		Bad debt account		Charge-off account		Total account	
	No.	%	No.	%	No.	%	No.	%
1	22,971	5.3%	405	4.0%	0	0.0%	23,376	5.2%
2	80,839	18.6%	3,854	38.1%	1,689	32.5%	86,382	19.2%
3	90,434	20.8%	2,276	22.5%	1,405	27.1%	94,115	21.0%
4	186,869	43.1%	2,946	29.2%	1,697	32.7%	191,512	42.6%
5	8,368	1.9%	132	1.3%	119	2.3%	8,619	1.9%
6	15,056	3.5%	164	1.6%	101	2.0%	15,321	3.4%
7	11,729	2.7%	136	1.4%	66	1.3%	11,931	2.7%
8	7,501	1.7%	79	0.8%	47	0.9%	7,627	1.7%
9	3,317	0.8%	45	0.5%	22	0.4%	3,384	0.8%
10	1,916	0.4%	19	0.2%	19	0.4%	1,954	0.4%
11	1,824	0.4%	21	0.2%	17	0.3%	1,862	0.4%
12	2,960	0.7%	28	0.3%	7	0.1%	2,995	0.7%
13	132	0.0%	0	0.0%	1	0.0%	133	0.0%
Total	433,916	100.0%	10,105	100.0%	5,190	100.0%	449,211	100.0%

Table 5. Risk related to account age.

Current Balance	Normal account		Bad debt account		Charge-off account		Total account	
	No.	%	No.	%	No.	%	No.	%
0	178,493	41.1%	2,406	23.8%	11	0.2%	180,910	40.3%
1 – 100,000	219,727	50.6%	5,568	55.1%	3,374	65.0%	228,669	50.9%
100,001-200,000	22,769	5.3%	1,151	11.4%	1,046	20.2%	24,966	5.6%
200,001-300,000	8,684	2.0%	643	6.4%	560	10.8%	9,887	2.2%
300,001-400,000	3,005	0.7%	212	2.1%	136	2.6%	3,353	0.8%
400,001-500,000	981	0.2%	70	0.7%	43	0.8%	1,094	0.2%
500,001-600,000	193	0.0%	41	0.4%	9	0.8%	243	0.1%
600,001-700,000	26	0.0%	5	0.1%	9	0.8%	40	0.0%
Total	433,916	100.0%	10,105	100.0%	5,190	100.0%	449,211	100.0%

Table 6. Risk related to current balance.

Payment amount	Normal account		Bad debt account		Charge-off account		Total account	
	No.	Percentage	No.	Percentage	No.	Percentage	No.	Percentage
0	334,814	77.2%	9,546	94.5%	5,104	98.3%	349,464	77.8%
1 – 100,000	64,254	14.8%	466	4.6%	75	1.5%	64,795	14.4%
100,001- 200,000	15,023	3.5%	37	0.4%	4	0.1%	15,064	3.4%
200,001- 300,000	6,177	1.4%	20	0.2%	4	0.1%	6,201	1.4%
300,001- 400,000	3,270	0.8%	6	0.1%	0	0.0%	3,276	0.7%
400,001- 500,000	2,358	0.5%	2	0.0%	0	0.0%	2,360	0.5%
500,001- 600,000	1,444	0.3%	6	0.1%	0	0.0%	1,450	0.3%
600,001- 700,000	1,042	0.2%	6	0.1%	3	0.1%	1,051	0.2%
700,001 and over	791	0.2%	0	0.0%	0	0.0%	791	0.2%
Total	433,916	100.0%	10,103	100.0%	5,190	100.0%	449,209	1.1%

Table 7. Risk related to payment amount.

Transaction amount	Normal account		Bad debt account		Charge-off account		Total account	
	No.	Percentage	No.	Percentage	No.	Percentage	No.	Percentage
0	426,195	98.2%	10,066	99.6%	5,190	100.0%	441,451	98.3%
1 – 100,000	6,367	1.5%	34	0.3%	0	0.0%	6,401	1.4%
100,001 – 200,000	923	0.2%	3	0.0%	0	0.0%	926	0.2%
200,001 – 300,000	431	0.1%	2	0.0%	0	0.0%	433	0.1%
Total	433,916	100.0%	10,105	100.0%	5,190	100.0%	449,211	100.0%

Table 8. Risk related to delinquent flag.

Delinquent flag	Normal account		Bad debt account		Charge-off account		Total account	
	No.	Percentage	No.	Percentage	No.	Percentage	No.	Percentage
0	25,625	5.9%	215	2.1%	13	0.3%	25,853	5.8%
1	54,796	12.6%	340	3.4%	4	0.1%	55,140	12.3%
2	2,826	0.7%	689	6.8%	3	0.1%	3,518	0.8%
3	177	0.0%	486	4.8%	4	0.1%	667	0.2%
4	12	0.0%	425	4.2%	2	0.0%	439	0.1%
5	8	0.0%	352	3.5%	3	0.1%	363	0.1%
6	7	0.0%	443	4.4%	27	0.5%	477	0.1%
7	15	0.0%	230	2.3%	18	0.4%	263	0.1%
8	1	0.0%	283	2.8%	36	0.7%	320	0.1%
9	0	0.0%	1	0.0%	3,105	59.8%	3,106	0.7%
B	84,804	19.5%	51	0.5%	8	0.2%	84,863	18.9%
Z	265,645	61.2%	6,590	65.2%	1,967	37.9%	274,202	61.0%
Total	433,916	100.0%	10,105	100.0%	5,190	100.0%	449,211	100.0%

Table 9. Risk related to B/C.

4.3 Fuzzy Input Features

A fuzzy rule-base system was used to obtain good input features. The fuzzy values were obtained in five steps. First, the membership functions were determined as follows.

$$\mu_{A1} = \begin{cases} 1, x_1 \leq 20000 \\ \frac{70000 - x_1}{50000}, 20000 < x_1 \leq 70000 \\ 0, x_1 > 70000 \end{cases} \quad (1)$$

$$\mu_{A2} = \begin{cases} 0, x_1 \leq 60000 \\ \frac{x_1 - 60000}{30000}, 60000 < x_1 \leq 90000 \\ \frac{120000 - x_1}{30000}, 90000 < x_1 \leq 120000 \\ 0, x_1 > 120000 \end{cases} \quad (2)$$

$$\mu_{A3} = \begin{cases} 0, x_1 \leq 100000 \\ \frac{x_1 - 100000}{50000}, 100000 < x_1 \leq 150000 \\ \frac{200000 - x_1}{50000}, 150000 < x_1 \leq 200000 \\ 0, x_1 > 200000 \end{cases} \quad (3)$$

$$\mu_{A4} = \begin{cases} 0, x_1 \leq 190000 \\ \frac{x_1 - 190000}{110000}, 190000 < x_1 \leq 300000 \\ 1, x_1 > 300000 \end{cases} \quad (4)$$

$$\mu_{B1} = \begin{cases} 1, x_2 \leq 20 \\ \frac{25 - x_2}{5}, 20 < x_2 \leq 25 \\ 0, x_2 > 25 \end{cases} \quad (5)$$

$$\mu_{B2} = \begin{cases} 0, x_2 \leq 24 \\ \frac{x_2 - 24}{15}, 24 < x_2 \leq 39 \\ \frac{54 - x_2}{15}, 39 < x_2 \leq 54 \\ 0, x_2 > 54 \end{cases} \quad (6)$$

$$\mu_{B3} = \begin{cases} 0, x_2 \leq 53 \\ \frac{x_2 - 53}{7}, 53 < x_2 \leq 60 \\ 1, x_2 > 60 \end{cases} \quad (7)$$

$$\mu_{C1} = \begin{cases} 1, x_3 \leq 10000 \\ \frac{40000 - x_3}{30000}, 10000 < x_3 \leq 40000 \\ 0, x_3 > 40000 \end{cases} \quad (8)$$

$$\mu_{C2} = \begin{cases} 0, x_3 \leq 30000 \\ \frac{x_3 - 30000}{30000}, 30000 < x_3 \leq 60000 \\ \frac{90000 - x_3}{30000}, 60000 < x_3 \leq 90000 \\ 0, x_3 > 90000 \end{cases} \quad (9)$$

$$\mu_{C3} = \begin{cases} 0, x_3 \leq 80000 \\ \frac{x_3 - 80000}{30000}, 80000 < x_3 \leq 110000 \\ \frac{140000 - x_3}{30000}, 110000 < x_3 \leq 140000 \\ 0, x_3 > 140000 \end{cases} \quad (10)$$

$$\mu_{C4} = \begin{cases} 0, x_3 \leq 130000 \\ \frac{x_3 - 130000}{120000}, 130000 < x_3 \leq 250000 \\ 1, x_3 > 250000 \end{cases} \quad (11)$$

$$\mu_{D1} = \begin{cases} 1, x_4 \leq 10000 \\ \frac{40000 - x_4}{30000}, 10000 < x_3 \leq 40000 \\ 0, x_4 > 40000 \end{cases} \quad (12)$$

$$\mu_{D2} = \begin{cases} 0, x_4 \leq 30000 \\ \frac{x_4 - 30000}{30000}, 30000 < x_4 \leq 60000 \\ \frac{90000 - x_4}{30000}, 60000 < x_4 \leq 90000 \\ 0, x_4 > 90000 \end{cases} \quad (13)$$

$$\mu_{D3} = \begin{cases} 0, x_4 \leq 80000 \\ \frac{x_4 - 80000}{30000}, 80000 < x_4 \leq 110000 \\ \frac{140000 - x_4}{30000}, 110000 < x_4 \leq 140000 \\ 0, x_4 > 140000 \end{cases} \quad (14)$$

$$\mu_{D4} = \begin{cases} 0, x_4 \leq 130000 \\ \frac{x_4 - 130000}{120000}, 130000 < x_4 \leq 250000 \\ 1, x_4 > 250000 \end{cases} \quad (15)$$

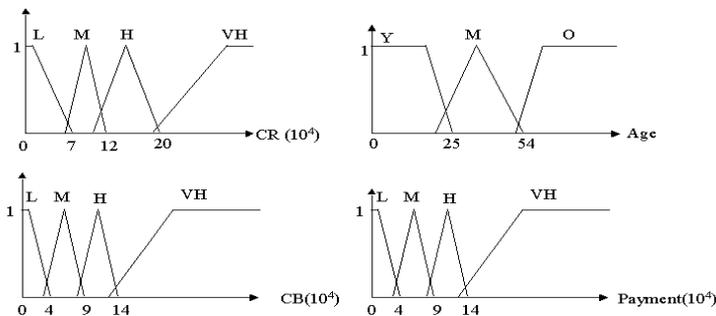


Fig. 1. The respective membership functions for CR (credit line), age, CB (current balance) and payment.

Hence, μ_{An} , μ_{Bn} , μ_{Cn} and μ_{Dn} denote fuzzy membership functions for a credit line, age, current balance and payment, respectively. n is the center of a triangular fuzzy set. The triangular fuzzy sets are plotted in Fig. 1. L, M, H, and VH denote the linguistic variables low, medium, high, very high in the amount feature. Y, M and O denote the linguistic variables young, middle and old for the age feature. Next, the fuzzy rules are created. The rule sets are shown in Tables 10 and 11.

Then, weights are assigned to each linguistic term using subsethood values. Next, the fuzzy membership values are calculated for each linguistic term in each subgroup as given in Tables 10 and 11. The fuzzy membership values are calculated according to each

classification result. Finally, the classification is calculated using the de-fuzzification to get a single value that represents the output fuzzy set, namely the risk ratio.

Payment	Current balance			
	Very high	High	Medium	Low
Very high	Common	Good	Excellent	Excellent
High	Good	Common	Good	Excellent
Medium	Worst	Worst	Common	Good
Low	Worst	Worst	Worst	Common

Table 10. Current balance to payment linguistic labels matrix.

Age	Credit line			
	Very high	High	Medium	Low
Young	Good	Common	Common	Worst
Middle	Excellent	Good	Common	Common
Old	Common	Common	Worst	Worst

Table 11. Credit line to age linguistic labels matrix.

4.4 Input output coding

Three types of input variables are used, namely qualitative, quantitative (or numeric) and ratio (Durham University, 2008). A binary encoding scheme is used to represent the presence 1, or absence 0, of a particular (qualitative) data. Quantitative data are normalized into the range [0, 1]. Ratios are the proportion of related variables calculated to signal the importance of data. We encode ratios by computing the proportion of related variables to describe the importance of the data.

Input variables comprise (1) *gender*, encoded using one bit (0 = female, 1 = male), (2) *customer age*, denotes the customer age between 20 and 80 years, (3) *age of the client account*, from 1 to 13 years, (4) *current balance*, denotes the total amount of money owed by cardholders in the range from 1 to 1,000,000, (5) *payment amount*, denotes the total amount of money debited by cardholders and is in the range from 1 to 1,000,000, (6) *transaction amount*, denotes the total amount of money consumed by cardholders from 1 to 1,000,000, (7) *delinquent flag*, records the status that late payments have been received and is encoded into 3 binary bits, where 000 indicates full pay, 001 minimum monthly payment, 010 delinquent within one month, 011 delinquent within two to four months, 100 delinquent within five to seven months and 101 delinquent above seven months, (8) *risk ratio*, is given by the FMS and encoded into one ratio bit, (9) *payment amount to current balance ratio*, denotes solvency and is encoded into one ratio bit.

The two output variables signal the *cardholder status*. These are coded as 00 normal, 01 bad debt or 10 charge-off accounts.

5. Experimental Results

The tools used for implementing the experimental system include JBUILDER 9.0, SQL 2000 and Windows 2000. Input values were normalized to the range from 0 to 1. After training,

the neural network is capable of classifying credit status. A predefined threshold of 0.8 was used to detect suspicious cases.

5.1 Procedure

A small dataset, provided by a local bank in Taiwan, was used to demonstrate how this method works. This data set contains 449,256 accounts belonging to three classes; namely 433,961 normal accounts, 10,105 bad debt accounts, and 5,190 charge-off accounts. There are only 0.35% abnormal accounts in practice. The experimental data set is divided into two subsets, namely 3,000 training examples and 10,000 test examples. The training set comprises 1,000 normal accounts, 1,000 bad accounts and 1,000 charge-off accounts. A two-way cross validation table was used to select input features. To obtain good input features a fuzzy rule-based system was incorporated. A risk ratio of variables with fuzzy value was created to enhance the prediction accuracy. After data transformation, the features to be input to the BPN were encoded in the [0, 1] interval. The BPN classifies input into one of three classes. The network is repetitively trained with different network parameters until it converges. We randomly selected 3,000 training examples from the total sample, where 1,000 examples were normal accounts, 1,000 were bad dept account and 1,000 were charge-off accounts.

The neural network learning parameters need to be set to avoid the effect of over-fitting and to maintain reasonable performance. Fig. 2 and 3 show system screenshots of the two main views. The learning parameters were tuned by running the simulations multiple times.

The back-propagation network comprised 11 input nodes, 7 hidden nodes, and 2 output nodes. The coding of the output vectors were as follows: bad debt accounts (1,0), charge-off accounts (0,1) and normal accounts (0,0). Table 12 shows BPN typical input output mapping examples.

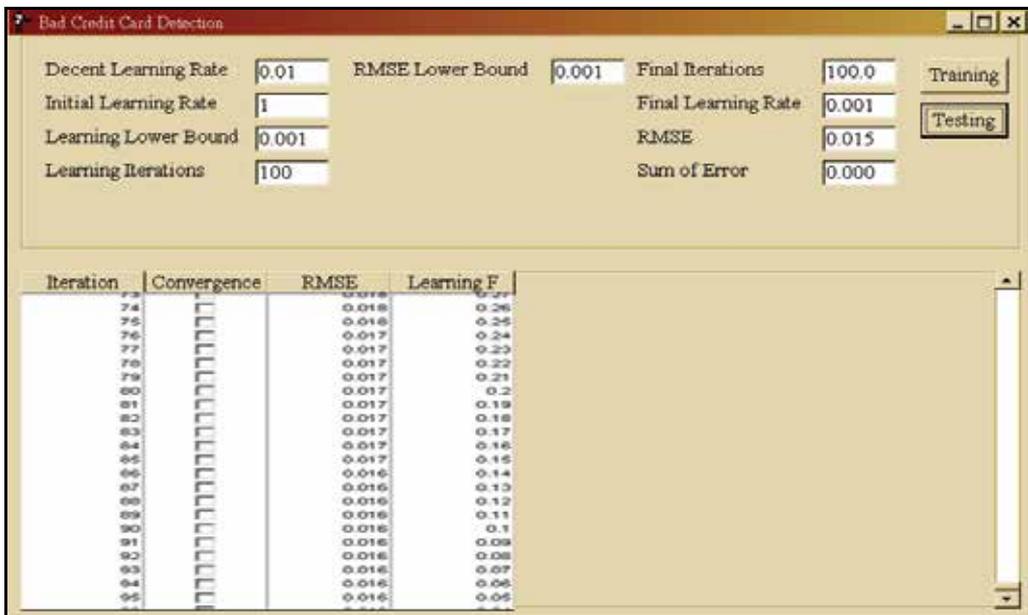


Fig. 2. The training screen.

Bad Credit Card Detection

Socio Economic Characteristics

Gender	E	ID Established Date	2008-08-12 00:30:00	Credit Limit	2000000
Card Issued Date	2008-02-27	Valid Date	3000	Client Age	34
Transaction	E	Occupation	34	Marital Status	1
Zip Code	11157	Billing Cycle	17		
Control Code	A	Control Date	2004-09-03		

Financial Characteristics

Current Balance	283217.0	Receiving Credit	3210		
Cash Advance by Person	0.0	Cash Advance by Card	0.0	Cash Advance Fee by Card	0.0
Transaction Amount by Person	500.0	Transaction Amount by Card	0.0	Current Balance	432110
Payment Amount by Person	0.0	Payment Amount by Card	0.0		

Training Results

Initial Learning Rate	0.010000	Training Learning Rate	0.010000	Root Mean Square Error	0.015253
Bad Credit Train	0.989778	Bad Credit Classified	1.000000	Bad Credit Membership Degree	1.0
Bad Debt Train	0.001191	Bad Debt Classified	0.000000	Bad Debt Membership Degree	0.0
Recall Rate	0.947000	Total Recall Rate	0.917919	Correct Classified Rate	Correct

First Previous Next [] All

Fig. 3. The test screen.

Name	Type	I/O	bad debt	charge-off	normal
X1	Binary	Input	1	1	0
X2	Quantification	Input	0.25	0.31	0.51
X3	Quantification	Input	0.18	0.23	0.23
X4	Quantification	Input	0.45	0.95	0.17
X5	Quantification	Input	0.18	0.15	0.36
X6	Quantification	Input	0.15	0.15	0.16
X7	Binary	Input	010	101	000
X8	Ratio	Input	0.44	0.95	0.16
X9	Ratio	Input	0.29	0.15	0.15
O1	Binary	Output	0.9997	0.0091	0.0215
O2	Binary	Output	0.0002	0.9905	0.0286

Table 12. Neural network mapping examples.

5.2 Performance Evaluation

Performance was measured in terms of precision and recall. Precision is defined as the proportion of classified instances that were correctly classified, and recall as the proportion of instances classified correctly (Cohn, 2003), or formally

$$recall = \frac{|A(q) \cap R(q)|}{|R(q)|}, \quad (16)$$

$$precision = \frac{|A(q) \cap R(q)|}{|A(q)|}. \quad (17)$$

5.3 Detection Performance

We started with a network with all possible input nodes. But all possible nodes are never needed to represent a system. Therefore, we used two-way cross validations to filter off redundant input features. Besides, we designed risk ratio of variables to raise the performance. We fixed the network parameters and set the initial learning rate to 10 with a decreasing rate of 0.001. We consolidated the performance achieved with the fuzzy BPN and the non-fuzzy BPN (see Table 13). Clearly, the fuzzy BPN yields better results than the conventional BPN.

Iterations	Proposed detection model		Conventional BPN model	
	Recall (R1)	Precision (P1)	Recall (R2)	Precision (P2)
100	85.20%	79.00%	75.80%	65.00%
200	87.95%	84.00%	80.15%	72.00%
300	88.30%	83.90%	81.30%	73.00%
500	90.00%	85.00%	83.60%	76.80%
1000	95.50%	95.00%	86.00%	80.00%
2000	95.35%	94.97%	86.15%	80.20%
Average	90.38%	86.98%	82.17%	74.50%

Table 13. The performance of the proposed detection model versus conventional BPN detection.

Additionally, a dataset comprising 10,000 simulated entries were used to evaluate and validate the system, where different normal to abnormal data ratios were considered to diagnose different behaviours. The results are listed in Table 14, and each experiment shows the size of the detected set, the number of addressed problems, the total precision P and the

total recall R rates. This experimental evidence demonstrates that the strategy is capable of effectively tackling more than 90% of the problems.

Result	Size	Number of addressed problems			Recall	Precision
		Normal	Bad debt	Charge-off		
Test Data-1	282	9706	137	145	94.00%	95.59%
Test Data-2	4512	5177	2036	2476	90.24%	93.55%
Test Data-3	5372	4068	2420	2952	89.53%	90.56%

Table 14. The comparisons of test results from different iterations.

5.4 Implications

The strategy outlined herein could be used for risk management, analysis of business rules, delinquent diagnosis and abnormal accounts forecasting. Risk management helps reduce issuers' losses due to bad debt. When detecting bad debt accounts, the issuers can reduce the credit line. Moreover cardholders can be offered realistic loan plans to help them overcome their financial difficulties (Lin, 2003). Analysis of business rules is used to establish a way to normalize the analysis that facilitates to compare the business rules across various types of credit card accounts. We could take these rules to develop a business model in the knowledge management system. Delinquent diagnosis is used to analyze the existing delinquent factors as a high-level understanding of process and control systems of an application. Delinquent diagnosis is a way to monitor the delinquent accounts early to avoid losses due to bad debt. Abnormal account forecasting is commonly used to recognize portfolio dynamics and behaviour patterns.

6. Conclusions and Future Work

A novel scheme for the bad credit account detection was proposed. A fuzzy rule-based system was used to provide inputs for a back-propagation neural network that was used for classifying accounts. The proposed system has been tested on real credit data and it is capable of detecting bad accounts in the large data set with a success rate of more than 90%. Future work includes integrating the proposed system with credit card risk management systems and the introduction of noise reduction mechanism for discarding outlier accounts, i.e., observations that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism. Finally, it would be desirable to integrate other AI algorithms (e.g., GA) with data mining to enhance predictive accuracy and apply the algorithm to relational (e.g., spatial) data.

7. References

- Aleskerov, E.; Freisleben, B. & Rao, B. (1997). CARDWATCH: a neural network based database mining system for credit card fraud detection. *Proceedings of IEEE Int. Conf. on Computational Intelligence for Financial Engineering*, pp. 220-226, NY, USA, March 1997.

- Brause, R.; Langsdorf, T. & Hepp, M. (1999). Neural data mining for credit card fraud detection. *Proceedings of 11th IEEE Int. Conf. on Tools with Artificial Intelligence*, pp. 103-106, Chicago, IL, USA, November 1999.
- Burns, P. & Stanley, A. (2001). Managing consumer credit risk. *Federal Reserve Bank of Philadelphia Payment Cards Center Discussion Paper*, no. 01-03, pp. 1-7, November 2001.
- Chakraborty, D. & Pal, N.R. (2001). Integrated feature analysis and fuzzy rule-base system identification in a neuro-fuzzy paradigm. *IEEE Trans. Systems, Man, and Cybernetics*, vol. 31, no. 4, pp. 391-400, June 2001.
- Chiang, S.C. (2003). *The relationship between credit-evaluation factors and credit card default risk—an example of the credit cards issued by the a financial institute in Taiwan*. Master Thesis, Department of Insurance, Feng Chia University, Taichung, Taiwan, June 2003.
- Cohn, T. (2003). Performance metrics for word sense disambiguation. *Proceedings of the Australasian Language Technology Workshop*, pp. 49-56, Victoria, Australia, December 2003.
- Huang, Y.-P.; Chang, T.-W.; Chen, Y.-R. & Sandnes, F.E. (2008). A back propagation based real-time license plate recognition system. *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 22, no. 2, pp. 233-251, March 2008.
- Huang, Y.-P. & Hsieh, W.-J. (2003). The application of grey model and back-propagation network to establish the alarm mechanism for the premium rate service. *Journal of Grey System*, vol. 6, no. 2, pp. 75-88, December 2003.
- Huang, Y.-P.; Hsu, L.-W. & Sandnes, F.E. (2007). An intelligent subtitle detection model for locating television commercials. *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 2, pp. 485-492, April 2007.
- Huang, Y.-P.; Huang, Y.-H. & Sandnes, F.E. (2006). A fuzzy inference model-based non-reassuring fetal status monitoring system. *Int. Journal of Fuzzy Systems*, vol. 8, no. 1, pp. 57-64, March 2006.
- Kijsirikul, B. & Chongkasemwongse, K. (2001). Decision tree pruning using backpropagation neural networks. *Proceedings of IEEE Int. Conf. on Neural Networks*, vol. 3, pp. 1876-1880, Washington D.C., USA, Month 2001.
- Kuo, Y.F.; Lu, C.T.; Sirwongwattana, S. & Huang, Y.-P. (2004). Survey of fraud detection techniques. *Proceedings of IEEE Int. Conf. on Networking, Sensing and Control*, pp. 749-754, Taipei, Taiwan, March 2004.
- Lee, H.M.; Chen, C.M.; Chen, J.M. & Jou, Y.L. (2001). An efficient fuzzy classifier with feature selection based on fuzzy entropy. *IEEE Trans. on Systems, Man and Cybernetics*, vol. 31, no. 3, pp.426-432, June 2001.
- Lee, M.H. (2002). *A study on the credit risk of the credit card holder*. Master Thesis, Department of Insurance, Feng Chia University, Taichung, Taiwan, June 2001.
- Lin, C.J. (2003). *Data mining for risk improving of credit card*. Master Thesis, Department of Computer Science and Engineering, Tamkang University, Taipei, Taiwan, June 2003.
- Lin, J. (2005). Interest-rate cap dropped as bankers offer relief plan. *Taipei Times*, Friday, Dec. 16, 2005.
- Yu, H.H. (2003). *The research of applying improved artificial neural network to credit card customer relationship management*. Master Thesis, Department of Business Management, National Taipei University of Technology, Taipei, Taiwan, June 2003.

Zhang, D. & Zhou, L. (2004). Discovering golden nuggets: data mining in financial application. *IEEE Trans. on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, vol. 34, no. 4, pp. 513-522, November 2004.

Cardservice International website (2008), <http://www.aboutcsi.com>.

Central Bank of China website (2008), <http://www.cbc.gov.tw>.

Durham University website (2008), <http://www.dur.ac.uk>.

Indiana University Office of the Treasurer website (2008), <http://www.indiana.edu>.

National Credit Card Center of R.O.C website (2008), <http://www.nccc.com.tw>.

ShopSite website (2008), <http://www.shopsite.com>.

Taiwan Financial Supervisory Commission website (2008), <http://www.fscey.gov.tw>.

Taiwan Joint Credit Information Center website (2008), <http://www.jcic.org.tw>.

The International Commercial Bank of China website (2008), <http://www.icbc.com.tw>.

Improved Chaotic Associative Memory for Successive Learning

Takahiro IKEYA and Yuko OSANA
Tokyo University of Technology
Japan

1. Introduction

Recently, neural networks are drawing much attention as a method to realize flexible information processing. Neural networks consider neuron groups of the brain in the creature, and imitate these neurons technologically. Neural networks have some features, especially one of the important features is that the networks can learn to acquire the ability of information processing.

In the field of neural network, many models have been proposed such as the Back Propagation algorithm (Rumelhart et al., 1986), the Self-Organizing Map (Kohonen, 1994), the Hopfield network (Hopfield, 1982) and the Bidirectional Associative Memory (Kosko, 1988). In these models, the learning process and the recall process are divided, and therefore they need all information to learn in advance.

However, in the real world, it is very difficult to get all information to learn in advance. So we need the model whose learning and recall processes are not divided. As such model, Grossberg and Carpenter proposed the Adaptive Resonance Theory (ART) (Carpenter & Grossberg, 1995). However, the ART is based on the local representation, and therefore it is not robust for damage. While in the field of associative memories, some models have been proposed (Watanabe et al., 1995; Osana & Hagiwara, 1999; Kawasaki et al., 2000; Ideguchi et al., 2005). Since these models are based on the distributed representation, they have the robustness for damaged neurons. However, their storage capacity is very small because their learning processes are based on the Hebbian learning. In contrast, the Hetero Chaotic Associative Memory for Successive Learning with give up function (HCAMSL) (Arai & Osana, 2006) and the Hetero Chaotic Associative Memory for Successive Learning with Multi-Winners competition (HCAMSL-MW) (Ando et al., 2006) have been proposed in order to improve the storage capacity.

In this research, we propose an Improved Chaotic Associative Memory for Successive Learning (ICAMSL). The proposed model is based on the Hetero Chaotic Associative Memory for Successive Learning with give up function (HCAMSL) (Arai & Osana, 2006) and the Hetero Chaotic Associative Memory for Successive Learning with Multi-Winners competition (HCAMSL-MW) (Ando et al., 2006). In the proposed ICAMSL, the learning process and recall process are not divided. When an unstored pattern is given to the

network, the ICAMSL can learn the pattern successively. Moreover, the storage capacity of the proposed ICAMSL is larger than that of the conventional HCAMSL/HCAMSL-MW.

2. Chaotic Neural Network

A neural network composed of the chaotic neurons is called a chaotic neural network (Aihara et al., 1990).

The dynamics of the chaotic neuron i in the neural network composed of N chaotic neurons is represented by the following equation:

$$x_i(t+1) = f \left[\sum_{j=1}^M v_{ij} \sum_{d=0}^t k_s^d A_j(t-d) + \sum_{j=1}^N w_{ij} \sum_{d=0}^t k_m^d x_j(t-d) - \alpha \sum_{d=0}^t k_r^d x_i(t-d) - \theta_i \right] \quad (1)$$

where $x_i(t)$ shows the output of the neuron i at the time t , M is the number of the external inputs, v_{ij} is the connection weight from the external input j to the neuron i , $A_j(t)$ is the strength of the external input j at the time t , w_{ij} is the connection weight between the neuron i and the neuron j , α is the scaling factor of the refractoriness, k_s , k_m and k_r are the damping factors and θ_i is the threshold of the neuron i . $f(\cdot)$ is the following output function:

$$f(u) = \frac{1}{1 + \exp(-u/\varepsilon)} \quad (2)$$

where ε is the steepness parameter. It is known that dynamic associations can be realized in the associative memories composed chaotic neurons.

3. Multi Winners Self-Organizing Neural Network

Here, we briefly explain the Multi Winners Self-Organizing Neural Network (MWSOINN) (Huang & Hagiwara., 1997) which is used in the proposed model.

Figure 1 shows the structure of the MWSOINN. The MWSOINN consists of the Input/Output Layer and the Distributed Representation Layer, and each layer has 2- dimensional structure. In the learning process of the MWSOINN, the distributed representation patterns corresponding to input analog patterns are generated by the multi winners self-organizing process, and the connection weights between layers are trained by the error correction learning. In the recall process of the MWSOINN, when a stored pattern or a part of the stored patterns is given to the Input/Output Layer, the corresponding distributed representation pattern appears in the Distributed Representation Layer, and then the corresponding analog pattern appears in the Input/Output Layer.

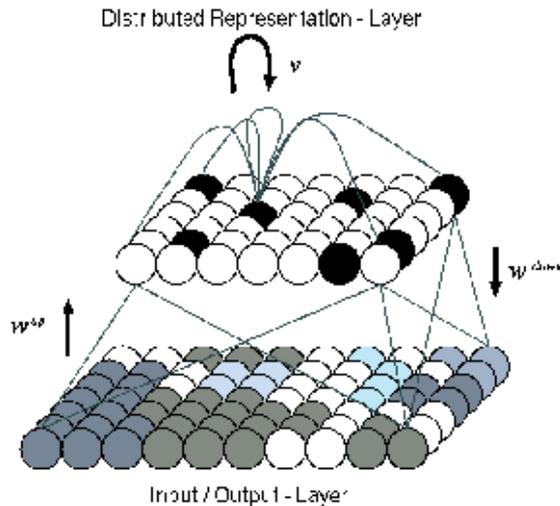


Fig. 1. Structure of MWSONN.

4. Improved Chaotic Associative Memory for Successive Learning

4.1 Outline of ICAMSL

Here, we explain the outline of the proposed Improved Chaotic Associative Memory for Successive Learning (ICAMSL). The proposed ICAMSL has three stages; (1) Pattern Search Stage, (2) Distributed Pattern Generation Stage and (3) Learning Stage.

When an unstored pattern set is given to the network, the proposed ICAMSL distinguishes an unstored pattern set from stored patterns and can learn the pattern set successively. When a stored pattern set is given, the ICAMSL recalls the patterns. When an unstored pattern set is given to the network, the ICAMSL changes the internal pattern for input pattern set by chaos and presents other pattern candidates (we call this the Pattern Search Stage). When the ICAMSL can not recall the desired patterns, the distributed pattern is generated by the multi-winners competition (Huang & Hagiwara., 1997) (Distributed Pattern Generation Stage), and it learns the input pattern set as an unstored pattern set (Learning Stage). Figure 2 shows the flow of the proposed ICAMSL.

4.2 Structure of ICAMSL

The proposed ICAMSL is a kind of the hetero-associative memories. Figure 3 shows a structure of the ICAMSL. This model has two layers; an Input/Output Layer (I/O Layer) composed of conventional neurons and a Distributed Representation Layer (DR Layer) composed of chaotic neurons (Aihara et al., 1990). In this model, there are the connection weights between neurons in the Distributed Representation Layer and the connection weights between the Input/Output Layer and the Distributed Representation Layer.

As shown in Fig.3, the Input/Output Layer has plural parts. The number of parts is decided depending on the number of patterns included in the pattern set. In the case of Fig.3, the Input/Output Layer consists of P parts corresponding to the patterns $1 \sim P$.

In this model, when a pattern set is given to the Input/Output Layer, the internal pattern corresponding to the input patterns is formed in the Distributed Representation Layer. Then,

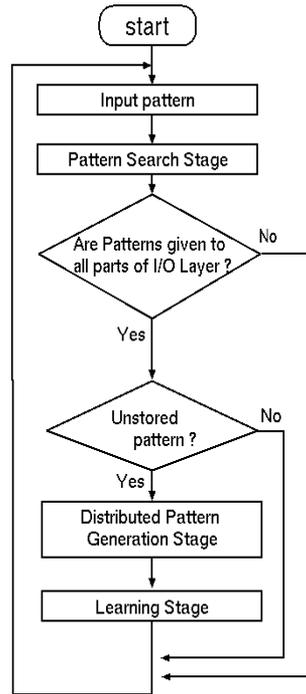


Fig. 2. Flow of Proposed ICAMSL.

in the Input/Output Layer, an output pattern set is generated from the internal pattern. The ICAMSL distinguishes an unstored pattern set from stored patterns by comparing the input patterns with the output pattern.

In this model, the output of the neuron i in the Distributed Representation Layer at the time $t+1$, $x_i(t+1)$ is given by the following equations.

$$x_i(t+1) = \phi[\xi_i(t+1) + \eta_i(t+1) + \zeta_i(t+1)] \quad (3)$$

$$\begin{aligned} \xi_i(t+1) &= \sum_{j=1}^M v_{ij} \sum_{d=0}^t k_s^d A_j(t-d) \\ &= k_s \xi_i(t) + \sum_{j=1}^M v_{ij} A_j(t) \end{aligned} \quad (4)$$

$$\begin{aligned} \eta_i(t+1) &= \sum_{j=1}^N w_{ij} \sum_{d=0}^t k_m^d x_j(t-d) \\ &= k_m \eta_i(t) + \sum_{j=1}^N w_{ij} x_j(t) \end{aligned} \quad (5)$$

$$\begin{aligned}\zeta_i(t+1) &= -\alpha \sum_{d=0}^t k_r^d x_i(t-d) - \theta_i \\ &= k_r \zeta_i(t) - \alpha x_i(t) - \theta_i(1 - k_r)\end{aligned}\quad (6)$$

In Eqs. (3)~(6), M is the number of neurons in the Input/Output Layer, v_{ij} is the connection weight between the neuron j in the Input/Output Layer and the neuron i in the Distributed Representation Layer, N is the number of neurons in the Distributed Representation Layer, $A_j(t)$ is the external input j to the Input/Output Layer at the time t , w_{ij} is the connection weight between the neuron i and the neuron j in the Distributed Representation Layer, $\alpha(t)$ is the scaling factor of the refractoriness at the time t , and k_s , k_m and k_r are the damping factors. $\phi(\cdot)$ is the following output function:

$$\phi(u_i) = \tanh(u_i / \varepsilon) \quad (7)$$

where ε is the steepness parameter.

The output of the neuron j in the Input/Output Layer at the time t , $x_j^{IO}(t)$ is given as follows.

$$x_j^{IO}(t) = \phi^{IO} \left(\sum_{i=1}^N v_{ij} x_i^D(t) \right) \quad (8)$$

$$\phi^{IO} = \begin{cases} 1, & u \geq 0 \\ -1, & u < 0 \end{cases} \quad (9)$$

4.3 Pattern Search Stage

In the Pattern Search Stage, when an input pattern set is given, the ICAMSL distinguishes the pattern set from stored patterns. When an unstored pattern set is given, the ICAMSL changes the internal pattern for the input pattern by chaos and presents the other pattern candidates. Until the ICAMSL recalls the desired patterns, the following procedures are repeated. If the ICAMSL can not recall the desired patterns, when the stage is repeated certain times, the ICAMSL finishes the stage.

4.3.1 Pattern Assumption

In the proposed ICAMSL, only when the input patterns are given to all parts of the Input/Output Layer, the patterns are judged. When the input pattern $A(t)$ is similar to the recalled pattern $x^{IO}(t)$, the ICAMSL can assume that input pattern is one of the stored patterns. The ICAMSL outputs the pattern formed by the internal pattern in the Distributed Representation Layer. The similarity $s(t)$ is defined by

$$s(t) = \frac{1}{M} \sum_{j=1}^M A_j(t) x_j^{IO}(t) \quad (10)$$

The ICAMSL regards the input patterns as a stored pattern set, when the similarity rate $s(t)$ is larger than the threshold s^{th} ($s(t) \geq s^{th}$).

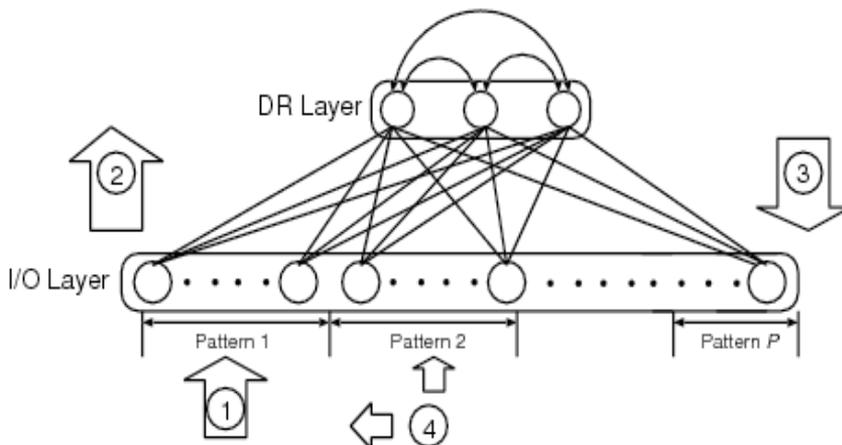


Fig. 3. Structure of Proposed ICAMSL.

4.3.2 Pattern Search

When the ICAMSL assumes that the input patterns are an unstored pattern set, the ICAMSL changes the internal pattern $x^D(t)$ for the input pattern by chaos and presents the other pattern candidates.

In the chaotic neural network, it is known that dynamic association can be realized if the scaling factor of the refractoriness $\alpha(t)$ is suitable (Aihara et al., 1990). Therefore, in the proposed model, $\alpha(t)$ is changed as follows:

$$\alpha(t) = ((\alpha_{\max}(t) - \alpha_{\min})(1 - s(t)) + \alpha_{\min}) / \alpha_{DIV} \quad (11)$$

$$\alpha_{\max}(t) = Mv_{\max} + Nw_{\max} \quad (12)$$

$$v_{\max} = \max\{|v_{11}|, \dots, |v_{ij}|, \dots, |v_{NM}|\} \quad (13)$$

$$w_{\max} = \max\{|w_{11}|, \dots, |w_{i'j'}|, \dots, |w_{NN}|\} \quad (14)$$

where α_{\min} is the minimum of α , $\alpha_{\max}(t)$ is the maximum of α at the time t , $s(t)$ is the similarity between the input pattern and the output pattern at the time t (the time when the Pattern Search Stage started), and α_{DIV} is the constant.

4.4 Distributed Pattern Generation Stage

In the Distributed Pattern Generation Stage, the distributed pattern corresponding to the input pattern is generated by the multi-winners competition (Huang & Hagiwara, 1997).

4.4.1 Calculation of Outputs of Neurons in I/O Layer

When the input pattern $A_j(t)$ is given to the Input/Output Layer, the output of the neuron j in the Input/Output Layer x_j^{IO} is given by

$$x_j^{IO} = S_f(A_j(t)) \quad (15)$$

where $S_f(\cdot)$ is the ramp function and is given by

$$S_f(u) = \begin{cases} u, & u > 0 \\ 0, & u \leq 0 \end{cases} \quad (16)$$

4.4.2 Calculation of Initial Outputs of Neurons in DR Layer

The output of the neuron i in the Distributed Representation Layer $x_i^{D(0)}$ is calculated by

$$x_i^{D(0)} = C_f \left(\sum_{j=1}^M v_{ij} x_j^{IO} - \theta_i \right) \quad (17)$$

where v_{ij} is the connection weight from the neuron j in the Input/Output Layer to the neuron i in the Distributed Representation Layer, θ_i is the threshold of the neuron i in the Distributed Representation Layer and M is the number of neurons in the Input/Output Layer. The output function $C_f(\cdot)$ is given by

$$C_f(u) = \tanh(u/T) \quad (18)$$

where T is the steepness parameter in the sigmoidal function.

4.4.3 Competition between Neurons in DR Layer

The competition dynamics is given by the following equations:

$$x_i^D = C_f(u_i - \theta_i) \quad (19)$$

$$u_i = \sum_{i'=1}^N w_{ii'} x_{i'}^D \quad (20)$$

where x_i^D is the output of the neuron i in the Distributed Representation Layer, u_i is the internal state of the neuron i in the Distributed Representation Layer and N is the number of neurons in the Distributed Representation Layer.

4.5 Learning Stage

In the Pattern Search Stage, if the ICAMSL can not recall the desired pattern set, it learns the input pattern set as an unstored pattern set. The Learning Stage has two phases; (1) Hebbian

Learning Phase and (2) anti-Hebbian Learning Phase. If the signs of the outputs of two neurons are same, the connection weight between these two neurons is strengthened. By this learning, the connection weights are changed to learn the input patterns, however the Hebbian learning can only learn a new input pattern set. In the proposed ICAMSL, the anti-Hebbian Learning Phase is employed as similar as the original HCAMSL (Arai & Osana, 2006). In the anti-Hebbian Learning Phase, the connection weights are changed in the opposite direction in the case of the Hebbian Learning Phase. The proposed ICAMSL can learn a new pattern set without destroying the stored patterns by the anti-Hebbian Learning. Figure 4 shows the learning stage of the proposed ICAMSL.

4.5.1 Hebbian Learning Phase

In the Hebbian Learning Phase, until the similarity rate $s(t)$ becomes 1.0, the update of the connection weights is repeated.

The connection weight between the Input/Output Layer and the Distributed Representation Layer v_{ij} and the connection weight in the Distributed Representation Layer $w_{ii'}$ are updated as follows:

$$v_{ij}^{(new)} = v_{ij}^{(old)} + \gamma_v^+ x_i^{D(comp)} A_j(t) \quad (21)$$

$$w_{ii'}^{(new)} = w_{ii'}^{(old)} + \gamma_w^+ x_i^{D(comp)} x_{i'}^{D(comp)} \quad (22)$$

where γ_v^+ is the learning rate of the connection weight v_{ij} in the Hebbian Learning Phase, and γ_w^+ is the learning rate of the connection weight $w_{ii'}$ in this phase.

4.5.2 Give Up Function

When the similarity rate $s(t)$ does not become 1.0 even if the connection weights are updated T_n times, the ICAMSL gives up to memorize the pattern set. If the ICAMSL gives up to learn the pattern set, the anti-Hebbian Learning Phase is not performed.

4.5.3 Anti-Hebbian Learning Phase

The anti-Hebbian Learning Phase is performed after the Hebbian Learning Phase. In this phase, the connection weight v_{ij} and $w_{ii'}$ are changed in the opposite direction in the case of the Hebbian Learning Phase. The anti-Hebbian Learning makes the relation between the patterns is learned without destroying the stored patterns.

In this phase, v_{ij} and $w_{ii'}$ are updated by

$$v_{ij}^{(new)} = v_{ij}^{(old)} + \gamma_v^- x_i^{D(comp)} A_j(t) \quad (23)$$

$$w_{ii'}^{(new)} = w_{ii'}^{(old)} + \gamma_w^- x_i^{D(comp)} x_{i'}^{D(comp)} \quad (24)$$

where $\gamma_v^-(\gamma_v^- > \gamma_v^+ > 0)$ is the learning rate of the connection weight v_{ij} in the anti-Hebbian Learning Phase, and $\gamma_w^-(\gamma_w^- > \gamma_w^+ > 0)$ is the learning rate of the connection weight w_{ij} in this phase.

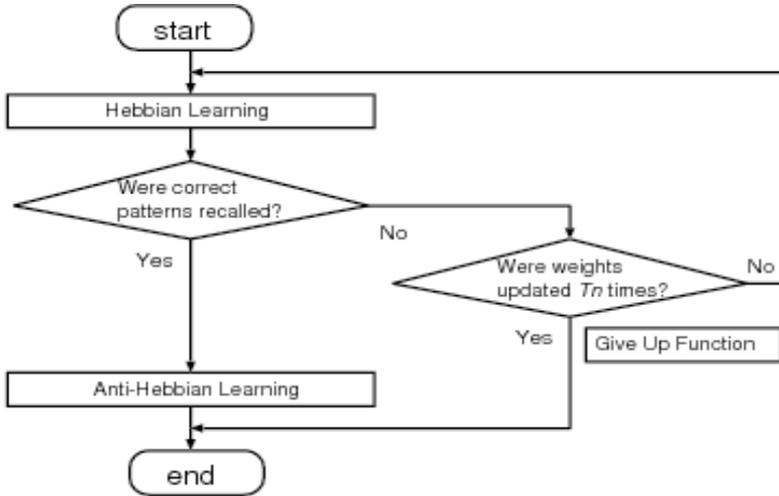


Fig. 4. Learning Stage.

5. Computer Experiment Results

In this section, we show the computer experiment results to demonstrate the effectiveness of the proposed ICAMSL. The computer experiments were carried out under the conditions shown in Table 1.

5.1 Successive Learning and One-to-Many Associations

Figure 5 shows the successive learning and one-to-many associations in the proposed ICAMSL. As seen in Fig.5, the patterns "lion", "mouse" and "penguin" were given to the network at $t=1$. At $t=1$, the ICAMSL could not recall the correct patterns because no pattern was stored in the network. During $t=2 \sim 11$, the ICAMSL changed the internal patterns by chaos and presented the other pattern candidates, however it could not recall the correct patterns. As a result, the ICAMSL regarded the input patterns as unstored patterns, at $t=13$, the patterns "lion", "mouse" and "penguin" were trained as new patterns. At $t=14$, the patterns "lion", "mouse" and "duck" were given to the network. At this time, since only the pattern set "lion", "mouse" and "penguin" was memorized in the network, the ICAMSL recalled the patterns "lion", "mouse" and "penguin". During $t=15 \sim 24$, the ICAMSL changed the internal patterns by chaos and presented the other pattern candidates, however it could not recall the correct patterns. As a result, the ICAMSL regarded the input patterns as unstored patterns, at $t=28$, the "lion", "mouse" and "duck" were trained as new patterns. At $t=29$, the patterns "lion" and "mouse" were given to the network, the ICAMSL recalled "lion", "mouse" and "penguin" ($t=30$) and "lion", "mouse" and "duck" ($t=35$). From these results, we confirmed that the proposed ICAMSL can learn patterns successively and realize one-to-many associations.

Learning Parameters		
the number of pattern searches in Pattern Search Stage		10
initial value of all connection weights		-1.0 ~ 1.0
learning rate in Hebbian Learning	γ_v^+, γ_w^+	1.0
learning rate in anti-Hebbian Learning	γ_v^-, γ_w^-	2.0
threshold of similarity rate	s^{th}	1.0
Chaotic Neuron Parameters		
constant for refractoriness	α_{DIV}	25
minimum of scaling factor α	α_{min}	0.0
damping factor	k_s	0.5
damping factor	k_m	0.0
damping factor	k_r	0.95
threshold of neurons	θ	0.0
steepness parameter	ε	1.0
Competition Parameters		
steepness parameter	T	0.0005

Table 1. Experimental Conditions.

5.2 Storage Capacity

Here, we examined the storage capacity of the proposed ICAMSL. In this experiment, we used the ICAMSL which has 800 neurons (400 neurons for pattern 1 and 400 neurons for pattern 2) in the Input/Output Layer and 225 neurons in the Distributed Representation Layer. We used random patterns to store and Fig.6 shows the average of 100 trials. In this figure, the horizontal axis is the number of stored pattern pairs, and the vertical axis is the perfect recall rate. In this figure, the storage capacities of the model without give up function (HCAMSL-MW) (Ando et al., 2006), the model without the Distributed Pattern Generation Stage (HCAMSL) (Arai & Osana, 2006) and the model without the give up function and the Distributed Pattern Generation Stage are also shown for reference.

From these results, we confirmed that the storage capacity of the proposed ICAMSL is larger than that of the conventional HCAMSL/HCAMSL-MW.

6. Conclusions

In this research, we have proposed the Improved Chaotic Associative Memory for Successive Learning (ICAMSL). The proposed model is based on the Hetero Chaotic Associative Memory for Successive Learning with give up function (HCAMSL) (Arai & Osana, 2006) and the Hetero Chaotic Associative Memory for Successive Learning with Multi-Winners competition (HCAMSL-MW) (Ando et al., 2006). In the proposed ICAMSL, the learning process and recall process are not divided. When an unstored pattern is given to the network, the ICAMSL can learn the pattern successively. We carried out a series of computer experiments and confirmed that the proposed ICAMSL can learn patterns

successively and realize one-to-many associations, and the storage capacity of the ICAMSL is larger than that of the conventional HCAMSL/HCAMSL-MW.

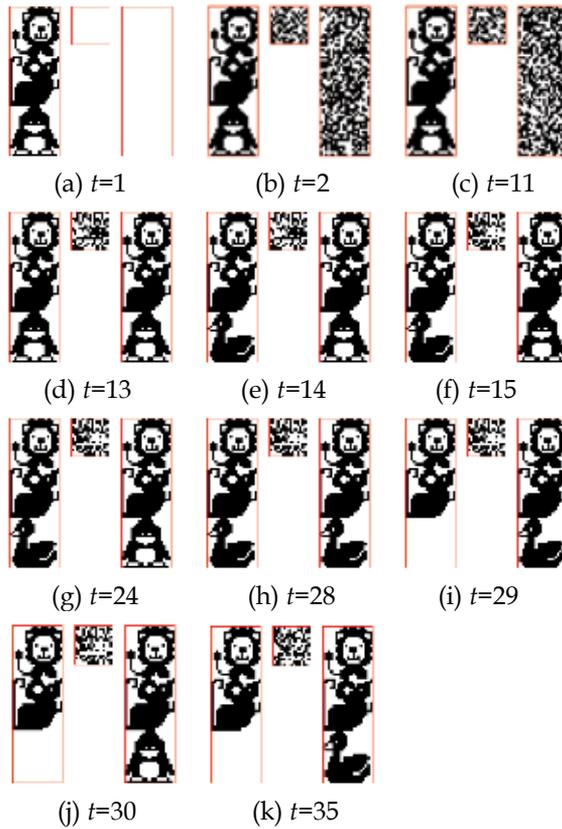


Fig. 5. Successive Learning in Proposed Model.

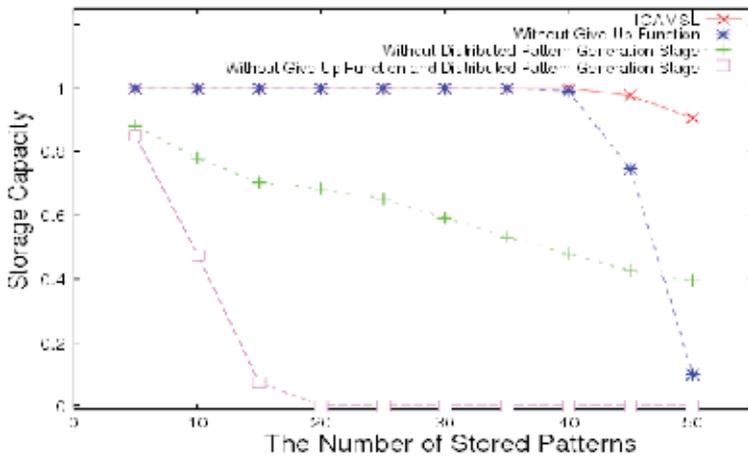


Fig. 6. Storage Capacity.

7. References

- K. Aihara, T. Takabe and M. Toyoda. (1990). Chaotic neural networks, *Physics Letter A*, Vol.144, No.6, 7, pp.333-340
- M. Ando, Y. Okuno and Y. Osana. (2006). Hetero chaotic associative memory for successive learning with multi-winners competition, *Proceedings of IEEE and INNS International Joint Conference on Neural Networks*, Vancouver
- T. Arai and Y. Osana. (2006). Hetero chaotic associative memory for successive learning with give up function – One-to-many associations –, *Proceedings of IASTED Artificial Intelligence and Applications*, Innsbruck
- G. A. Carpenter and S. Grossberg. (1995). *Pattern Recognition by Self-organizing Neural Networks*, The MIT Press
- J. J. Hopfield. (1982). Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the United States of America*, 79, pp. 2554-2558
- J. T. Huang and M. Hagiwara. (1997). A multi-winners selforganizing neural network, *IEEE International Conference on System, Man and Cybernetics*, pp. 2499-2504
- M. Ideguchi, N. Sato and Y. Osana. (2005). Hetero chaotic associative memory for successive learning and action study of robot, *Proceedings of International Symposium on Nonlinear Theory and its Applications*, Bruges
- N. Kawasaki, Y. Osana and M. Hagiwara. (2000). Chaotic associative memory for successive learning using internal patterns, *IEEE International Conference on Systems, Man and Cybernetics*
- T. Kohonen. (1994). *Self-Organizing Maps*, Springer
- B. Kosko. (1988). Bidirectional associative memories, *IEEE Transactions on System, Man and Cybernetics*, SMC-18, No. 1, pp. 49-60
- Y. Osana and M. Hagiwara. (1999). Successive learning in chaotic neural network, *International Journal of Neural Systems*, Vol.9, No.4, pp.285-299
- D. E. Rumelhart, J. L. McClelland and the PDP Research Group. (1986). *Parallel Distributed Processing, Exploitations in the Microstructure of Cognition*, Vol.11 : Foundations, The MIT Press
- M.Watanabe, K. Aihara and S. Kondo. (1995). Automatic learning in chaotic neural networks, *Institute of Electronics, Information and Communication Engineers-A*, Vol.J78-A, No.6, pp.686-691

Kohonen Feature Map Associative Memory with Refractoriness based on Area Representation

Tomohisa IMABAYASHI, Yuko OSANA
Tokyo University of Technology
Japan

1. Introduction

Recently, neural networks are drawing much attention as a method to realize flexible information processing. Neural networks consider neuron groups of the brain in the creature, and imitate these neurons technologically. Neural networks have some features, especially one of the important features is that the networks can learn to acquire the ability of information processing.

In the field of neural networks, a lot of models have been proposed such as the Back Propagation algorithm (Rumelhart et al., 1986), the Kohonen Feature Map (KFM) (Kohonen, 1994), the Hopfield network (Hopfield, 1994), and the Bidirectional Associative Memory (Kosko, 1988). In these models, the learning process and the recall process are divided, and therefore they need all information to learn in advance.

However, in the real world, it is very difficult to get all information to learn in advance, so we need the model whose learning process and recall process are not divided. As such model, Grossberg and Carpenter proposed the ART (Adaptive Resonance Theory) (Carpenter & Grossberg, 1995). However, the ART is based on the local representation, and therefore it is not robust for damaged neurons in the Map-Layer. While in the field of associative memories, some models have been proposed (Watanabe et al., 1995; Osana & Hagiwara, 1999; Kawasaki et al., 2000). Since these models are based on the distributed representation, they have the robustness for damaged neurons. However, their storage capacities are small because their learning algorithm is based on Hebbian learning.

On the other hand, the Kohonen Feature Map associative memory (KFM associative memory) (Ichiki et al., 1993) has been proposed. Although the KFM associative memory is based on the local representation as similar as the ART (Carpenter & Grossberg, 1995), it can learn new patterns successively (Yamada et al., 1999), and its storage capacity is larger than that of models in refs.(Watanabe et al., 1995; Osana & Hagiwara, 1999; Kawasaki et al., 2000). It can deal with auto and hetero associations and the associations for plural sequential patterns including common terms (Hattori et al., 2002). Moreover, the KFM associative memory with area representation (Abe & Osana, 2006) has been proposed. In the model, the area representation (Ikeda & Hagiwara, 1997) was introduced to the KFM associative memory, and it has robustness for damaged neurons. However, it can not deal with one-to-many associations and associations of analog patterns.

In this research, we propose a Kohonen Feature Map Associative Memory with Refractoriness based on Area Representation (KFMAM-R-AR). The proposed model is based on the KFM associative memory with area representation (Abe & Osana, 2006) and the neurons in the Map-Layer have refractoriness. In the proposed model, one-to-many associations are realized by refractoriness of neurons. Moreover, by improvement of the calculation of the internal states of the neurons in the Map-Layer, the proposed model has enough robustness for damaged neurons when analog patterns are memorized.

2. KFM Associative Memory with Area Representation

Here, we briefly review the KFM Associative Memory with Area Representation (KFMAM-AR) (Abe & Osana, 2006) which is used in the proposed model.

2.1 Structure

Figure 1 shows the structure of the KFM associative memory with area representation (Abe & Osana, 2006). As shown in Fig.1, the KFMAM-AR has two layers; (1) Input/Output (I/O)-Layer and (2) Map-Layer, and the I/O-Layer is divided into some parts.

In the KFMAM-AR, since one concept is expressed by the winner neuron and some neurons located adjacent to the winner neuron, it has the robustness for damaged neurons in the Map-Layer.

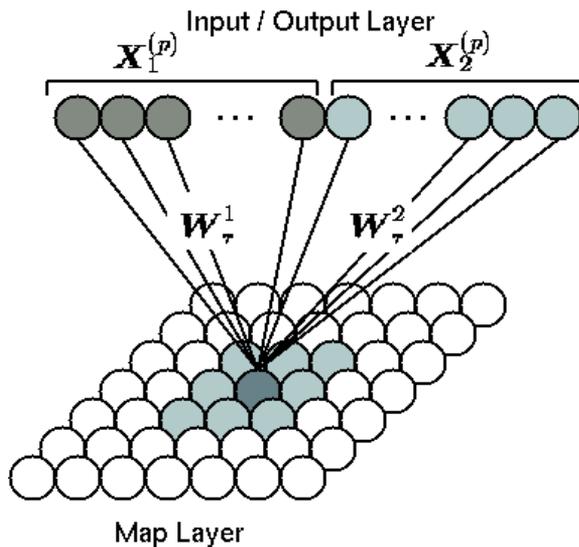


Fig. 1. Structure of KFMAM-AR.

2.2 Learning Process

The learning algorithm for the KFMAM-AR is based on the conventional sequential learning algorithm for the KFM associative memory (Ichiki et al., 1993).

Let us consider the case where the I/O-Layer composed of N parts corresponding to the pattern $X_1^{(p)}, X_2^{(p)}, \dots, X_N^{(p)}$ in the KFMAM-AR. In this case, the learning vector $X^{(p)}$ is given by

$$\mathbf{X}^{(p)} = \begin{pmatrix} \mathbf{X}_1^{(p)} \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \mathbf{X}_2^{(p)} \\ \vdots \\ 0 \end{pmatrix} + \dots + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mathbf{X}_N^{(p)} \end{pmatrix} \tag{1}$$

where $\mathbf{X}^{(p)} \in \{0,1\}^M$, $\mathbf{X}_j^{(p)} \in \{0,1\}^{M_j}$ and M_j is the number of neurons corresponding to the j th partial pattern in the $\mathbf{X}^{(p)}$. In addition, M is the number of neurons in the I/O-Layer. Let \mathbf{W}_i be the connection weights between the neurons in the I/O-Layer and the neuron i in the Map-Layer:

$$\mathbf{W}_i = \begin{pmatrix} \mathbf{W}_i^1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \dots + \begin{pmatrix} 0 \\ \vdots \\ \mathbf{W}_i^j \\ \vdots \\ 0 \end{pmatrix} + \dots + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mathbf{W}_i^N \end{pmatrix} \tag{2}$$

where \mathbf{W}_i^j is the connection weight between the neuron i in the Map-Layer and the neurons of the j th part in the I/O-Layer.

In the sequential learning algorithm for the KFMAM-AR, the connection weights are learned as follows:

- (1) The initial values of weights are chosen randomly.
- (2) The Euclid distance between the learning vector $\mathbf{X}^{(p)}$ and the connection weights vector \mathbf{W}_i , $d(\mathbf{X}^{(p)}, \mathbf{W}_i)$ is calculated by

$$d(\mathbf{X}^{(p)}, \mathbf{W}_i) = \sqrt{\sum_{k=1}^M (X_k^{(p)} - W_{ik})^2} . \tag{3}$$

- (3) The winner neuron r whose Euclid distance is minimum is found.

$$r = \underset{i}{\operatorname{argmin}} d(\mathbf{X}^{(p)}, \mathbf{W}_i) \tag{4}$$

- (4) If $d(\mathbf{X}^{(p)}, \mathbf{W}_i) > \theta^l$, the connection weights of the winner neuron r are fixed. The connection weights except those of fixed neurons are changed by

$$\mathbf{W}_i(t+1) = \mathbf{W}_i(t) + H(d_i) \eta(t) h_{r_i}(\mathbf{X}^{(p)} - \mathbf{W}_i(t)) \tag{5}$$

where h_{r_i} is the neighborhood function and is given by

$$h_{ri} = \exp\left(\frac{-\|r - i\|^2}{2\sigma(t)^2}\right) \quad (6)$$

where $\sigma(t)$ is the following decreasing function:

$$\sigma(t) = \sigma_i \left(\frac{\sigma_f}{\sigma_i}\right)^{\frac{t}{T}} \quad (7)$$

In this equation, σ_i is the initial value of $\sigma(t)$ and $\sigma(t)$ varies from σ_i to σ_f ($\sigma_i > \sigma_f$). T is the upper limit of the learning iterations.

In Eq.(5), $\eta(t)$ is the learning rate and is given by

$$\eta(t) = \frac{-\eta_0(t-T)}{T} \quad (8)$$

where η_0 is the initial value of $\eta(t)$, and $H(d_i)$ is calculated by

$$H(d_i) = \frac{1}{1 + \left(-\frac{d_i - D}{\varepsilon}\right)} \quad (9)$$

In this equation, d_i is the Euclid distance between the neuron i and the nearest weights fixed neuron in the Map-Layer, D is the constant and ε is the steepness parameter of the function $H(d_i)$. Owing to $H(d_i)$, weights of neurons close to the fixed neurons are semi-fixed, that is, they become hard to be learned.

(5) (2)~(9) are iterated until $d(\mathbf{X}^{(p)}, \mathbf{W}_i) \leq \theta^l$ is satisfied.

(6) The connection weights of the winner neuron r , W_r are fixed.

(7) (2)~(6) are iterated when a new pattern set is given.

2.3 Recall Process

Since the conventional KFM associative memory is based on the local representation, it has not the robustness for damaged neurons in the Map-Layer. In contrast, the KFMAM-AR is based on the area representation (Ikeda & Hagiwara, 1997) and has the robustness for damaged neurons. The area representation is an intermediate representation of the local representation and the distributed representation. In the area representation, one concept is expressed by the winner neuron and some neurons located adjacent to the winner neuron.

2.3.1 Recall Process for Binary Patterns

In the recall process of the KFMAM-AR, when the binary pattern X is given to the I/O-

Layer, the output of the neuron i in the Map-Layer x_i^{map} is calculated by

$$x_i^{map} = \begin{cases} 1, & \text{if } d(\mathbf{X}, \mathbf{W}_i) < \theta_b^{map} \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where θ_b^{map} is the threshold of the neuron in the Map-Layer as follows:

$$\theta_b^{map} = d_{min} + a(d_{max} - d_{min}) \quad (11)$$

$$d_{min} = \min_i d(\mathbf{X}, \mathbf{W}_i) \quad (12)$$

$$d_{max} = \max_i d(\mathbf{X}, \mathbf{W}_i) \quad (13)$$

In Eq.(11), a ($0 < a < 0.5$) is the coefficient.

Then, the output of the neuron k in the I/O-Layer x_k^{in} is calculated as follows:

$$x_k^{in} = \begin{cases} 1, & \text{if } u_k^{in} \geq \theta_b^{in} \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

$$u_k^{in} = \frac{1}{\sum_i x_i^{map}} \sum_{i: x_i=1} W_{ik} \quad (15)$$

where θ_b^{in} is the threshold of the neuron in the I/O-Layer, u_k^{in} is the internal state of the neuron k in the I/O-Layer.

2.3.2 Recall Process for Analog Patterns

In the recall process of the KFM-AR, when the analog pattern \mathbf{X} is given to the I/O-Layer, the output of the neurons i in the Map-Layer x_i^{map} is calculated by

$$x_i^{map} = \begin{cases} 1, & \text{if } d(\mathbf{X}, \mathbf{W}_i) < \theta_a \\ 0, & \text{otherwise} \end{cases} \quad (16)$$

where θ_a is the threshold of the neuron in the Map-Layer.

Then, the output of the neuron k in the I/O-Layer x_k^{in} is calculated as follows:

$$x_k^{in} = \frac{1}{\sum_i x_i^{map}} \sum_{i: x_i=1} W_{ik} \quad (17)$$

3. KFM Associative Memory with Refractoriness based on Area Representation

The conventional KFM associative memory (Ichiki et al., 1993) and KFMAM-AR (Abe & Osana, 2006) cannot realize one-to-many associations. In this paper, we propose the Kohonen Feature Map Associative Memory with Refractoriness based on Area Representation (KFMAM-R-AR) which can realize one-to-many associations. The proposed model is based on the KFMAM-AR, and the neurons in the Map-Layer have refractoriness. In the proposed model, one-to-many associations are realized by the refractoriness of neurons.

On the other hand, although the conventional KFMAM-AR can realize associations for analog patterns, it does not have enough robustness for damaged neurons. In this research, the model which has enough robustness for damaged neurons when analog patterns are memorized is realized by improvement of the calculation of the internal states of neurons in the Map-Layer.

3.1 Learning Process

In the proposed model, the patterns are trained by the learning algorithm of the KFMAM-AR described in 2.2.

3.2 Recall Process

In the recall process of the proposed model, when the pattern X is given to the I/O-Layer, the output of the neuron i in the Map-Layer x_i^{map} is calculated by

$$x_i^{map}(t) = H^{recall}(d(\mathbf{r}, \mathbf{i})) f(u_i^{map}(t)) \quad (18)$$

$$H^{recall}(d(\mathbf{r}, \mathbf{i})) = \frac{1}{1 + \exp\left(\frac{d(\mathbf{r}, \mathbf{i}) - D}{\varepsilon}\right)} \quad (19)$$

where D is the constant which decides area size, ε is the steepness parameter. $d(\mathbf{r}, \mathbf{i})$ is the Euclid distance between the winner neuron r and the neuron i and is calculated by

$$r = \underset{i}{\operatorname{argmax}} u_i^{map}(t) \quad (20)$$

Owing to $H^{recall}(d(\mathbf{r}, \mathbf{i}))$, the neurons which are far from the winner neuron become hard to fire. $f(u_i^{map}(t))$ is calculated by

$$f(u_i^{map}(t)) = \begin{cases} 1, & \text{if } u_i^{map}(t) > \theta^{map} \text{ and } u_i^{map}(t) > \theta^{min} \\ 0, & \text{otherwise} \end{cases} \quad (21)$$

where $u_i^{map}(t)$ is the internal state of the neuron i in the Map-Layer at the time t , θ^{map} and θ^{min} are the thresholds of the neuron in the Map-Layer. θ^{map} is calculated as follows:

$$\theta^{map} = u_{min} + a(u_{max} - u_{min}) \tag{22}$$

$$u_{min} = \min_i u_i^{map}(t) \tag{23}$$

$$u_{max} = \max_i u_i^{map}(t) \tag{24}$$

where $a(0.5 < a < 1)$ is the coefficient.

In Eq.(18), when the binary pattern X is given to the I/O-Layer, the internal state of the neuron i in the Map-Layer at the time t , $u_i^{map}(t)$ is calculated by

$$u_i^{map}(t) = 1 - \frac{d^{in}(X, W_i)}{\sqrt{N^{in}}} - \alpha \sum_{d=0}^t k_r^d x_i^{map}(t-d) \tag{25}$$

where $d^{in}(X, W_i)$ is the Euclid distance between the input pattern X and the connection weights W_i . In the recall process, since all neurons in the I/O-Layer not always receive the input, the distance for the part where the pattern was given is calculated as follows:

$$d^{in}(X, W_i) = \sqrt{\sum_{k \in C} (X_k - W_{ik})^2} \tag{26}$$

where C shows the set of the neurons in the I/O-Layer which receive the input. In Eq.(25), N^{in} is the number of neurons which receive the input in the I/O-Layer, α is the scaling factor of the refractoriness and $k_r(0 \leq k_r < 1)$ is the damping factor. The output of the neuron k in the I/O-Layer at the time t , $x_k^{in}(t)$ is calculated by

$$x_k^{in}(t) = \begin{cases} 1, & \text{if } u_k^{in}(t) \geq \theta_b^{in} \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

$$u_k^{in}(t) = \frac{1}{\sum_i x_i^{map}(t)} \sum_{i: x_i > \theta^{out}} W_{ik} \tag{28}$$

where θ^{in} is the threshold of the neuron in the I/O-Layer, θ^{out} is the threshold for the

output of the neuron in the Map-Layer.

On the other hand, when the analog pattern is given to the I/O-Layer at the time t , $u_i^{map}(t)$ is calculated by

$$u_i^{map}(t) = \frac{1}{N^{in}} \sum_{k \in C}^{N^{in}} g(X_k - W_{ik}) - \alpha \sum_{d=0}^t k_r^d x_i^{map}(t-d). \quad (29)$$

Here, $g(\cdot)$ is calculated as follows:

$$g(b) = \begin{cases} 1, & |b| < \theta^b \\ 0, & \text{otherwise} \end{cases} \quad (30)$$

where θ^b is the threshold.

In the conventional KFMAM-AR, the neurons whose Euclid distance between the input vector and the connection weights are not over the threshold fire. In contrast, in the proposed model, the neurons which have many elements whose difference between the weight vector and the input vector are small can fire. The output of the neuron k in the I/O-Layer at the time t , $x_k^{in}(t)$ is calculated as follows:

$$x_k^{in}(t) = \frac{1}{\sum_i x_i^{map}(t)} \sum_{i: x_i^{map} > \theta^{out}} W_{ik}. \quad (31)$$

4. Computer Experiment Results

In this section, we show the computer experiment results to demonstrate the effectiveness of the proposed model. Table 1 shows the experimental conditions.

4.1 Association Result for Binary Patterns

Here, we show the association result of the proposed model for binary patterns. In this experiment, the number of neurons in the I/O-Layer was set to 800(= 400 × 2) and the number of neurons in the Map-Layer was set to 400. Figure 2 (a) shows an example of stored binary pattern pairs.

Figure 3 shows the association result of the proposed model when “lion” was given. As shown in this figure, the proposed model could realize one-to-many associations.

4.2 Association Result for Analog Patterns

Here, we show the association result of the proposed model for analog patterns. In this experiment, the number of neurons in the I/O-Layer was set to 800(= 400 × 2) and the number of neurons in the Map-Layer was set to 400. Figure 2 (b) shows an example of stored analog pattern pairs.

Figure 4 shows the association result of the proposed model when “lion” was given. As shown in this figure, the proposed model could realize one-to-many associations for analog patterns.

4.3 Storage Capacity

Here, we examined the storage capacity of the proposed model. In this experiment, we used the proposed model which has 800(= 400 × 2) neurons in the I/O-Layer and 400/800 neurons in the Map-Layer. We used random patterns and Figs.5 and 6 show the average of 100 trials. In these figures, the horizontal axis is the number of stored pattern pairs, and the vertical axis is the storage capacity. As shown in these figures, the storage capacity of the proposed model for the training set including one-to-many relations is as large as that for the training set including only one-to-one relations.

Parameters for learning		
threshold(learning)	θ^l	10^{-7}
initial value of η	η_0	0.1
initial value of σ	σ_i	3.0
last value of σ	σ_f	0.5
steepness parameter	ε	0.01
coefficient (range of semi-fixed)	D	3.0
Parameters for recall (common)		
scaling factor of refractoriness	α	1.0
damping factor	k_r	0.9
steepness of H^{recall}	ε	0.01
coefficient (size of area)	D	3.0
threshold (minimum)	θ^{min}	0.5
threshold (output)	θ^{out}	0.99
Parameters for recall (binary)		
coefficient (threshold)	a	0.9
threshold in the I/O-Layer	θ_b^{in}	0.5
Parameter for recall (analog)		
threshold (difference)	θ^d	0.1

Table 1. Experimental Conditions.

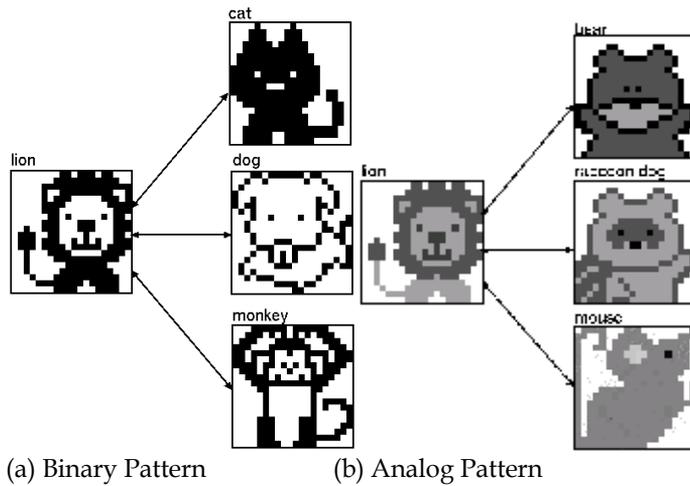


Fig. 2. An Example of Stored Patterns.

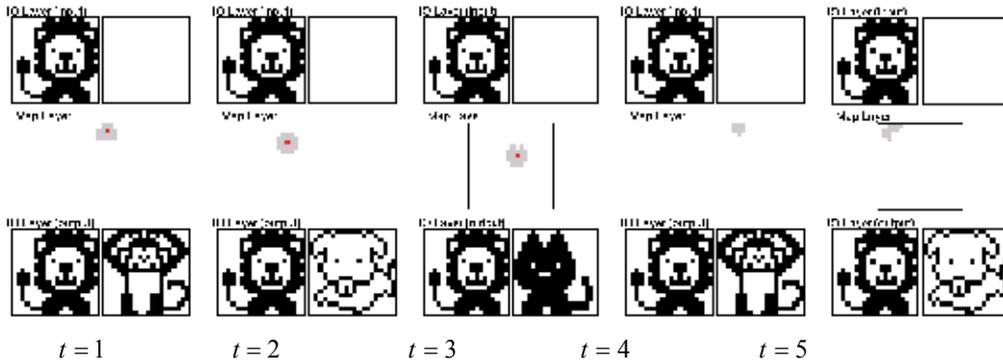


Fig. 3. Association Result for Binary Patterns.

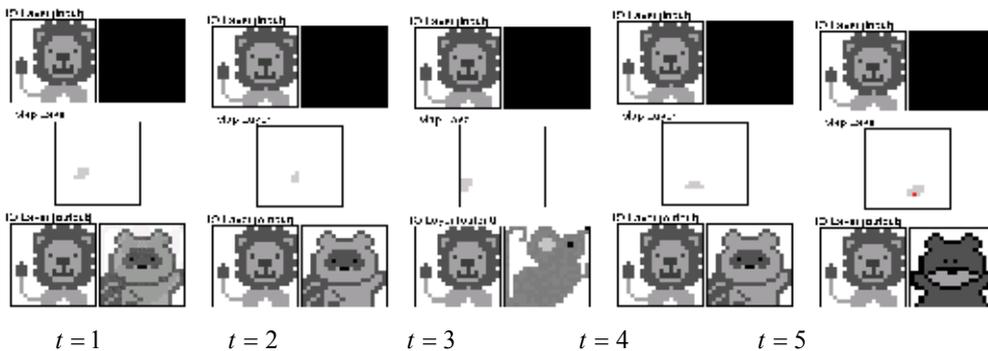


Fig. 4. Association Result for Analog Patterns.

4.4 Recall Ability for One-to-Many Associations

Here, we examined the recall ability in one-to-many associations of the proposed model. In

this experiment, we used the proposed model which has 800(= 400 × 2) neurons in the I/O-Layer and 400 neurons in the Map-Layer. We used one-to-P (P = 1, 2, ···, 30) random patterns and Fig.7 shows the average of 100 trials. In Fig.7, the horizontal axis is the number of stored pattern pairs, and the vertical axis is the recall rate. As shown in Fig.7, the proposed model could recall all patterns when P is smaller than 15 (binary patterns) / 4 (analog patterns). Although the proposed model could not recall all patterns corresponding to the input when P was 30, it could recall about 25 binary patterns / 17 analog patterns.

4.5 Noise Reduction Effect

Here, we examined the noise reduction effect of the proposed model. Figure 8 shows the noise sensitivity of the proposed model for analog patterns. In this experiment, we used the proposed model which has 800(= 400 × 2) neurons in the I/O-Layer and 400 neurons in the Map-Layer and 9 random analog patterns (three sets of patterns in one-to-three relations) were stored. Figure 8 shows the average of 100 trials.

In the proposed model, the minimum threshold of the neurons in the Map-Layer θ^{min} influences the noise sensitivity. As shown in Fig.8, we confirmed that the proposed model is more robust for noisy input when θ^{min} is small.

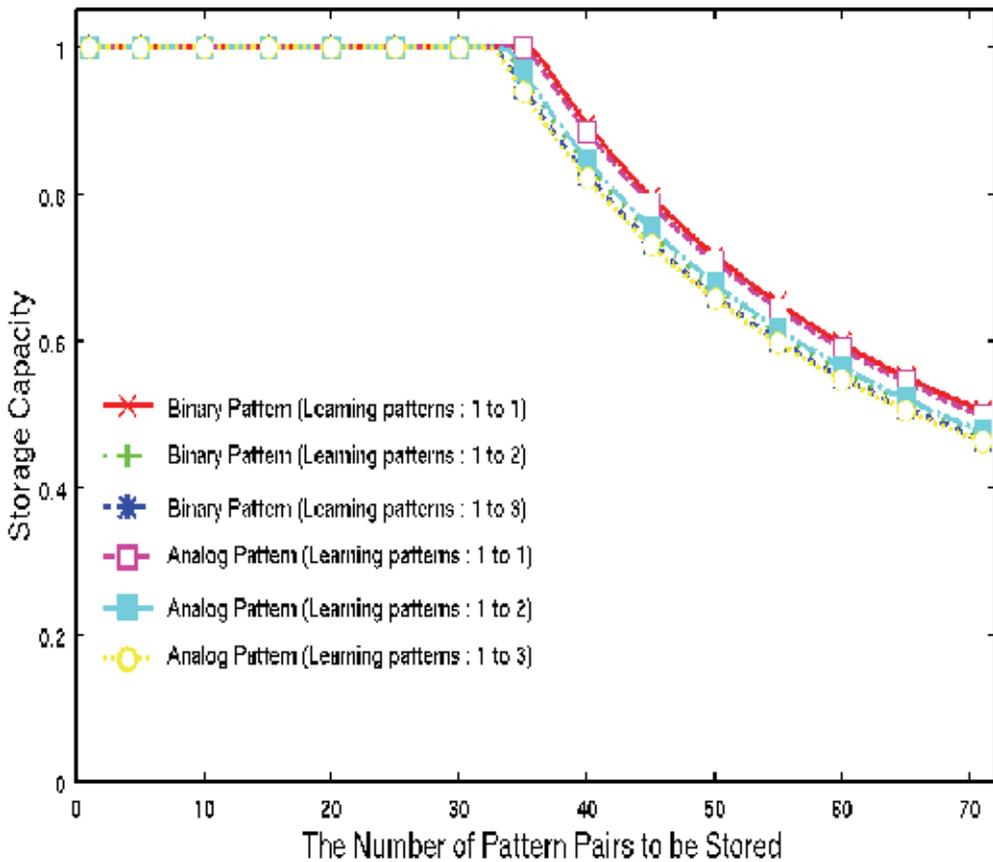


Fig. 5. Storage Capacity (400 neurons in the Map-Layer).

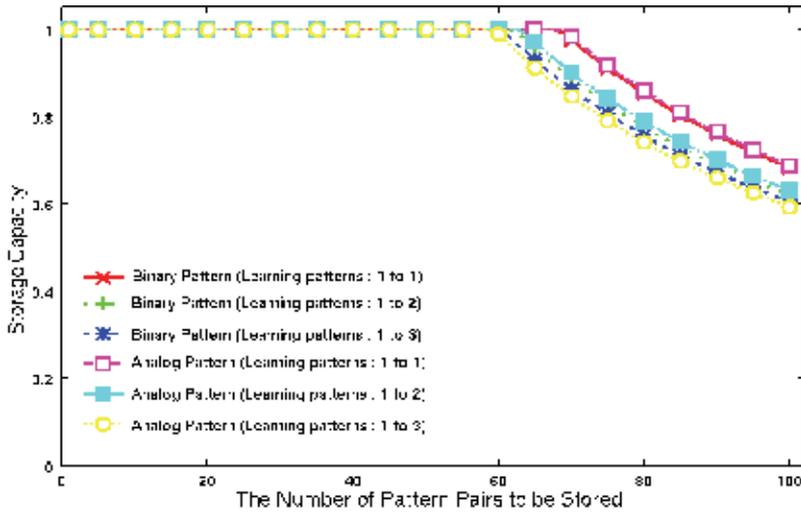


Fig. 6. Storage Capacity (800 neurons in the Map-Layer).

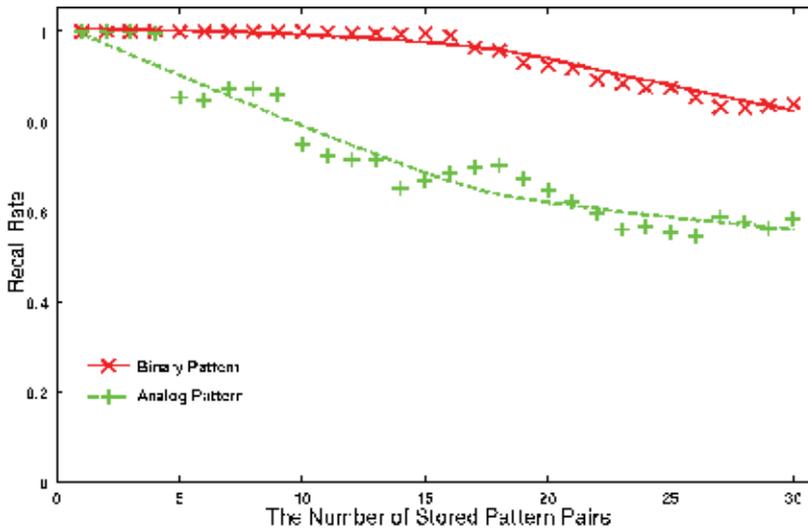


Fig. 7. Recall Ability in One-to-Many Associations.

4.6 Robustness for Damaged Neurons

Here, we examined the robustness for damaged neurons of the proposed model.

Figure 9 shows the robustness for damaged neuron of the proposed model. In this experiment, we used the proposed model which has 800(= 400 × 2) neurons in the I/O Layer and 400 neurons in the Map-Layer and 9 random patterns (three sets of patterns in one-to-three relations) were stored. In this experiment, *n*% of the neurons in the Map-Layer were damaged randomly. Figure 9 shows the average of 100 trials. In this figure, the results of the conventional KFMAM-AR were also shown.

From this result, we confirmed that the proposed model has the robustness for damaged neurons.

5. Conclusion

In this research, we have proposed the KFM Associative Memory with Refractoriness based on Area Representation. The proposed model is based on the KFAMM-AR (Abe & Osana, 2006) and the neurons in the Map-Layer have refractoriness. We carried out a series of computer experiments and confirmed that the proposed model has following features.

- (1) It can realize one-to-many associations of binary patterns.
- (2) It can realize one-to-many associations of analog patterns.
- (3) It has robustness for noisy input.
- (4) It has robustness for damaged neurons.

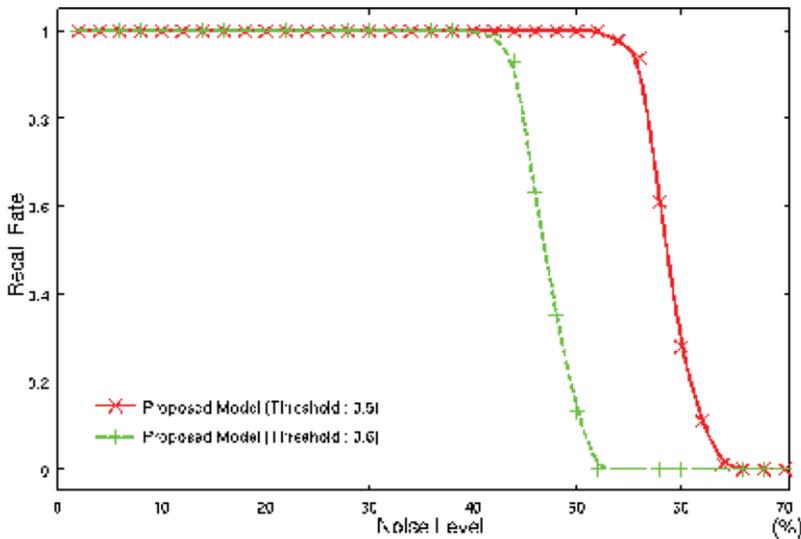


Fig. 8. Sensitivity to Noise (Analog Pattern).

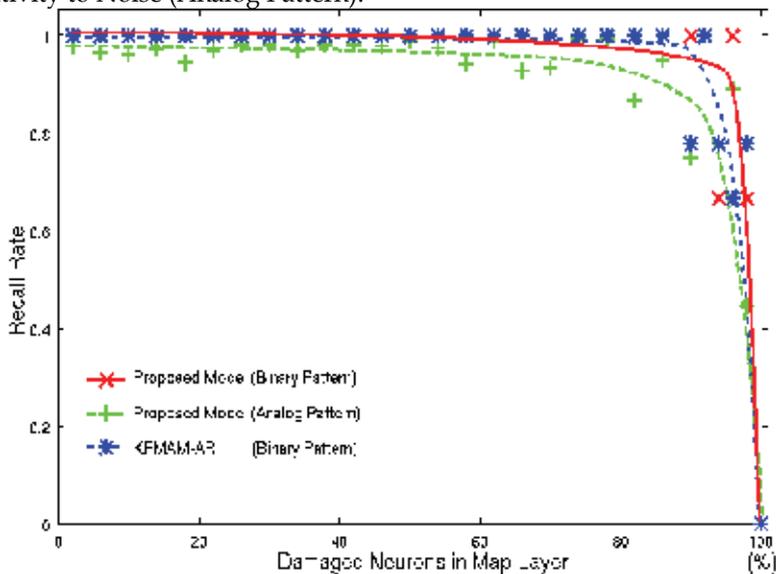


Fig. 9. Robustness for Damaged Neurons.

6. References

- Abe, H. & Osana, Y. (2006). Kohonen feature map associative memory with area representation. *Proceedings of IASTED Artificial Intelligence and Applications*, Innsbruck.
- Carpenter, G. A. & Grossberg, S. (1995). Pattern Recognition by Self-organizing Neural Networks. *The MIT Press*.
- Hattori, M. Arisumi, H. & Ito, H. (2002). SOM Associative Memory for Temporal Sequences. *Proceedings of IEEE and INNS International Joint Conference on Neural Networks*, pp. 950-955, Honolulu.
- Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of National Academy Sciences USA*, Vol.79, pp.2554-2558.
- Ichiki, H. Hagiwara, M. & Nakagawa, M. (1993). Kohonen feature maps as a supervised learning machine. *Proceedings of IEEE International Conference on Neural Networks*, pp.1944-1948.
- Ikeda, N. & Hagiwara, M. (1997). A proposal of novel knowledge representation (Area representation) and the implementation by neural network. *International Conference on Computational Intelligence and Neuroscience*, III, pp. 430-433.
- Kawasaki, N. Osana, Y. & Hagiwara, M. (2000) Chaotic associative memory for successive learning using internal patterns. *IEEE International Conference on Systems, Man and Cybernetics*.
- Kohonen, T. (1994). *Self-Organizing Maps*, Springer.
- Kosko, B. (1988). Bidirectional associative memories. *IEEE Transactions on Neural Networks*, Vol.18, No.1, pp.49-60.
- Osana, Y. & Hagiwara, M. (1999). Successive learning in chaotic neural network. *International Journal of Neural Systems*, Vol.9, No.4, pp.285-299.
- Rumelhart, D. E. McClelland, J. L. & the PDP Research Group. (1986). Parallel Distributed Processing. *Exploitations in the Microstructure of Cognition, Foundations*, The MIT Press, Vol.11.
- Watanabe, M. Aihara, K. & Kondo, S. (1995). Automatic learning in chaotic neural networks. *IEICE-A*, Vol.J78-A, No.6, pp.686-691.
- Yamada, T. Hattori, M. Morisawa, M. & Ito, H. (1999). Sequential learning for associative memory using Kohonen feature map. *Proceedings of IEEE and INNS International Joint Conference on Neural Networks*, paper no.555, Washington D.C.

Incremental Motion Planning With Las Vegas Algorithms

Jouandeau Nicolas, Touati Youcef and Ali Cherif Arab
*University Paris8
France*

1. Introduction

Las Vegas algorithm is a powerful paradigm for a class of decision problems that has at least a theoretical exponential resolving time. Motion planning problems are one of those and are out to be solved only by high computational systems due to such a complexity (Schwartz & Sharir, 1983). As Las Vegas algorithms have a randomized way to meet problem solutions (Latombe 1991), the complexity is reduced to polynomial runtime. In this chapter, we present a new single shot random algorithm for motion planning problems. This algorithm named RSRT for Rapidly-exploring Sorted Random Tree is based on inherent relation analysis between Rapidly-exploring Random Tree components, named RRT components (LaValle, 2004). RRT is an improvement of previous probabilistic motion planning algorithms to address problems that involve wide configuration spaces. As the main goal of the discipline is to develop practical and efficient solvers that automatically produce motion, RRT methods successfully reduce the complexity in exploring the space partially and producing non-deterministic solutions close to optimal ones. In the classical RRT algorithm, space is explored by repeating successively three phases: generation of a random configuration in the whole space (including free and non-free space); selection of a nearest configuration; and generation of a new configuration obtained by numerical integration over a fixed time step. Then the motion planning process is discretized into steps from the initial configuration to other configurations in the space. In such a way, RRT algorithms are the motion planners last generation that generally addresses a large set of motion planning problems. Mobile, geometrical or functional constraints, input methods and collision detection are unspecified. As it is possible to measure solutions provided by RRT, RSRT or other improvements in spaces with arbitrary dimension, experiments are realized on a wide set of path planning problems involving various mobiles in static and dynamic environments. We experiment the RSRT and other RRT algorithms using various configurations spaces to produce a massive experiment analysis: from free flying to constraint mobiles, from single to articulated mobiles, from wide to narrow spaces, from simple to complex distance metric evaluations, from special to randomly generated spaces. These experiments show practical performances of each improvement, and results reflect their classical behavior on each type of motion planning problems.

2. RRT Sampling Based-planning

2.1 Principle

In its original formulation (LaValle, 1998), RRT method is described as a tree $G = (V, E)$, where V is the set of vertices and E the set of edges in the research space. From an initial configuration q_{init} , the objective is to generate a sequence of commands, leading a mobile M , to explore all the configurations space C . The RRT method can solve this problem by searching solution which spans a tree, where the configuration q_{init} , describes the root node. One can note that nodes and arcs represent respectively eligible configurations of M and commands which are applied to move between the configurations. RRT method is a random incremental search of configurations which permits a uniform exploration of the space. The RRT implementation consists on a three phases: generate a configuration q_{rand} , select a configuration q_{prox} inside the current tree, and integrate a new configuration q_{new} from q_{prox} towards q_{rand} .

During the first phase, a random function is implemented to select an element of a configurations space. The second phase consists of choosing q_{prox} of G , which is the nearest element of q_{rand} . This phase is based on a metric ρ . Finally, a new configuration q_{new} from q_{prox} towards q_{rand} is generated and the objective is to implement a control which leads to bring q_{prox} closer to q_{rand} . The new configuration q_{new} is generated by integrating from q_{prox} , during a predefined time interval.

2.2 Graph construction of RRT method

Firstly, the RRT method is developed to solve planning problem in mobile robotic. In the original algorithm, the possible constraints associated to M are not mentioned. During the formulation of G , changes to be made for adding new constraints are minors, and the precision depends mainly on the chosen local planning method. The graph elementary construction in RRT method is described according to algorithm ALG. 1.

```

consRrt (qinit, k, Δt, C)
  init (qinit, G) (1)
  for i in 1 to k (2)
    qrand = randConfig ( C )
    qprox = nearestConfig (qrand, G) (3)
    qnew = newConfig (qprox, qrand, Δt) (4)
    addConfig (qnew, G)
    addEdge (qprox, qnew, G)
  return G

nearestConfig (qrand, G)
  d = inf
  foreach q in G
    if ρ ( q, qrand ) < d (5)
      qprox = q
      d = ρ ( q, qrand )
  return qprox

```

ALG. 1. Original RRT algorithm formulation

We remark that the algorithm implements three functions. The first one, `randConfig`, ensures a uniform partition of random samples in C , and guaranties uniform exploration (Yershova & LaValle, 2004 and Lindemann et al., 2004). Function `nearestConfig` selects the nearest configuration q_{rand} of G . This relation proximity is defined by a distance metric ρ , as it is illustrated in (ALG. 1. (5)). In the case of probabilistic methods PRM and RRT, the nearest neighbour search with arbitrary dimension can be optimised (Yershova & LaValle, 2007 and Yershova & LaValle, 2002). Reducing of the search time of a nearest neighbour permits to use a complex distance metric. A new configuration q_{new} can be defined by `newConfig` from q_{prox} towards q_{rand} . Knowing that M is subject to holonomic constraints, a control inputs can be applied to move from q_{prox} towards q_{rand} with displacements amplitudes Δt . Functions `addConfig` and `addEdge` add respectively q_{new} to the list of nodes of G and arcs between q_{prox} and q_{new} .

2.3 Cardinality and layer

For each new configuration q_{new} in the generation phase, RRT method adds a configuration by propagating q_{prox} of G . In this case, no restriction on q_{new} is imposed according to configurations set G . So, q_{new} can be similar to q_{exist} , which can make possible to span a graph with or without cycle. For example, let's define $Card$ as a cardinal of set, thus, if $Card(V) = Card(E) + 1$, then we can conclude that the graph is non-cyclic. To avoid stacking of identical movements, each nodes q_{prox} can't be extended towards q_{rand} for creating q_{new} , if it doesn't already have a similar descendent.

If q_{prox} is extended towards q_{rand} , a new arc between q_{prox} and q_{new} is inserted in E .

If $Card(V) \leq Card(E)$, we can conclude that the graph contains at least one cycle. Thus, is q_{new} deleted and a new arc is inserted in E between q_{prox} and q_{exist} .

Creating cycles leads to decrease an expansion number of G in unexplored zones. However, it permits to list possible solutions in the case of halt. Knowing that this scenario is more topologic than geometric, RRT method is better without cycle [LAV98]. Fig 1 shows the expansion of G respectively after 100, 500 and 1500 samples. Random samples have been uniformly spread in the square. q_{init} is initially in the center of the square. The mobile is a simple point (without geometric shape) with holonomic constraints.

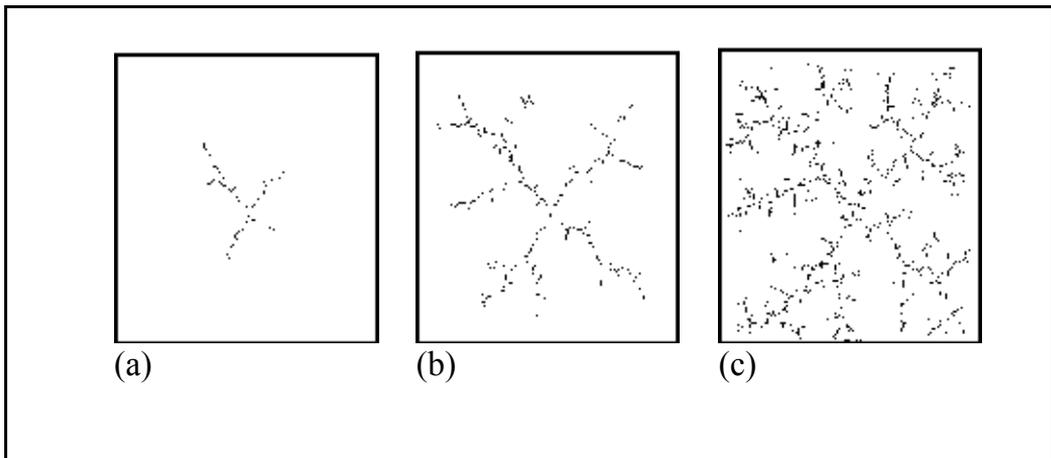


Fig. 1. Expansion of G in a free square (a after 100, b after 500 and c after 1500 samples)

2.4 Natural expansion

The random distributions of samples which performs expansions, directs naturally the growth of G towards the wider regions of space. This can be verified by constructing Voronoi diagram which associates, for each new node of C , one Voronoi cell. For each iteration of RRT method, the localization probability of the next random sample is more important towards the largest cells of Voronoi diagram, which is defined by a previous random samples set.

Let's C_k be a distribution of k random samples in the configurations space C . the distribution C_k converges in term of probability to C under condition of the uniformity of a random samples partition in C (LaValle & Kuffner, 2000).

Knowing that Delaunay triangulation is a dual of Voronoi diagram, an example of a graph expansion associated to RRT method is presented in Fig. 2. Graphs presented in (a), (b) and (c) illustrate respectively the results of 25, 275 and 775 expansions including those of Delaunay triangulations illustrated in (a'), (b') and (c'). The space is two dimensional squares without obstacles. For each iteration, adding a new item leads to construct a new triangulation. In Fig. 2, initial configuration is represented by a circle in the center of the space.

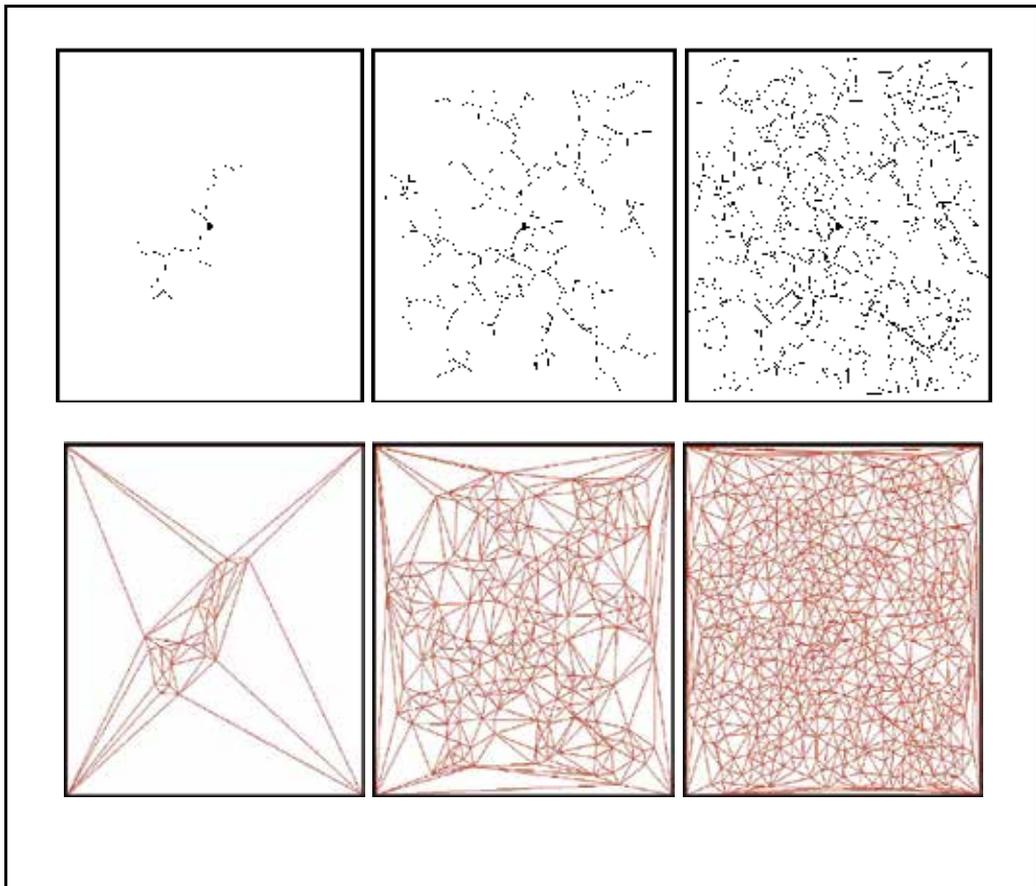


Fig. 2. Triangulation analysis due to samples

The evolution of new configurations of G along iterations is illustrated in Fig 3. X-axis represents the configuration number contained in the graph and Y-axis represents the percentage of the entire surface S . The surface graph represents the average, minimal, and maximal surface variations. In this case, the average surface is the average triangles surfaces. The standard deviation graph represents the average, minimal and maximal standard deviations. The initial configuration divides the space into four triangles with 0.25 in term of surface and zero in standard deviation. The average area of triangles decreases linearly according to the number of configurations.

In Figures 2, 3 and 4, positions in (a), (b) and (c) are placed around area average and standard deviations curves. Maximum and minimum variations can increase or decrease according to their relative positioning to the decreasing average value. Due to the logarithmic scale, position of minimal variations vis-à-vis average values shows the almost-equality between average value and minimum value. On the other hand, position of maximal variations shows triangles much larger than the average value before a density threshold (8.15 times larger before 353 configurations). From 353 configurations, the ratio between the higher triangle and the average value progresses in stair-steps. Two stair-steps p_0 and p_1 are placed on average and standard deviations curves as it's illustrated in Fig 3. This ratio tends to be stabilized around 2 from p_1 . The initial configuration position has no influence on statistics relative to its expansion.

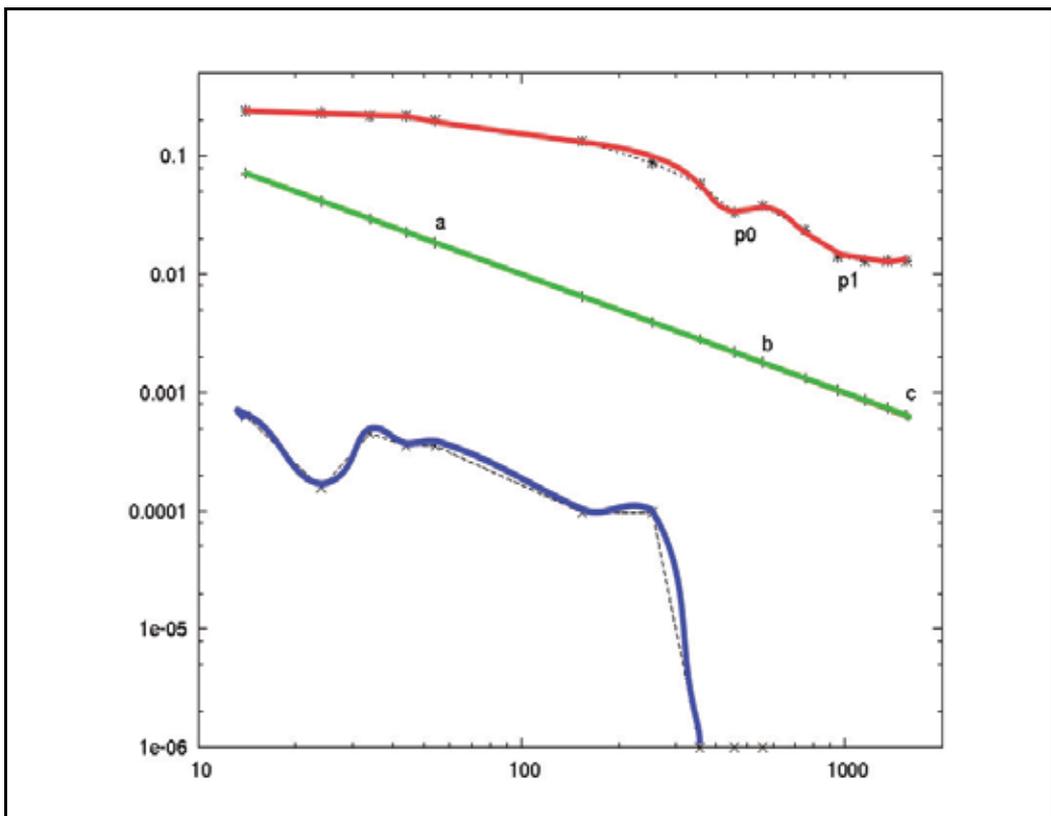


Fig. 3. Evolution of average, min and max of triangles areas during sampling

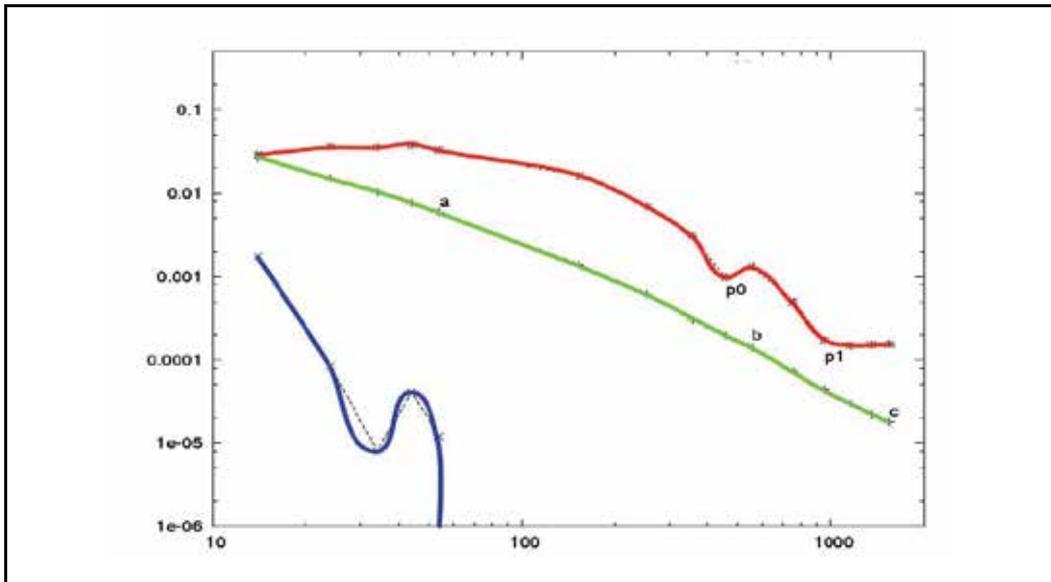


Fig. 4. Evolution of standard deviation, min and max of deviation areas during sampling

2.5 End condition

A query of a mobile trajectory planning can be formulated according to a pair of configuration-objective, which is instantiated on q_{obj} or on a set of configurations C_{obj} . Restricting the search to a single configuration-objective can penalize the mobiles which are subjected to dynamics or non-holonomics constraints. To improve the convergence towards the objective, RRT resolutions implement a configuration q_{obj} , whose components are not fixed. Thus, the planning problem consists to find a path connecting q_{init} to an element of C_{obj} . From q_{init} , graph G seeks to achieve a configuration q_{obj} . This can be done by a successive adding of new configuration q_{new} in the tree G . Variable k defines the number of iterations required to solve the problem. In the case of k is not sufficient it is possible to continue conducting research on new k iterations from the previously generated tree. The construction of G is achieved when $q_{obj} \cap G = \emptyset$.

3. Related Works

In the previous section, C is presented without obstacle in an arbitrary space dimension. At each iteration, a local planner is used to connect each couples (q_{new}, q_{obj}) in C . The distance between two configurations in T is defined by the time-step Δt . The local planner is composed by temporal and geometrical integration constraints. The resulting solution accuracy is mainly due to the chosen local planner. k defines the maximum depth of the search. If no solution is found after k iterations, the search can be restarted with the previous T without re-executing the init function. This principle can be enhanced with a bidirectional search, shortened Bi-RRT (LaValle & Kuffner, 1999). Its principle is based on the simultaneous construction of two trees (called T_{init} and T_{obj} that grows respectively from q_{init}

and q_{obj} . The two trees are developed towards each other while no connection is established between them. This bidirectional search is justified because the meeting configuration of the two trees is nearly the half-course of the initial configuration space. Therefore, the resulting resolution time complexity is reduced (Russell & Norvig, 2003).

RRT-Connect is a variation of Bi-RRT that consequently increase the Bi-RRT convergence towards a solution (Kuffner & LaValle, 2000) thanks to the enhancement of the two trees convergence. This has been settled :

- to ensure a fast resolution for “simple” problems (in a space without obstacle, the RRT growth should be faster (ALG.2. (1)) than in a space with many obstacles)
- to maintain the probabilistic convergence property. Using heuristics modify the probability convergence towards the goal and also should modify its evolving distribution. Modifying the random sampling can create local minima that could slow down the algorithm convergence

```

connectRrt (q, Δt, T)
  r = ADVANCED
  while r equals ADVANCED
    r = expandT (q, Δt, T)
  return r

```

(1)

ALG. 2. Connecting a configuration q to T with RRT-Connect.

As it makes RRT less incremental, RRT-Connect is more adapted for non-differential constraints (Cheng, 2001). It iteratively realize expansion by replacing a single iteration (ALG. 1. (2)) with connectT function which corresponds to a succession of successful single iterations (ALG. 2. (1)). An expansion towards a configuration q becomes either an extension or a connection.

```

connectBiRrt (qinit, qobj, k, Δt, C)
  init (qinit, Ta)
  init (qobj, Tb)
  for i in 1 to k
    qrand = randConfig (C)
    r = expandRrt (qrand, Δt, Ta)
    if r not equals TRAPPED
      if r equals REACHED
        qco = qrand
      else
        qco = qnew
        if connectRrt (qco, Ta, Tb)
          Return solution
    swap (Ta, Tb)
  return TRAPPED

```

ALG. 3. Expanding two graphs with RRTConnect

According that two trees are constructed by Bi-RRT, growth is realized inside two trees named T_a and T_b and a successful connection of q_{new} towards q_{rand} in T_a , implies many other extensions (as many as the free space admits new free configurations, i.e. q_{new} in C_{free}) of q_{prox} found in T_b towards q_{new} . This new configuration q_{new} becomes a convergence configuration named q_{co} (ALG. 3).

To improve the construction of T to an adequate progression of G in C_{free} , previous works propose :

- to deviate from its initial distribution the random sampling Bi-RRT and RRT-Connect. Other Variations of RRT-Connect are called RRT-ExtCon, RRT-ConCon and RRT-ExtExt; they modify the construction strategy of one of the two trees. The priorities of extension and connection are balanced with new values according to previous extensions (LaValle, 1998)
- to adapt q_{prox} selection to a collision probability (Cheng & LaValle, 2001)
- to restrict q_{prox} selection in an accessibility vicinity of the previous q_{prox} in the variation called RC-RRT (Cheng & LaValle, 2002)
- to bias sampling towards free spaces (Lindemann & LaValle, 2004)
- to parallelize growing operations for n distinct graphs in the variation OR parallel Bi-RRT and to share G with a parallel q_{new} sampling in the variation embarrassingly parallel Bi-RRT (Carpin & Pagello, 2002)
- to focus the sampling of special parts of C to control the RRT growth (Cortès & Siméon, 2004 and Lindemann & LaValle, 2003 and Yerushova et al. 2005)

By adding the collision detection in the configuration space, the selection of nearest neighbor q_{prox} is garanted by a collision detector. The collision detection is expensive in computing time, the distance metric evaluation ρ is subordinate to the collision detector.

```

expandRrt(q , Δt , T )
  qprox = closestConfig ( q , T )
  dmin = rho (qprox , q )
  success = FALSE
  foreach u in U
    qtmp = integrate ( q , u , Δt )
    if isCollisionFree (qtmp , qprox , M , C)
      d = ro (qtmp , qrand )
      if d < dmin
        qnew = qtmp
        success = TRUE
  if success equals TRUE
    insert (qprox , qnew , T )
    if qnew equals q
      return REACHED
    return ADVANCED
  return TRAPPED

```

ALG. 4 Expanding according to a collision detector

As U defines the set of admissible orders available to the mobile M , the size of U mainly defines the computation times needed to generate, validate and select the closest configuration with as the best expansion configuration. For each expansion, the function `expandRrt` (ALG. 3.) returns three possible values: REACHED if the configuration q_{new} is connected to T , ADVANCED if q is only an extension of q_{new} which is not connected to T , and TRAPPED if q cannot accept any successor configuration q_{new} .

The construction of T corresponds to the repetition of such a sequence. The collision detection discriminates the two possible results of each sequence :

- the insertion of q_{new} in T (i.e. without obstacle along the path between q_{prox} and q_{new})
- the rejection of each q_{prox} successors (i.e. due to the presence of at least one obstacle along each successors path rooted at q_{prox})

The rejection of q_{new} induces an expansion probability related to its vicinity (and then also to q_{prox} vicinity); the more the configuration q_{prox} is close to obstacles, the more its expansion probability is weak. It reminds one of fundamentals RRT paradigm: free spaces are made of configurations that admit various number of available successors; good configurations admit many successors and bad configurations admit only few ones. Therefore, the more good configurations are inserted in T , the better the RRT expansion will be. The problem is that we do not previously know which good and bad configurations are needed during the RRT construction, because the solution of the considered problem is not yet known. This problem is also underlined by the parallel variation (Carpin & Pagello, 2002) called OR Bi-RRT (i.e. to define the depth of a search in a specific vicinity). For a path planning problem p with a solution s available after n integrations starting from q_{init} , the question is to maximize the probability of finding a solution; According to the concept of "rational action", the response of P3 class to adapt a on-line search can be solved by the definition of a formula that defines the cost of the search in terms of "local effects" and "propagations" (Russell, 2002). These problems find a way in the tuning of the behaviour algorithm like CVP did (Cheng, 2001).

3.2 Tuning the RRT algorithm according to relations between components

In the case of a space made of a single narrow passage, the use of bad configurations (which successors generally collide) is necessary to resolve such problem. The weak probability of such configurations extension is one of the weakness of the RRT method (Jaillet L. et al. 2005).

To bypass this weakness, we propose to reduce research from the closest element (ALG. 4) to the first element of C_{free} . This is realized by reversing the relation between collision detection and distance metric; the solution of each iteration is validated by subordinating collision tests to the distance metric; the first success call to the collision detector validates a solution. This inversion induces :

- a reduction of the number of calls to the collision detector proportionally to the nature and the dimension of U . Its goal is to connect the collision detector and the derivative function that produce each q_{prox} successor
- an equiprobability expansion of each node independently of their relationship with obstacles

The T construction (we called RSRT) is now based on the following sequence:

- the generation of a random configuration q_{rand} in C
- the selection of q_{prox} the nearest configuration to q_{rand} in T
- the generation of each successors of q_{prox} . Each successor is associated with its distance metric from q_{rand} . It produces a couple called s stored in S
- the sort of s elements by distance
- the selection of the first collision-free element of S and breaking the loop as soon as this first element is discovered

4. Results

Fig. 6. and 7. present two types of environment that have been chosen to test algorithms. In these environments, obstacles are placed. For each type, we have generated series of environments that gradually contains more obstacles. This is one element of these series that we call a problem. For each problem, we generate 10 different instances, to realise statistics on solutions provide (Fig. 5). The number of obstacles is defined by the sequence 2, 4, 8 ... 512 and also until the resulting computing time is less than 60 sec. We have fixed this limit to see what could be possible in an embedded system. The two types of environment correspond to a simple mobile robot and a small arm with 6-DOF. We used the Proximity Query Package (PQP) library to test collisions and the Open Inventor library to visualize solutions. For each mobile in each environment, we have applied a uniform inputs set dispatched over translation and rotation.

Considering generic systems, we have apply different mover's model:

- that consider the trajectory as a list of position
- that consider the trajectory as a list of position with a velocity for each DOF

Each set of instances are associated with different distances metrics (Euclidian, scaled Euclidian and Manhattan distances).

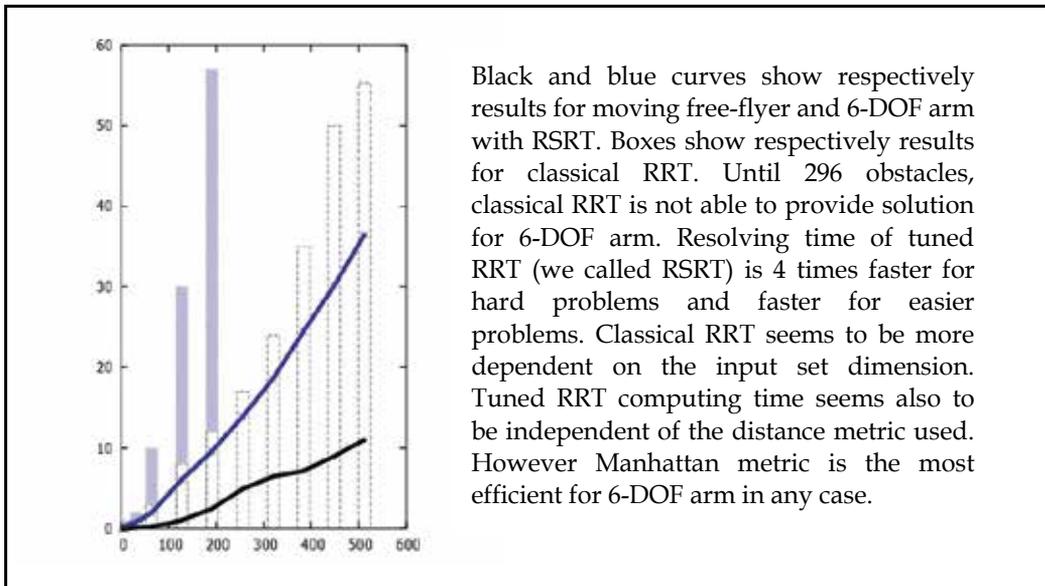


Fig. 5. Computing resolving times while gradually increasing environment complexity

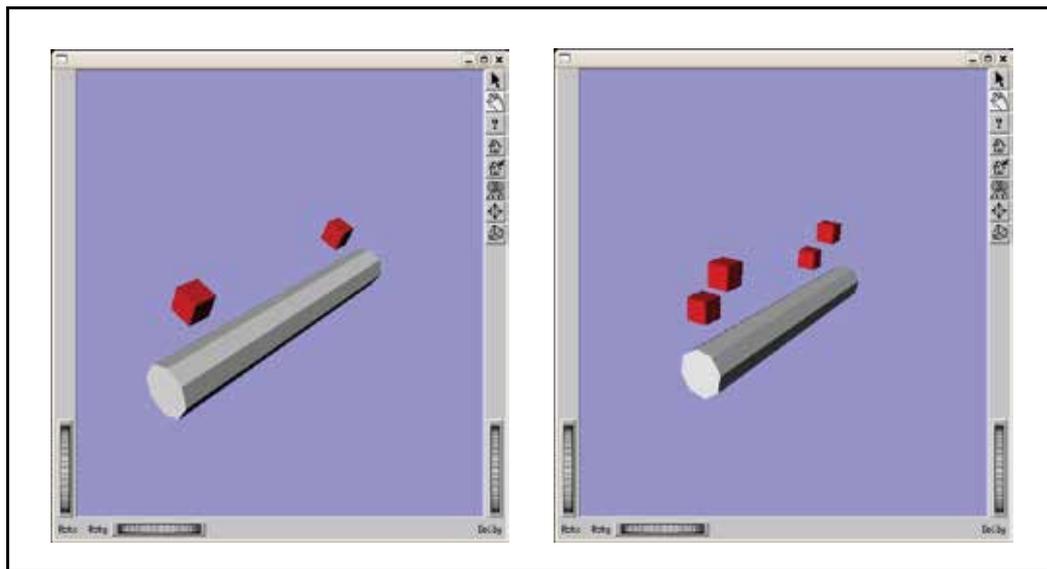


Fig. 6. Moving simple mobile and increasing gradually environment complexity

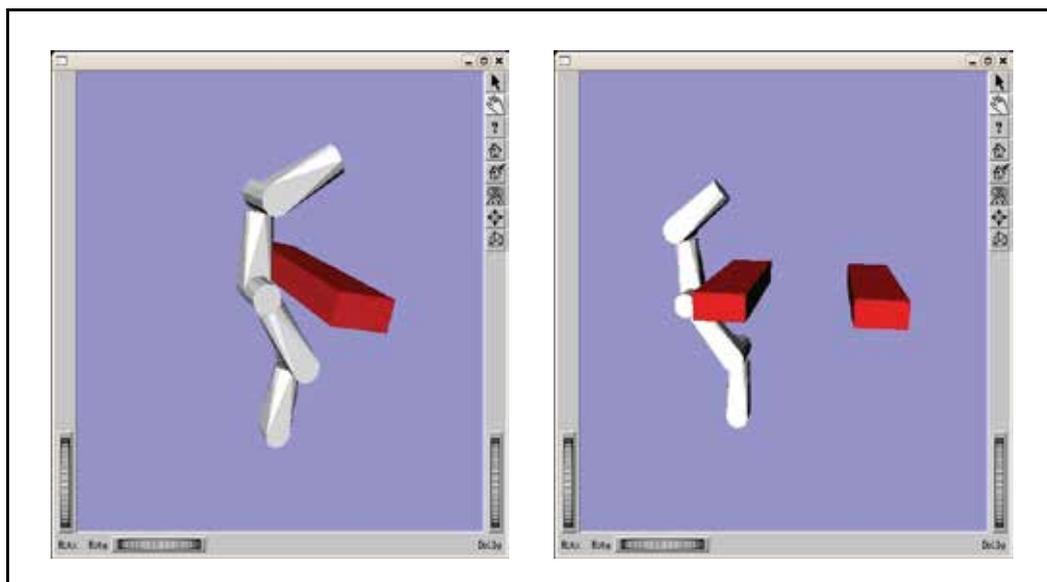


Fig. 7. Moving articulated mobile and increasing gradually environment complexity

7. Conclusion

We have described a way of tuning RRT algorithm, to solve more efficiently hard problems. RSRT algorithm accelerates consequently the required computing time. The result have been tested on a wide set of problems that have an appropriate size to be embedded. This approach allows RRT to deal with motion planning strategies based on statistical analysis.

8. References

- Carpin, S. & Pagello, E. (2002). On Parallel RRTs for Multi-robot Systems, *8th Conf. of the Italian Association for Artificial Intelligence (AI*IA)*
- Cheng, P. & LaValle, S. (2002). Resolution Complete Rapidly-Exploring Random Trees, *Int. Conf. on Robotics and Automation (ICRA)*
- Cheng, P. (2001) Reducing rrt metric sensitivity for motion planning with differential constraints, *Master's thesis, Iowa State University*
- Cheng, P. & LaValle, S. (2001). Reducing Metric Sensitivity in Randomized Trajectory Design, *Int. Conf. on Intelligent Robots and Systems (IROS)*
- Cortès, J. & Siméon, T. (2004). Sampling-based motion planning under kinematic loop-closure constraints, *Workshop on the Algorithmic Foundations of Robotics (WAFR)*
- Jaillet L. et al. (2005). Adaptive Tuning of the Sampling Domain for Dynamic-Domain RRTs, *IEEE International Conference on Intelligent Robots and Systems (IROS)*
- Kuffner, J. & LaValle, S. (2000). RRT-Connect: An efficient approach to single-query path planning, *Int. Conf. on Robotics and Automation (ICRA)*
- Latombe, J. (1991). Robot Motion Planning (4th edition), *Kluwer Academic*
- LaValle, S. (2004). Planning Algorithms, [on-line book] <http://misl.cs.uiuc.edu/planning/>
- LaValle, S. & Kuffner, J. (2000). Rapidly-exploring random trees: Progress and prospects, *Workshop on the Algorithmic Foundations of Robotics (WAFR)*
- LaValle, S. & Kuffner, J. (1999). Randomized kinodynamic planning, *Int. Conf. on Robotics and Automation (ICRA)*
- LaValle, S. (1998). Rapidly-exploring random trees: A new tool for path planning, *Technical Report 98-11, Dept. of Computer Science, Iowa State University*
- Lindemann, S. & LaValle, S. (2004). Incrementally reducing dispersion by increasing Voronoi bias in RRTs, *Int. Conf. on Robotics and Automation (ICRA)*
- Lindemann, S. et al. (2004). Incremental Grid Sampling Strategies in Robotics, *Int. Workshop on the Algorithmic Foundations of Robotics (WAFR)*
- Lindemann, S.R. & LaValle, S.M. (2003). Current issues in sampling-based motion planning, *Int. Symp. on Robotics Research (ISRR)*
- Lozano-Pérez, T. (1983). Spatial Planning: A Configuration Space Approach, *Trans. on Computers*
- Russell, S. & Norvig, P. (2003). Artificial Intelligence, A Modern Approach (2nd edition), *Prentice Hall*
- Russell, S. (2002). Rationality and Intelligence, *Press O.U.*, ed.: Common sense, reasoning, and rationality
- Schwartz, J. & Sharir, M. (1983). On the piano movers problem: I, II, III, IV, V, *Technical report, New York University, Courant Institute, Department of Computer Sciences*
- Yershova, A. & LaValle, S. (2007). Improving Motion Planning Algorithms by Efficient Nearest Neighbor Searching, *IEEE Transactions on Robotics* 23(1):151-157
- Yershova, A. et al. (2005). Dynamic-domain rrts: Efficient exploration by controlling the sampling domain, *Int. Conf. on Robotics and Automation (ICRA)*
- Yershova, A. & LaValle, S. (2004). Deterministic sampling methods for spheres and SO(3), *Int. Conf. on Robotics and Automation (ICRA)*
- Yershova, A. & LaValle, S. (2002). Efficient Nearest Neighbor Searching for Motion Planning, *Int. Conf. on Robotics and Automation (ICRA)*

Hierarchical Fuzzy Rule-Base System for MultiAgent Route Choice

Habib M. Kammoun, Ilhem Kallel & Adel M. Alimi
*Research Group on Intelligent Machines REGIM, University of Sfax
Tunisia*

1. Introduction

In view of both complexity and dynamicity of road networks and the sharp increase of vehicle number, accidents and traffic jam situations in all road networks have become wide spread all over the world. A solution for these problems is to develop and invest in traffic management using intelligent techniques from artificial intelligence and soft computing. An accurate management will improve traffic efficiency over time and space with dynamic interventions. This latter means the need of an auto detection of jam situations or incidents, so vehicles will be adapted according to the new road network situation. In this way, it appears the necessity of an intelligent route choice system helping drivers to attempt their destinations.

The route choice concerns the selection of better itinerary from a set of feasible itineraries between an origin and a destination in road network. The route choice process improves the fluency of road network, reduces the number of traffic congestion and allows a dynamic assignment of traffic flows (Bierlaire et al., 2008). It is clear that route choice models play a crucial role in many transport applications (for example, it is the core of traffic assignment models). Furthermore, a better understanding of route choice decision-making behaviour will make possible to explain the modification of traffic flow.

A large number of research efforts are dedicated to studying the route choice problem. In this way, one of the most realistic technique used until now for route choice is the fuzzy logic (Teodorovic & Kikuchi, 1990). This model is capable of incorporating subjectivity, ambiguity, and uncertainty from perceptions for an accurate traffic management. The results of route choice based on fuzzy model are better than those using discrete choice models (Ben-Akiva & Lerman, 1985) (Bekhor et al., 2002).

Nowadays, after many developments in information acquisition technologies, route choice becomes even more complicated when more traffic information is available in real-time to drivers. In addition to all the usual factors that affect travel decisions (such as travel time, travel distance), additional factors (such as type of road, traffic flow speed, weather conditions, and personal preferences) affect also the final choice. So, the itinerary selection process made by drivers, taking into account many factors, is most often very complicated. It is extremely hard to formulate a suitable mathematical model due to the subjectivity, uncertainty, and dynamicity of traffic flows and other factors. Thus, the development of

Fuzzy Rule-Based System (FRBS) seems justified in this situation through its capability to approximate a real continuous function with a good accuracy.

However, the application of FRBS is difficult according to the rule-explosion problem due to the large number of criteria. In order to deal with this problem, we propose in this chapter a route choice model based on hierarchical FRBS. This system is encapsulated into an intelligent vehicle agent defined into a hierarchical multiagent architecture of an advanced road network developed previously in order to deal with its complexity (Kammoun et al., 2008).

The paper is organized as follows: Next section presents an overview on the use of fuzzy logic in route choice problem. The third section describes the road network architecture, the hierarchical FRBS, and the itinerary selection model. The multiagent simulation part is detailed in the fourth section. The fifth section presents experiments and discusses results. Finally, we conclude by summarizing our contribution and presenting some directions for future work.

2. Literature review

In this review, we focus our attention especially on route choice models which are based on fuzzy logic. In fact, the fuzzy logic appeared in 1965 by Zadeh introducing the concept of a fuzzy set (Zadeh, 1965). It is shown as a very promising mathematical approach characterized by subjectivity, ambiguity, uncertainty, and imprecision. The model based on this approach is robust to small variations and easy to design. Figure 1 shows the basic elements of a fuzzy logic system: fuzzifier, rules, inference, and defuzzifier.

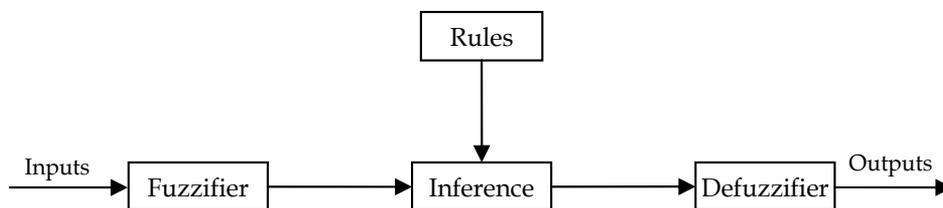


Fig. 1. Basic elements of fuzzy logic system

2.1 Previous works

The route choice problem has been dealt with several techniques using in the most cases the discrete choice models as logit and probit models (Ben-Akiva & Lerman, 1985) (Bekhor et al., 2002). However, these models can't consider subjectivity, ambiguity, and uncertainty from perceptions. Furthermore, they present an efficiency gap for addressing the complexity and the dynamicity of transportation systems.

In order to enhance these problems, research in soft computing field is still exploring the application of fuzzy set theory, using a set of "if-then" rules. This theory has been also used as a framework to solve other transportation problems as traffic assignment problem, accident analysis and prevention, traffic controller in roads intersection, and traffic light controller. For more details on transportation engineering based on fuzzy logic, see the state of art presented by Teodorovic (Teodorovic, 1999) which summarize important works in nineties.

Research in route choice problem based on fuzzy logic began by modelling a simple two-route choice problem in order to select the better route (itinerary) (Teodorovic & Kikuchi, 1990). The basis of this model is the use of fuzzy linguistic rules such as:

“IF perceived travel times on path A IS much longer than perceived travel time on path B, THEN fuzzy preference indices for A IS very strong”.

After this work, other works have been developed in order to improve the first one. In this way, Lotan and Koutsopoulos present a modelling framework for route choice under the presence of information, based on fuzzy logic and approximate reasoning (Lotan & Koutsopoulos, 1993). Later than, Lotan improves his framework with two stages: the first one presents the information integration and the second one presents the decision process (Lotan, 1998). In experimentation stage, both familiar and unfamiliar drivers have participated.

In order to deal with route choice behaviour, a fuzzy reasoning approach has been developed by Akiyama et al. (Akiyama et al., 1993). This approach has improved with a multi-stage fuzzy reasoning approach to solve the multi-route choice problem (Akiyama & Tsuboi, 1996). In fact, authors propose two approximate reasoning stages for driver decision making process. In the first one, travel time, degree of congestion, and risk of accidents are factors used to determine the utility of each feasible route. The second stage determines the frequency degree for each route, based on the difference between route utilities associated with the shortest path and the second shortest path, and the difference between route utilities associated with the second shortest path and the third shortest path. We note that only the case of three feasible routes has been considered. The fuzzy route choice model, taking into account imprecision and uncertainty, has been proved by Henn as a generalization of the standard logit model (Henn, 2000).

In this century, other works have been developed to deal with more complex route choice taking into account other factors. Ridwan has the first work that considers the spatial knowledge of individual travellers (Ridwan, 2004). He proposes a model of route choice, based on fuzzy travellers' preference relations, of which elements are fuzzy pairwise comparisons between feasible routes. In order to improve the route choice behaviour, Hawas proposes calibration methodology and knowledge base composition, using a combined approach of fuzzy logic and neural nets (Hawas, 2004). Four stages are developed to compute the final route utility based on both numerical and categorical inputs: prior-to-choice stage, following-the-trip choice stage, reliability level stage, and route choice stage. Furthermore, Peeta and Yu propose a fuzzy model, using a hybrid probabilistic-possibilistic model, in order to quantify the latent attractiveness of alternative routes with regard to the qualitative variables (Peeta & Yu, 2004). Concerning the description of route choice behaviour, Arslan and Khysti propose a hybrid model using concepts from fuzzy logic and analytical hierarchy process AHP (Arslan & Khysti, 2005). The route selection in this work is provided by pairwise comparisons with respect to related criteria (travel time, congestion and safety). The fuzzy linguistic rules used in this model have the following structure:

“IF alternative A IS more desirable AND alternative B IS much more desirable, THEN preference of A over B IS weak importance”.

Nowadays, in order to deal with the presence of many components in road networks, research direction in this area has improved by the integration of multiagent systems from artificial intelligence. In fact, after thirteen years, Teodorovic concludes that a number of emergent traffic phenomena cannot be analyzed successfully and explained using analytical models (Teodorovic, 2003). Then, it is necessary to develop models based on multiagent approach and especially on swarm intelligence. The interacting agents might be drivers, passengers, cars, etc. In this sense, Teodorovic propose a fuzzy ant system for transportation modelling. Furthermore, under the influence of real-time traffic information, Panwai and Dia model drivers as intelligent agents composed of various mental elements: beliefs, capabilities, commitments, behavioural rules, and commitment rules. The agent's knowledge relevant to route choice decision is constructed using the fuzzy-neural approach based on socioeconomic data (Panwai & Dia, 2006).

2.2 Towards the use of hierarchical fuzzy logic

In recent years, many developments in information acquisition technologies through Advanced Traveller Information Systems (ATIS) have been done. Thus, many factors that affect route choice decision such as travel distance, travel speed, weather conditions, travel time, personal preferences, work information, and other traffic information, are available in real-time. It is clear that for an accurate selection, route choice model have to consider all available information.

Regarding the increasing number of selection criteria used to select the better alternative, the application of fuzzy logic to route choice problem with a large number of inputs involve the problem of rule-explosion. For example, if a system requires n input variables each partitioned into m membership functions, the total number of rules required to model the system by using one single fuzzy inference system is m^n . In order to deal with this problem, some hierarchical fuzzy systems have been proposed (Lee et al., 2003) (Rattasiri & Halgamuge, 2003). With a hierarchical architecture, the number of rules increases linearly related to the number of inputs $(n-1).m^2$ rather than exponentially. We note that the hierarchical FRBS has been successfully used in several applications areas showing a real improvement in precision and interpretability (Kallel et al., 2005).

Furthermore, few models (Panwai & Dia, 2006) have been proposed to solve the complexity of a fuzzy-route choice problem by distribution and parallelisation under the multiagent approach (Wooldridge, 2002).

Until now, fuzzy-route choice is applied by evaluating only two or three alternatives with a few number of selection criteria. This encourage us to go on further and to develop a hierarchical FRBS, in a cooperative multiagent system for traffic management; it can take into account a large number of selection criteria and compare a variable number of feasible routes.

3. Description of the multiagent-fuzzy system

The whole system includes two parts: a multiagent system for both distributed and cooperative management, and a hierarchical fuzzy system for itineraries evaluation and better itinerary selection.

3.1 Multiagent approach for traffic management

The traffic management in urban or interurban road networks is an important task in any country, and especially in industrialized countries, by its implications in economy and quality of life. It is clear that an accurate traffic management will allow more efficient vehicle routing in order to improve traffic efficiency. Due to the interaction of many components in road networks, it is necessary to have a representative architecture and real-time model capable to provide better routing while avoiding congested and jam roads.

In previous years, road network is based on traditional stationary equipments, such as loop detector, microwave detector and so on, that can cover only specific road section. Thus, collect traffic information is done only at limited road sections. Then, we cannot achieve an accurate management with little information.

In advanced road network model, all vehicles have to be able to connect with advanced equipments. Vehicles are equipped with the Global Positioning System GPS to locate the current position, a Route Guidance System RGS interfaced with driver by an onboard system, the Geographic Information System GIS providing a digital map, and wireless connection equipment to collect real-time information from the traffic control system. We note that some traffic information (number of vehicles in each road, speed, direction, etc.) can be collected, at regular intervals of time, with GPS (Zito et al., 1995). Recent developments on this equipment promote the research field in real-time road traffic information in order to improve route choice decision (Gong et al., 2007) (Bierlaire et al., 2008).

Using the natural geographic distribution of road networks, vehicles are regrouped by city and road. In order to deal with the complexity of road network and the large number of components, we propose hierarchical road network architecture based on multiagent approach (Kammoun et al., 2008). In our architecture, we assume that each road direction is supervised by an agent called Road Supervisor Agent RSA; the traffic control in each city is kept by an agent called City Agent CA; and each vehicle is considered as an agent called Intelligent Vehicle Agent IVA. Therefore, the use of both hierarchical and decentralized approaches is very interesting, and provides a flexible interaction between components and an adaptive behaviour according to the current traffic state. Figure 2 presents our hierarchical architecture for an advanced road network. The details of each agent are:

- City Agent CA: manages the connected city to obtain better road network exploitation. It maintains static information as roads characteristics and dynamic information as traffic information, road work information, weather information, etc., in each road in the city. By collaboration with RSA, CA receives, at each window time, information about each road. Furthermore, it cooperates with other CA managing other cities to collect real-time information, if a vehicle would reach another city;
- Road Supervisor Agent RSA: there are many RSA in one city. Each one supervises the state of traffic flow in the corresponding road, implements the control action of road, and achieves coordination control and integrated management by coordinating with the corresponding CA;
- Intelligent Vehicle Agent IVA: the vehicles are considered as reactive agents evolving in dynamic environment. In fact, using a multiagent reflexive reasoning, this agent is composed itself of three agents:

- Interface Agent: ensures the link between the driver's vehicle and the system. This agent communicates with GPS to receive its coordinates and informs the driver about the better path, the next road, and other information computed and collected by the decision making agent;
- Decision Making Agent: encapsulates the cooperative route choice algorithm. This agent selects the better alternative to reach vehicle's destination, while avoiding congested and jammed areas according to driver's preferences and history, under the influence of real-time traffic information collected by collaboration with CA and RSA. This algorithm is executed before each intersection (crossroad) to select the better next road for an adaptive routing;
- Effector Agent: Guide the driver to move the vehicle in order to reach destination (forward, turn left, turn right). It used in automatic driving or in simulation.

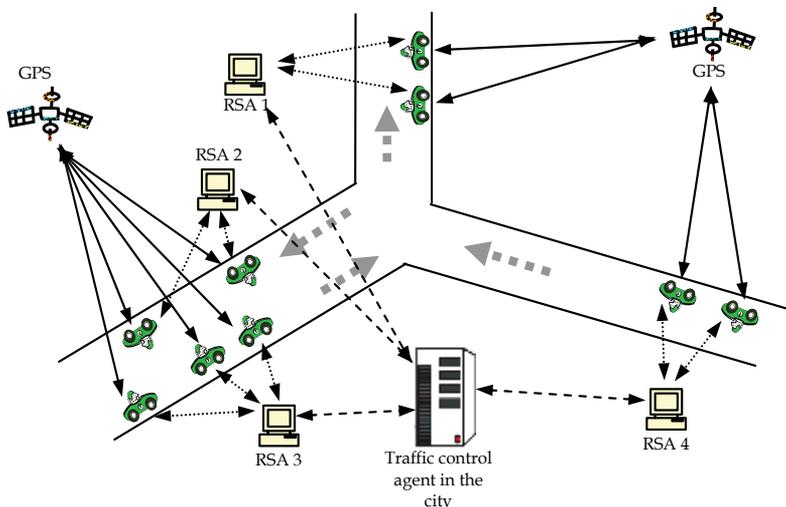


Fig. 2. Information flow in the modern road network

The sequence diagram shown in figure 3 illustrates the collaboration between agents to help driver. This sequence of actions is repeated in each intersection until reach destination. On the other hand, figure 4 illustrates the collaboration between RSA and CA in each window time to maintain real-time information.

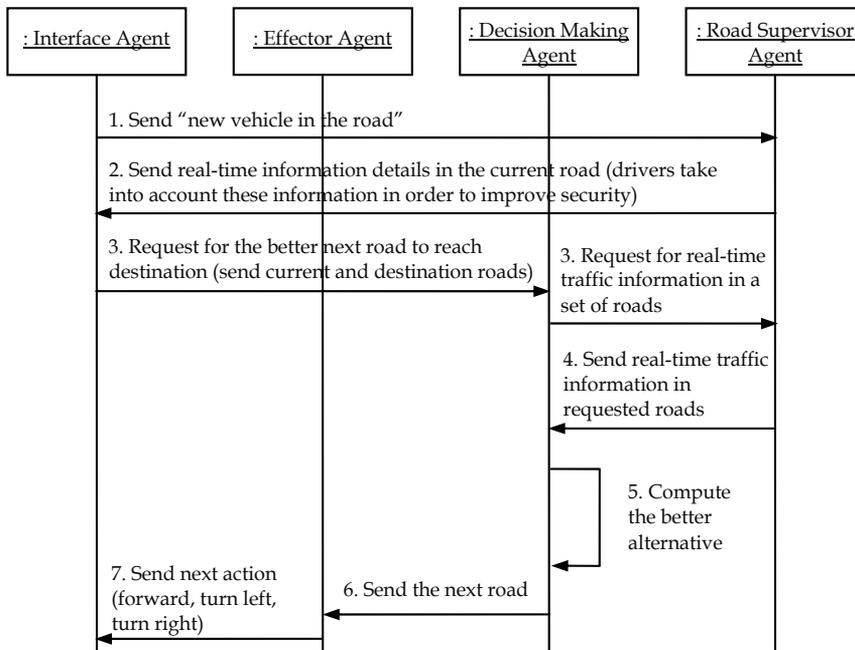
3.2 Hierarchical FRBS for an itinerary evaluation

After a first attempt to develop a hierarchical FRBS for route choice problem, we present in this chapter an improvement of our previous works (Kammoun et al., 2007). We note that this work is the first dealing with a hierarchical FRBS for route choice problem. The aim of this system is to evaluate one itinerary (path, a set of roads) according to selection criteria, to available real-time information in feasible itineraries, and to driver's preferences.

We have chosen eight inputs that have an important influence for road evaluation. However, the architecture can support other selection criteria. The following factors are the most important criteria, more used, and accessible from the vehicle information system:

- Vehicle number: corresponds to the average of vehicle number throughout roads composing the itinerary. This value takes into account the number of lanes and the road length;
- Congested vehicles number: corresponds to the average of vehicle number in congestion situation;
- Path work information: corresponds to the average of work information in a set of roads;
- Maximum path speed: corresponds to the average of speed allowed in a set of roads;
- Path familiarity: corresponds to the familiarity degree of driver in a set of roads;
- Driver speed: represents the average of speed usual used by the driver;
- Time of day: represents the time of travel;
- Weather conditions: represents the average of weather conditions in a set of roads.

We note that in a hierarchical architecture, outputs from certain Fuzzy Inference System (FIS) are used as inputs for the following FIS. In this case, it is difficult to design this kind of system because the intermediate outputs do not have physical meaning. To deal with this problem, we choose inputs combination that reduces limitations associated with the loss of physical meaning in intermediate outputs/inputs. So, inputs are regrouped by four categories according to traffic criteria, road criteria, driver criteria, and environment criteria. All FRBSs have two inputs and one output. Figure 5 illustrates the set of eight FRBSs used to evaluate one itinerary.



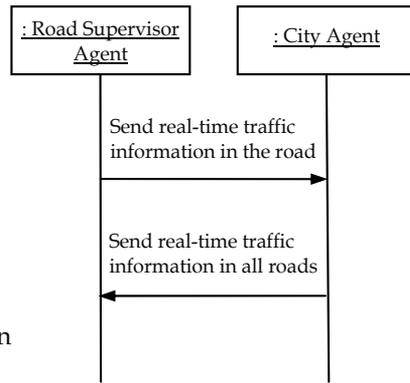


Fig. 3. Sequence diagram for next road selection

Fig. 4. Interaction between road supervisor and city agents

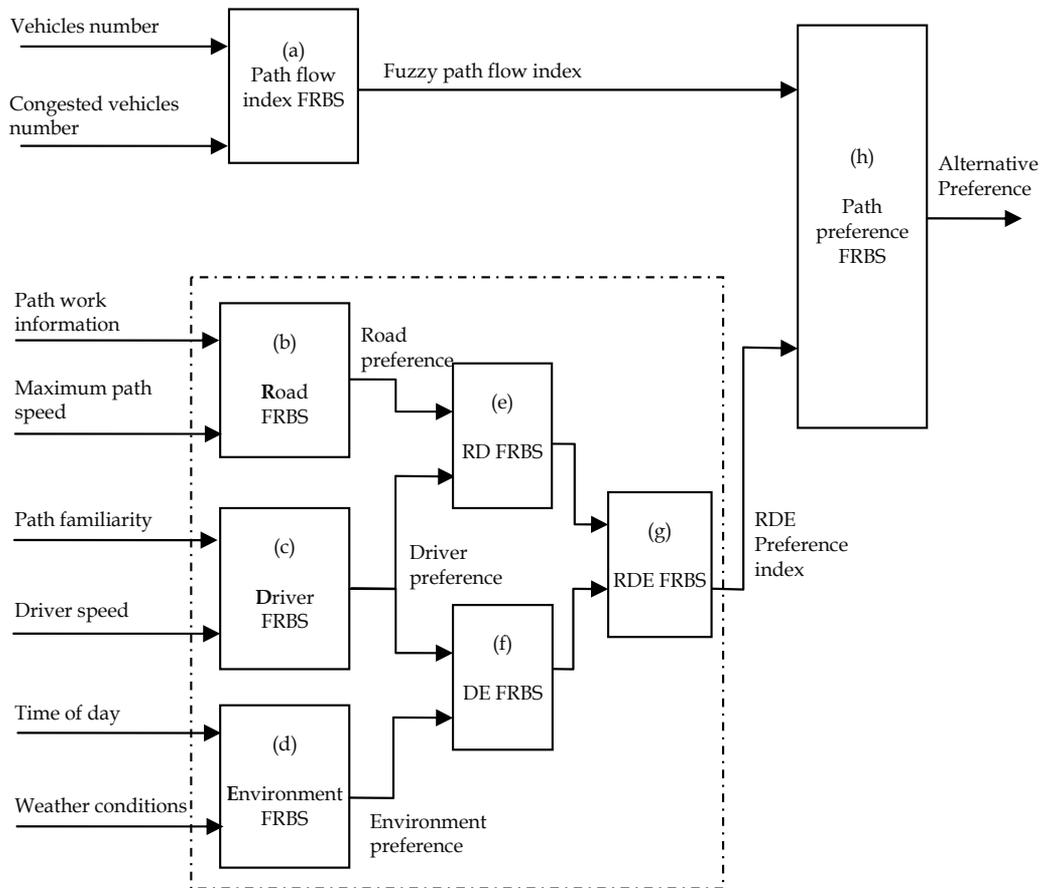


Fig. 5. Flowchart of the Hierarchical FRBS for an itinerary evaluation

3.2.1 Fuzzy data base

Classically, the crisp value corresponding to fuzzy variable is fuzzified into a set of couples (fuzzy set, membership degree) by matching the crisp value against the membership function of each fuzzy set of the fuzzy variable. In the following sub-section, we present inputs and output of each FRBS.

- a) Path flow index FRBS
 - Inputs:
 - Vehicle number: low, medium, and high
 - Congested vehicles number: low, medium, and high
 - Output:
 - Fuzzy path flow index: weak, medium, and strong
- b) Road FRBS
 - Inputs:
 - Path work information: No Path Work and Path Work
 - Maximum path speed: Slow, Medium, and High
 - Output:
 - Road preference: weak, medium, and strong
- c) Driver FRBS
 - Inputs:
 - Path familiarity: unfamiliar, medium, and familiar
 - Driver speed: slow, medium, and high
 - Output:
 - Driver preference: weak, medium, and strong
- d) Environment FRBS
 - Inputs:
 - Time of day: morning, midday, evening, and night
 - Weather conditions: bad, medium, and good
 - Output:
 - Environment preference: weak, medium, and strong
- e) RD FRBS
 - Inputs:
 - Road preference: weak, medium, and strong
 - Driver preference: weak, medium, and strong
 - Output:
 - RD preference: weak, medium, and strong
- f) DE FRBS
 - Inputs:
 - Driver preference: weak, medium, and strong
 - Environment preference: weak, medium, and strong
 - Output:
 - DE preference: weak, medium, and strong
- g) RDE FRBS
 - Inputs:
 - RD preference: weak, medium, and strong
 - DE preference: weak, medium, and strong
 - Output:
 - RDE preference: weak, medium, and strong
- h) Path preference FRBS
 - Inputs:
 - Fuzzy path flow index: weak, medium, and strong
 - RDE preference: weak, medium, and strong
 - Output:
 - Alternative preference: weak, medium, and strong

Each of these inputs is defined by trapezoidal and triangular membership functions.

3.2.2 Fuzzy rule base

The fuzzy rule base is partitioned in sets of rules representing basic behaviours and weighting coefficients attached to the fuzzy rules. The rule base of each FRBS is built by combination of input and output variables. Each base is generated by experts in the transportation area. The total number of rules is 66 rules. As for fuzzy rules, FRBSs use ordinary rules of the type 'IF *condition* THEN *action*', where the *condition* part is a

conjunction or disjunction of such propositions, and the *action* part is an elementary fuzzy proposition of the form 'fuzzy-variable is fuzzy-set'.

We present in the following table (table 1) the fuzzy rule base of path preference FRBS. In this rule base, we provide greater importance to the fuzzy path flow index input. In fact, we assume that this input have more importance than path preference according to the road, driver, and environment.

Rule n.	Inputs		Output
	PFI	RDE Preference	
1	weak	weak	weak
2	weak	medium	weak
3	weak	strong	weak
4	medium	weak	weak
5	medium	medium	medium
6	medium	strong	medium
7	strong	weak	strong
8	strong	medium	strong
9	strong	strong	strong

Table 1. Fuzzy rule base of path preference FRBS

3.2.3 Fuzzy inference and defuzzification

In the inference process, Mamdani (max-min) inference method is used. In max-min process, firstly the minimum membership values of the used rules outputs are selected (i.e. the minimum membership value of each input sub-set), then maximum of the minimums are determined (i.e. the maximum membership value of each output sub-set is selected). This process is applied to all valid rules in each FRBS and a geometric shape is obtained. Then, the defuzzification process is applied to get the crisp value. Due to output importance for make decision in route choice problem, the Center of Gravity CoG method (also known as centroid method) is used for defuzzification. The implied fuzzy set is transformed to a crisp output, calculating the area in a particular output membership function, by equation 1.

$$\mu^{crisp} = \frac{\sum_i b_i \int \mu_{(i)}}{\sum_i \int \mu_{(i)}} \quad (1)$$

Where μ^{crisp} is the crisp output value, b_i is the center of each output membership function (called also the weight factor).

3.3 Itinerary selection

Starting with idea that better path is one of k shortest paths regarding only path length; we propose to select the k shortest paths as k feasible paths. The k shortest paths problem is a natural and long-studied generalization of the shortest path problem, in which not one but several paths in decreasing order of length are required (Eppstein, 1998). So, each itinerary

from the set of k itineraries is evaluated with the hierarchical FRBS previously presented. The recommended itinerary is the one having the highest alternative preference (see figure 6). In order to do an adaptive routing according to the real-time information, this process is repeated before each intersection and after user request.

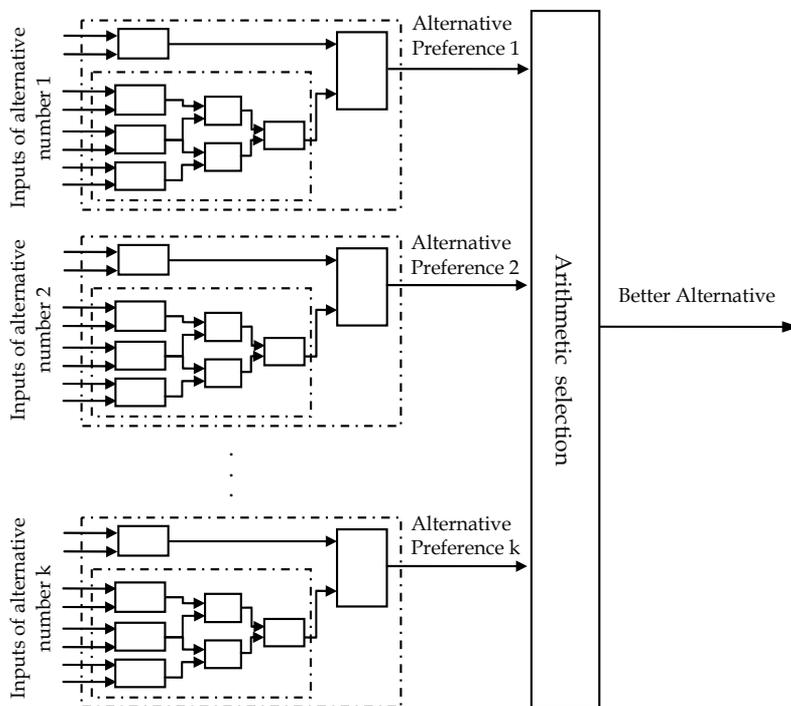


Fig. 6. Flowchart of the itinerary selection steps

4. Simulation and results' discussion

As we need to test our approach taking into account the variation of traffic information, and to visualize the evolution of the road network, we choose to develop a multiagent (MA) simulation model. In fact, it is very helpful in explaining collective behaviour as a result of individual actions. The MA simulation is based on the idea that it is possible to represent entities behaviours in one environment, and agent's interaction phenomenon. At each simulation step, each agent can receive a set of information describing the surrounding situation in the environment (Drogoul et al., 2003).

Since few years, we note the birth of some MA platforms. These platforms provide both a model for developing MA system and an environment for running distributed agent-based applications. To develop our simulator, we choose the MadKit platform as a generic MA platform (MultiAgent Development Kit) (Gutknecht & Ferber, 2001). Our choice is firstly based on comparison with other known MA platforms (Mulet et al., 2006) (Ricordel & Demazeau, 2000). Among its advantages, it is possible to make traffic services fully extensible and easily replaceable. MadKit allows a fast development of distributed agent system by providing standard services for communication and life cycle management

of the agents. This platform can support thousands of agents interact and perform tasks together by using a simple agent with reactive tasks. In addition, Madkit is a free platform with open source, and it can be programmatically extended using Java programming language. We use in the following multiagent simulation the version 4.1.2 of November 2005. The platform is stable since this date. We use also the TurtleKit tool (Michel et al., 2005) presented as a reactive agent execution tool that runs on the 'synchronous engine' of MadKit platform. This tool aims at providing to advanced users the simplicity of a Logo simulation model while proposing flexibility, modularity and extensibility.

In order to control simulations, we use a launcher agent having the role to set up, launch, and manage turtles (see figure 7a). At each simulation step, an agent receives a set of information describing the surrounding environment situations. Figure 7b presents an example of virtual maps, created by the observer agent from TurtleKit tool. Simulation environments contain roads with a fixed number of lanes and vehicles having each one both origin road and destination road.

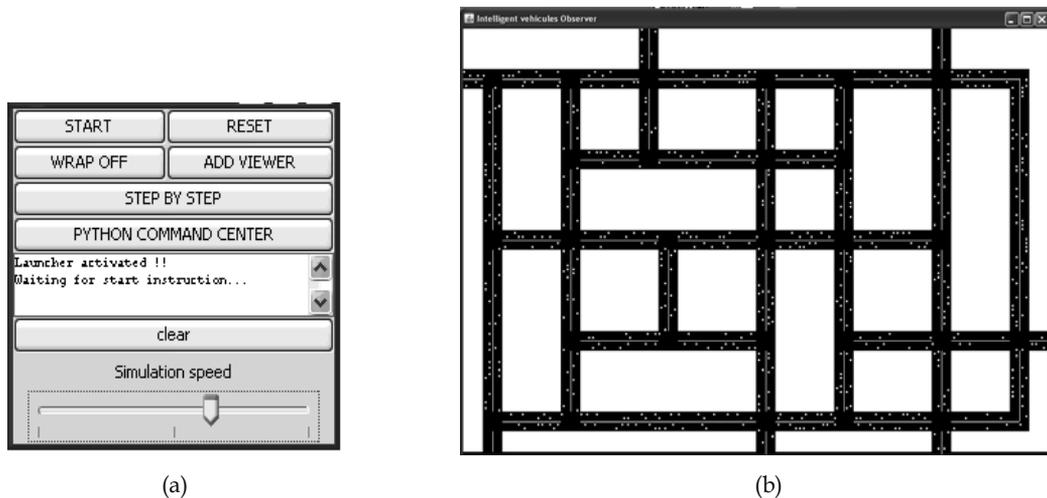


Fig. 7. The simulator (a) Launcher agent, (b) Example of simulation environment

Regarding the hierarchical FRBS, we use the Matlab Fuzzy Logic Toolbox. In this case, a connection between Java and Matlab should be done. We use the JMatLink library that provides some native method for connection (Müller & Waller, 1999). JMatLink includes methods and objects that allow Java to initialize a workspace, write data members of any format to the workspace, read from the work space, and execute command line functions.

In the other hand, for the k shortest paths algorithm, we use an implemented algorithm with Java provided by JGraphT¹. It is a free Java graph library that provides mathematical graph-theory objects and algorithms. The k shortest paths algorithm used is similar to Bellman-Ford except that at each iteration, instead of storing only the best path, the

¹ <http://www.jgrapht.org>

algorithm stores the k best paths from origin vertex to destination vertex in an increasing order of length.

We represent the road network, shown in figure 7, by a directed weighted graph $G=(V, E)$ where V is a set of vertices indicating intersections (crossroads) and E is a set of directed edges $e=(v_i, v_j)$ indicating roads (see figure 8). Each road direction between two adjacent vertices is represented by one directed edge. The weight of each edge is the road length. Given two vertices s and t , the k shortest paths algorithm provides k paths from s to t in increasing order of length. To better understand the broad outline of the system, we propose a class diagram, as the static structure, illustrated by figure 9.

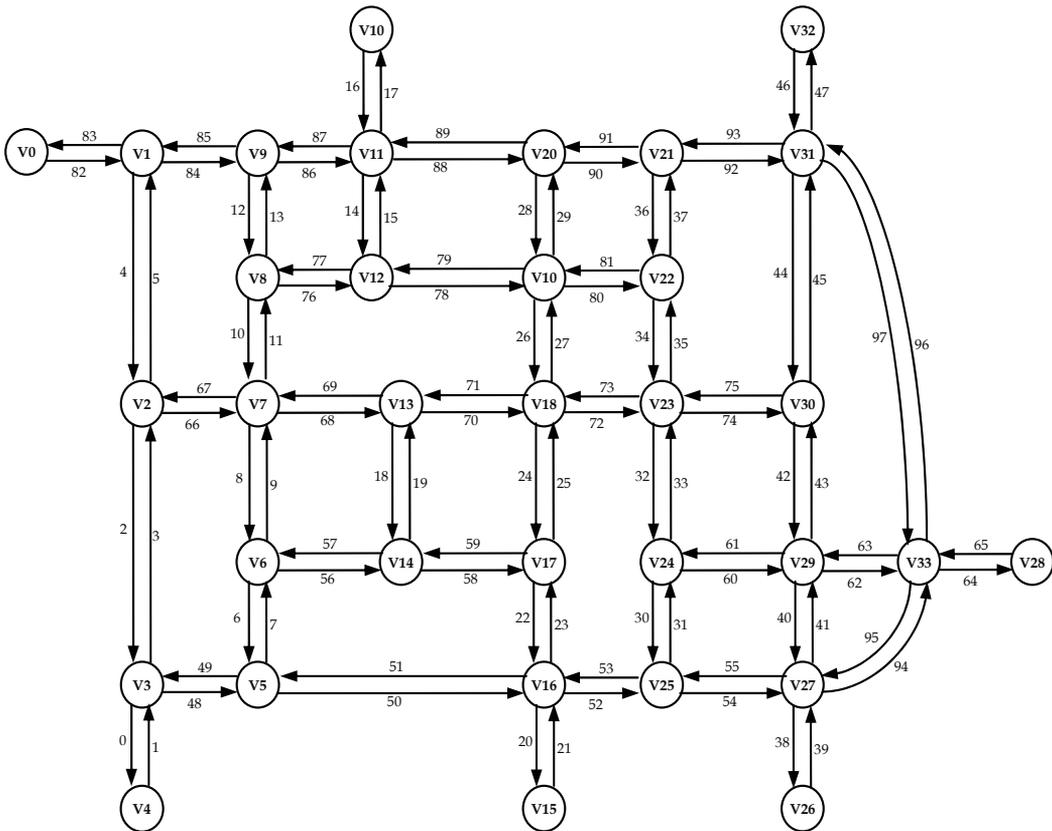


Fig. 8. An example of road network presented as directed weighted graph

Series of simulations are performed, with the environment of simulation presented in figures 7 and 8 (98 roads with 2 lanes and 34 intersections), using different couples (road origin, road destination) with random velocity. During simulation, we add randomly other vehicles having random origin and destination roads (using Random class of Java with uniform distribution). Furthermore, some congested and jammed area in intersection can occur. In order to represent the reality, some vehicles have a low probability to become jammed vehicles. A vehicle stops when perceiving halted ones and it continues moving in a

free path. In carrying out the k shortest paths algorithm, we eliminate paths having some loops (i.e. visit of an intersection more than one time).

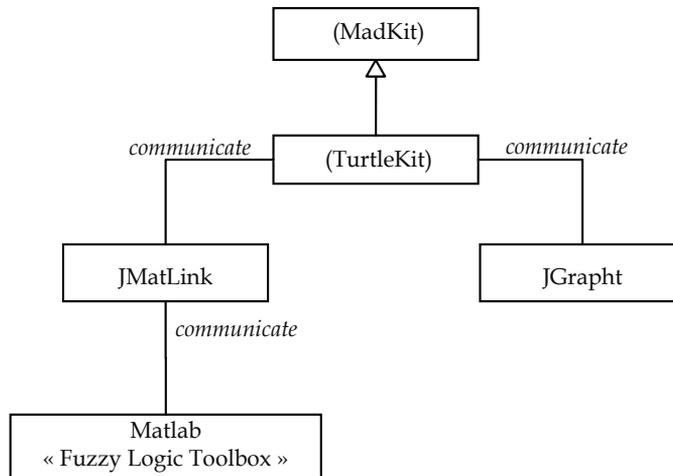


Fig. 9. Class diagram

In order to present some collaboration between agents, figure 10 illustrates a sample of communication messages between interface agent and decision making agent. In this sample, although possible paths have the same length, the driver chooses the second path according to the alternative preference computed with the hierarchical FRBS. This decision is occurred following a congestion in intersection represented by vertex number 13.

```

[Interface Agent] I'm in road 7 and I want to reach Road 24 ?
[DMA] 3 possible Paths ordered in increasing length:
[DMA] alternative 1: by Roads {7 - 56 - 19 - 70 - 24} - distance: 120
[DMA] alternative 2: by Roads {7 - 56 - 58 - 25 - 24} - distance: 120
[DMA] alternative 3: by Roads {7 - 9 - 68 - 70 - 24} - distance: 120
[DMA] Alternative preference for 1st alternative is 0.24
[DMA] Alternative preference for 2nd alternative is 0.71
[DMA] Alternative preference for 3rd alternative is 0.20
[DMA] The recommended path: alternative 2
[Interface Agent] Next road is 56
[Interface Agent] Turn right
  
```

Fig. 10. Sample of agents' communication

A comparison has been done between the shortest itinerary (according to the length) and the itinerary selected after applying hierarchical FRBS. Figure 11 presents the number of times when the route choice is the first shortest path, or the second shortest path, etc. 20000 vehicles with different couple origin-destination roads have been simulated with different k value ($k=3$, $k=5$, $k=7$) using the road network presented in figure 8.

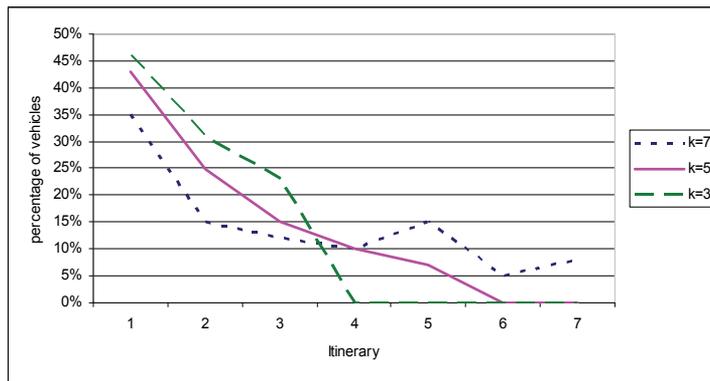


Fig. 11. Influence of parameter k on vehicles distribution

In order to view the modification of selected itinerary during the travel, another comparison has been done between the pre-selected itinerary and the real one. We remark that at about 38%, the itinerary has been changed during travel according to the real-time update of traffic information due to the reselection at each intersection. This confirms the adaptability of our hierarchical fuzzy multiagent system compared with its first version (Kammoun et al., 2007) (Kallel et al., 2008).

We remark also that:

- Interactions between different agents accomplished in a series of running simulations show the influence of an early detection of congestion and jam cases.
- More k is greater more the route selection is better.
- As regards the technical features, java programming language has some limitations to associate a thread to each agent. So, the management of a great number of concurrent threads is not efficient. The generic multiagent platform chosen can solve the problem. In fact, Madkit proposes a synchronous engine which can manage a big number (hundreds or even thousands) of agents in one thread. In spite of IVA agent is a MAS itself and the decision making agent of each IVA run the k shortest path algorithm, Madkit can support simulation in larger scenarios with a large number of agents.

Then, the results show the effectiveness of the hierarchical FRBS in the case of many selection criteria and the effectiveness of multiagent modelling and simulation in order to improve traffic management.

5. Conclusion and future work

In this paper, we present a hierarchical fuzzy rule-base system for route choice problem in order to evaluate one possible itinerary based on real-time information, driver's preferences, and other selection criteria. A cooperative route choice has been developed; it is based on an organisational multiagent architecture of the road network. Finally, several multiagent simulations with a virtual road network have been achieved in order to test the efficiency of the proposed hybrid method. These simulations are realized using 'TurtleKit' tool under the generic multiagent platform 'MadKit'. This chapter attests once more how multiagent modelling and simulation are among best achievable options used to solve complex

problem on road network. Furthermore, this chapter presents the feasibility of hierarchical fuzzy rule-base system to deal with a large number of selection criteria in order to improve the accuracy of route choice for a better management of road network.

As perspectives, we intend in the near future to apply learning method, such as the advanced one detailed in (Kallel et al., 2006), in order to optimize the rule base and tune the membership functions of the data base. Furthermore, real-time path learning for vehicle path planning can be added.

6. Acknowledgment

The authors thank the Tunisian General Direction of Scientific Research and Technological Renovation (DGRSRT), under the ARUB program 01/UR/11-02, Tunisia.

7. References

- Akiyama, T. & Tsuboi, H. (1996). Analysis of multi-route choice behaviour by fuzzy reasoning models, *Proceedings of the 3rd Symposium on Application of Fuzzy Sets Theory to Civil and Architectural Engineering*, pp. 31-38
- Akiyama, T.; Nakamura, K. & Sasaki, T. (1993). Traffic diversion model on urban expressway by fuzzy reasoning. *Proceedings of the 6th World Conference of Transport Research*, pp. 1011-1022, Lyon
- Alimi, A.M. (1997). A neuro-fuzzy approach to recognize Arabic handwritten characters. *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 3, pp.1397-1400, Houston, USA
- Arslan, T. & Khisty, C.J. (2005). A rational reasoning method from fuzzy perceptions in route choice. *Fuzzy Sets and Systems*, Vol. 150, No. 3, 419-435, ISSN 0165-0114
- Bekhor, S.; Ben-Akiva, M.E. & Ramming, M.S. (2002). Adaptation of logit kernel to route choice situation. *Transportation Research Record*, No. 1805, 78-85, ISSN 0361-1981
- Ben-Akiva, M. & Lerman, S.R. (1985). *Discrete Choice Analysis: Theory and Application to Travel Demand*, MIT Press, Cambridge
- Bierlaire, M. & Frejinger, E. (2008). Route choice modeling with network-free data. *Transportation Research Part C: Emerging Technologies*, Vol. 16, No. 2, 187-198
- Drogoul, A.; Vanbergue, D. & Meurisse, T. (2003). Multi-agent based simulation: Where are the agents?, In: *Multi-Agent-Based Simulation II* (LNCS), Sichman, J.S.; Bousquet, F.; Davidsson, P., (Eds.), Vol. 2581, 43-49, Springer, ISBN: 978-3-540-00607-7
- Eppstein, D. (1998). Finding the k shortest paths. *SIAM Journal on Computing*, Vol. 28, No. 2, 652-673
- Gong, J.; Yu, Z., & Chen, N. (2007). An analysis of drivers' route choice behavior in urban road networks based on GPS data, *Proceedings of the 1st International Conference on Transportation Engineering*, pp. 515-520, ASCE, China
- Gutknecht, O. & Ferber, J. (2001). The madkit agent platform architecture, In: *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems* (LNCS), Vol. 1887, 48-55, Springer, ISBN: 978-3-540-42315-7
- Hawas, Y.E. (2004). Development and calibration of route choice utility models: Neuro-fuzzy approach. *Journal of Transportation Engineering*, Vol. 130, No. 2, 171-182.

- Henn, V. (2000). Fuzzy route choice model for traffic assignment. *Fuzzy Sets and Systems*, Vol. 116, No. 1, 77-101, ISSN: 0165-0114
- Kallel, I. & Alimi, A.M. (2006). MAGAD-BFS: A learning method for beta fuzzy systems based on a multi-agent genetic algorithm. *International Journal of Soft Computing*, Vol. 10, No. 9, 757-772, ISSN: 1432-7643
- Kallel, I.; Jelleli, T. & Alimi, A.M. (2005). Hierarchical FLS design using multi-agent genetic approach, *Proceedings of the 1st International Workshop on Genetic Fuzzy Systems GFS*, pp. 142-147, Granada, Spain
- Kallel, I.; Mezghani, S. & Alimi, A.M. (2008). Towards a Fuzzy Evaluation of the Adaptivity Degree of an Evolving Agent, *Proceedings of the 3rd International Workshop on Genetic and Evolving Fuzzy Systems GEFS*, pp. 29-34, March 2008, Bommerholz, Germany
- Kammoun, H.M.; Kallel, I. & Alimi, A.M. (2007). A joint hierarchical fuzzy-multiagent model dealing with route choice problem RoSFuzMAS, *Proceedings of the 4th International Conference on Informatics in Control, Automation and Robotics ICINCO*, pp. 394-397, May 2007, Angers, France
- Kammoun, H.M.; Kallel, I.; Casillas, J. & Alimi, A.M. (2008). A road traffic multiagent simulation using turtleKit under madKit, *9th International Conference on Artificial Intelligence and Soft Computing ICAISC*, In: *Challenging Problems of Science series, Computational Intelligence: Method and Applications*, Exit publisher, pp. 503-514, June 2008, Poland.
- Lee, M.L.; Chung, H.Y. & Yu, F.M. (2003). Modeling of hierarchical fuzzy systems. *Fuzzy Sets and Systems*, Vol. 138, No. 2, 343-361, ISSN: 0165-0114
- Lotan, T. & Koutsopoulos, H.N. (1993). Models for route choice behavior in the presence of information using concepts from fuzzy set theory and approximate reasoning. *Transportation*, Vol. 20, No. 2, 129-155, ISSN 0049-4488
- Lotan, T. (1998). Modeling discrete choice behavior based on explicit information integration and its application to the route choice problem. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, Vol. 28, No. 1, 100-114, ISSN: 1083-4427
- Michel, F.; Beurrier, G. & Ferber, J. (2005). The turtleKit simulation platform: Application to complex systems, *Proceedings of the 1st IEEE International Conference on Signal-Image Technology & Internet-Based Systems SITIS*, pp. 122-127, November 2005, Yaounde, Cameroon
- Mulet, L.; Such, J.M. & Alberola, J.M. (2006). Performance evaluation of open-source multiagent platforms, *Proceedings of the 5th International joint Conference on Autonomous Agents and MultiAgent Systems AAMAS*, pp. 1107-1109, May 2006, Hakodate, Japan
- Müller, S. & Waller, H. (1999). Efficient integration of real-time hardware and web based services into MATLAB, *Proceedings of the 11th European Simulation Symposium and Exhibition ESS*, October 1999, Nuremberg, Germany
- Panwai, S. & Dia, H. (2006). A fuzzy neural approach to modelling behavioural rules in agent-based route choice models, *Proceedings of the 4th International Workshop on Autonomous Agents in Traffic and Transportation ATT@AAMAS*, pp. 70-79, May 2006, Hakodate, Japan
- Peeta, S. & Yu, J.W. (2004). Adaptability of a hybrid route choice model to incorporating driver behavior dynamics under information provision. *IEEE Transactions on*

- Systems, Man, and Cybernetics Part A: Systems and Humans*, Vol. 34, No. 2, 243-256, ISSN: 1083-4427
- Rattasiri, W. & Halgamuge, S.K. (2003). Computationally advantageous and stable hierarchical fuzzy systems for active suspension. *IEEE Transactions on Industrial Electronics*, Vol. 50, No. 1, 48-61, ISSN: 1557-9948
- Ricordel, P.M. & Demazeau, Y. (2000). From analysis to deployment: A multi-agent platform survey, In: *Engineering Societies in the Agents World (LNCS)*, Vol. 1972, 93-105, Springer, ISBN: 978-3-540-41477-3
- Ridwan, M. (2004). Fuzzy preference based traffic assignment problem. *Transportation Research Part C: Emerging Technologies*, Vol. 12, No. 3-4, 209-233, ISSN: 0968-090X
- Teodorović, D. & Kikuchi, S. (1990). Transportation route choice model using fuzzy inference technique, *Proceedings of the 1st International Symposium on Uncertainty Modeling and Analysis*, pp. 140-145
- Teodorovic, D. (1999). Fuzzy logic systems for transportation engineering: The state of the art. *Transportation Research Part A: Policy and practice*, Vol. 33, No. 5, 337-364, ISSN 0965-8564
- Teodorović, D. (2003). Transport modeling by multi-agent systems: A swarm intelligence approach. *Transportation Planning and Technology*, Vol. 26, No. 4, 289-312, ISSN 0308-1060
- Wooldridge, M. (2002). *An introduction to multiagent systems*, John Wiley and Sons, ISBN 0-471-49691-X, England
- Zadeh, L. (1965). Fuzzy sets. *Information and Control*, Vol. 8, 338-353, ISSN: 0019-9958
- Zito, R., D'Este, G., & Taylor, M.A.P. (1995). Global positioning systems in the time domain: How useful a tool for intelligent vehicle-highway systems? *Transportation Research, Part C: Emerging Technologies*, Vol. 3, No. 4, 193-209, ISSN: 0968-090X

The Artificial Neural Networks applied to servo control systems

Yuan Kang¹, Yi-Wei Chen², Ming-Huei Chu², Der-Ming Chry²

¹Department of Mechanical Engineering, Chung Yuan Christian University

²Department of Mechatronical Engineering, TungNan University

Abstract: This chapter utilizes the direct neural control (DNC) based on back propagation neural networks (BPN) with specialized learning architecture applied to the speed control of DC servo motor. The proposed neural controller can be treated as a speed regulator to keep the motor in constant speed, and be applied to DC servo motor speed control. The proposed neural control applied to position control for hydraulic servo system is also studied for some modern robotic applications.

A tangent hyperbolic function is used as the activation function, and the back propagation error is approximated by a linear combination of error and error's differential. The simulation and experiment results reveal that the proposed neural controller is available to DC servo control system and hydraulic servo system with high convergent speed, and enhances the adaptability of the control system.

Keywords: Neural networks, DC servo motor, Speed regulator, Speed control, Hydraulic servo System

1. Introduction

The neural controls have been put into use in various fields owing to their capability of on line learning and adaptability. In recent years, many learning strategies for neural control have been proposed and applied to some specified nonlinear control systems to overcome the unknown model and parameters variation problems. In this chapter, a direct neural controller with specialized learning architecture is introduced and applied to the DC servo and hydraulic servo control systems.

The general learning architecture and the specialized learning architecture are proposed and studied in early development of neural control [1]. The general learning architecture shown in Fig. 1, uses neural network to learn the inverse dynamic of plant, and the well-trained network is applied to be a feed forward controller. In this case, the general procedure may not be efficient since the network may have to learn the responses of the plant over a larger operational range than is actually necessary. One possible solution to this problem is to combine the general method with the specialized procedure, so that an indirect control strategy for general learning was proposed, which is shown in Fig. 2. For a general learning architecture with some specialized procedures, the off line learning of the

inverse dynamic of plant still have to learn the responses of the plant over a larger operational range.

The specialized learning architecture shown in Fig. 3, trains the neural controller to operate properly in regions of specialization only. Training involves using the desired response as input to the network. The network is trained to find the plant input, which drives the system output to the desired command. This is accomplished by using the error between the desired and actual responses of the plant to adjust the weights of the network with a steepest descent procedure. The weights are adjusted to decrease the errors during iterations. This procedure requires knowledge of the Jacobian of the plant.

There are two strategies to facilitate the specialized learning, one is direct control shown in Fig. 4 and the other is indirect control shown in Fig. 5 [2]. In the former, the plant can be viewed as an additional but no modifiable layer of the neural network, and the dash line of Fig. 4 means the weights update need the knowledge of plant. The latter, which has been used in many applications [3-5], is a two-step process including identification of dynamics of plant and control.

In the indirect control strategy, a sub-network (called "emulator") is required to be trained before the control phase, and the quality of the trained emulator is crucial to the controlling performance. It is therefore very important that the data sets for training the emulator must cover a sufficiently large range of input and output data pairs, but if some of these values are outside the input range that was used during the emulator's training, the back propagation through the emulator fails, causing poor or even unstable control performance.

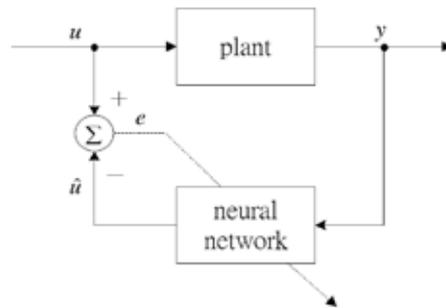


Fig. 1. The general learning architecture

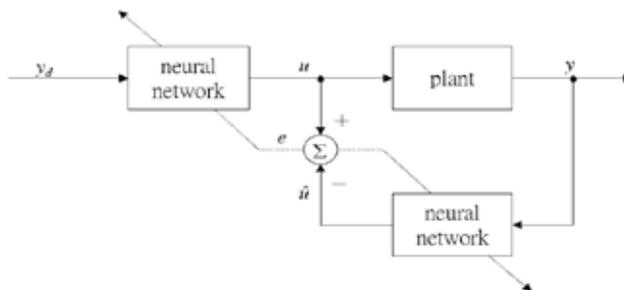


Fig. 2. The indirect control for general learning architecture

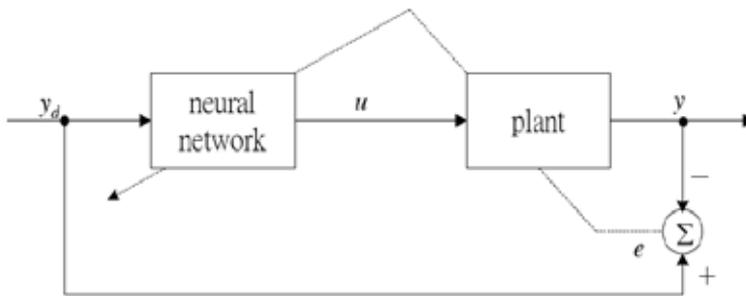


Fig. 3. The specialized learning architecture

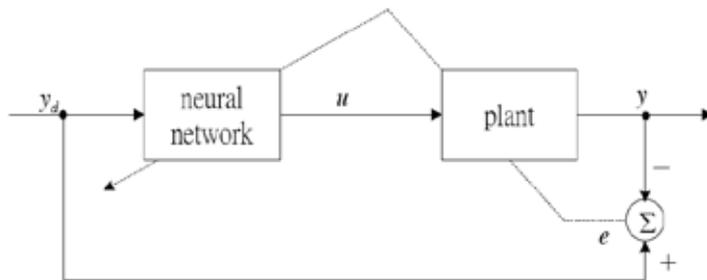


Fig. 4. The direct control for specialized learning architecture

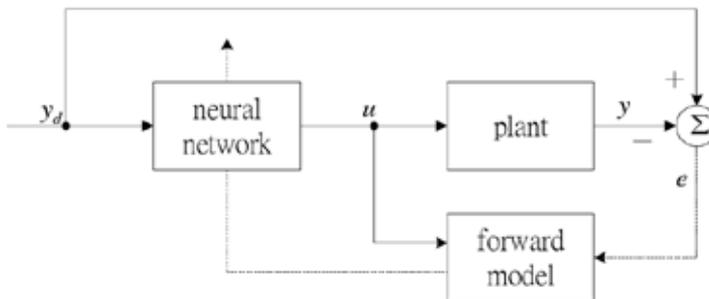


Fig. 5. The indirect control for specialized learning architecture

The direct control strategy can avoid this problem if a priori qualitative knowledge or Jacobian (the partial differential of plant output to input) of the plant is available. But it is usually difficult to approximate the Jacobian of an unknown plant. This chapter utilizes the direct neural control (DNC) based on back propagation neural networks (BPN) with specialized learning architecture applied to the speed controls of DC servo motor. The approximation methods of Jacobian are introduced for direct neural control scheme. The direct control strategies with the approximation methods have been successfully applied to DC servo and hydraulic servo control systems. The proposed neural controller also can be

treated as a speed regulator to keep the motor in constant speed. The corresponding performances are investigated and discussed.

2. The direct neural controller

2.1 The structure of direct neural control

A direct neural controller with three layers was shown in Fig. 6. A three layers neural network with one hidden layer is sufficient to compute arbitrary decision boundaries for the outputs [6]. Although a network with two hidden layers may give better approximation for some specific problems, but the networks with two hidden layers are more prone to fall into local minima [7], and more CPU time is needed. In the following section, a back propagation network (BPN) with single hidden layer is considered.

Another consideration is the right number of units in a hidden layer. Lippmann [8] has provided comprehensive geometrical arguments and reasoning to justify why the maximum number of units in a single hidden layer should equal to $M(N+1)$, where M is the number of output units and N is the number of input units. Zhang et al. [2] have tested different numbers units of the single hidden layer for a ship tracking control system. It was found that a network with three to five hidden units is often enough to give good results.

The structure of direct neural control is shown in Fig. 7. The proposed neural network has three layers with two units in the input layer, one unit in the output layer and fine number of units in the hidden layer. The $r = X$, X and $y = X$ denote as the command input, output of the reference model and the output of the plant respectively. The two inputs of the network are the error e and its differential \dot{e} between X_R and X_P .

The reference model can be designed according to a second order dynamic model; the damping ratio and natural frequency can be determined based on the specified performance index of control system.

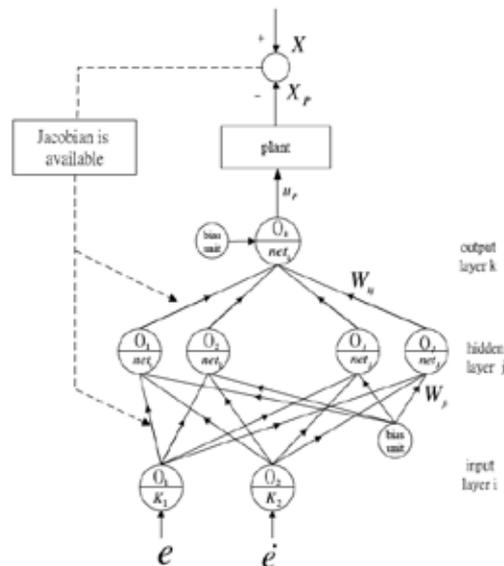


Fig. 6. A direct neural controller with three layers

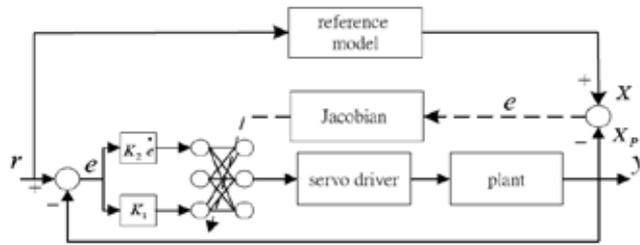


Fig. 7. The structure of a direct neural control system

2.2 The algorithms for direct neural controller

The proposed neural network shown in Fig. 6 has three layers with two units in the input layer, one unit in the output layer and right number of units in the hidden layer. The X_r , X and X_p denote the required command input, output of the reference model and the output of the controlled plant respectively. The two inputs of the network are the error and its differential between X_r and X_p . The reference model can be designed according to a second order transfer function; the damping ratio and natural frequency can be defined based on the specified performance index. The algorithms and weights update equations of the proposed direct neural controller are described by the following equations. The proposed direct neural controller has the hidden layer (subscript "j"), output layer (subscript "k") and layer (subscript "i"). The input signal is multiplied by gains K_1 , K_2 at the input layer, in order to be normalized between +1 and -1. A tangent hyperbolic function is used as the activation function of the nodes in the hidden and output layers, the number of units in hidden layer equals to J , the number of units in input layer equals to I , and the number of units in output layer equals to K , the net input to node j in the hidden layer is:

$$net_j = \sum(W_{ij} \cdot O_i) + \theta_j \quad i=1,2,\dots,I, j=1,2,\dots,J \tag{1}$$

the output of node j is

$$O_j = f(net_j) = \tanh(\beta \cdot net_j) \tag{2}$$

where $\beta > 0$, the net input to node k in the output layer is

$$net_k = \sum(W_{kj} \cdot O_j) + \theta_k \quad j=1,2,\dots,J, k=1,2,\dots,K \tag{3}$$

the output of node k is

$$O_k = f(net_k) = \tanh(\beta \cdot net_k) \tag{4}$$

The output O_k of node k in the output layer is treated as the control input u_p of the system for a single-input and single-output system. As expressed equations, W_{ji} represent the connective weights between the input and hidden layers and W_{kj} represent the connective weights between the hidden and output layers. θ_i and θ_k denote the bias of the hidden and output layers, respectively.

For the N th sampling time, the error function is defined as

$$E_N = \frac{1}{2}(X_N - X_{pN})^2 = \frac{1}{2}e_N^2 \quad (5)$$

where X_N and X_{pN} denote the outputs of the reference model and the outputs of the controlled plant at the N th sampling time, respectively. The weights matrix is then updated during the time interval from N to $N+1$.

$$\Delta W_N = W_{N+1} - W_N = -\eta \frac{\partial E_N}{\partial W_N} + \alpha \cdot \Delta W_{N-1} \quad (6)$$

where η is denoted as learning rate and α is the momentum parameter. The gradient of E_N with respect to the weights W_{kj} is determined by

$$\frac{\partial E_N}{\partial W_{kj}} = \frac{\partial E_N}{\partial net_k} \frac{\partial net_k}{\partial W_{kj}} = \delta_k O_j \quad (7)$$

and δ_k is defined as

$$\begin{aligned} \delta_k &= \frac{\partial E_N}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial X_p} \frac{\partial X_p}{\partial u_p} \frac{\partial u_p}{\partial O_n} \frac{\partial O_n}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial O_n} \frac{\partial O_n}{\partial net_k} \\ &= \sum_n \frac{\partial E_N}{\partial O_n} \beta(1 - O_n^2) \quad n = 1, 2, \dots, K \end{aligned} \quad (8)$$

where $\partial X_p / \partial u_p$ is defined to be the Jacobian of plant. Assume the Jacobian of the plant can be evaluated. The δ_k can be solved.

Similarly, the gradient of E_N with respect to the weights, W_{ji} is determined by

$$\frac{\partial E_N}{\partial W_{ji}} = \frac{\partial E_N}{\partial net_j} \frac{\partial net_j}{\partial W_{ji}} = \delta_j O_i \quad (9)$$

$$\begin{aligned}\delta_j &= \frac{\partial E_N}{\partial net_j} = \sum_m \frac{\partial E_N}{\partial net_k} \frac{\partial net_k}{\partial O_m} \frac{\partial O_m}{\partial net_j} \\ &= \sum_m \delta_k W_{km} \beta(1 - O_j^2) \quad m = 1, 2, \dots, J\end{aligned}\quad (10)$$

The weight-change equations on the output layer and the hidden layer are

$$\Delta W_{kj,N} = -\eta \frac{\partial E_N}{\partial W_{kj,N}} + \alpha \cdot \Delta W_{kj,N-1} = -\eta \delta_k O_j + \alpha \cdot \Delta W_{kj,N-1}\quad (11)$$

$$\Delta W_{j,N} = -\eta \frac{\partial E_N}{\partial W_{j,N}} + \alpha \cdot \Delta W_{j,N-1} = -\eta \delta_j O_j + \alpha \cdot \Delta W_{j,N-1}\quad (12)$$

where δ_j and δ_k can be evaluated from Eqs.(24) and (21). The connective weights in the neural network are updated during the time interval from N to N+1 .

$$W_{kj,N+1} = W_{kj,N} + \Delta W_{kj,N}\quad (13)$$

$$W_{j,N+1} = W_{j,N} + \Delta W_{j,N}\quad (14)$$

A tangent hyperbolic function is used as the activation function, so that the neural network controller output $O_k = u_p$ evaluated from Eq. (4) is between A1 and +1, which is multiplied by the scaling factor K_o to be the input of plant. The weights and biases is initialized randomly in the interval between +0.5 and A0.5, and updated by Eqs. (13) and (14).

2.3 The on line trained adaptive neural controller

The Jacobian of plant needs to be available to Eq.(8) for back propagation algorithm. However, the exact $\partial X_p / \partial u_p$ is difficult to determine because of the unknown plant dynamics. Two differential approximations are presented [1] by slightly changing each input to the plant at an operating point, and measuring the changes in the output. The jacobian is denoted by

$$\frac{\partial X_p}{\partial u_p} = \frac{X_p(u_p + \Delta u_p) - X_p(u_p)}{\Delta u_p}\quad (15)$$

Alternatively, by comparing the changes of the differential related variables with values in the previous iteration, the differential can be approximated using the relationship

$$\frac{\partial X_p(N)}{\partial u_p(N)} = \frac{X_p(N) - X_p(N-1)}{u_p(N) - u_p(N-1)} \quad (16)$$

It has been observed in earlier reported simulations [2] that the use of approximation (15) or (16) often causes ambiguity for network training when the controlled plant has large inertia or when disturbances are added. Ambiguity in training contrary to what would be expected from a clear understanding of the situation being investigated. A simple sign function proposed by Zhang et al. [2] is applied to approximate the Jacobian of plant, and called on-line trained adaptive neural controller for industrial tracking control application. Therefore, the differential $\frac{\partial X_p(N)}{\partial u_p}$ is approximated by the ratio of the signs of the changes in $X(N)_p$ and $u(N)_p$. The term $\frac{\partial X_p(N)}{\partial u_p(N)}$ is replaced by its sign, so that Eq.8 takes the form

$$\delta_k = \frac{\partial E_N}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial X_p} \text{sign}\left(\frac{\partial X_p}{\partial u_p}\right) \frac{\partial u_p}{\partial O_n} \frac{\partial O_n}{\partial net_k} \quad (17)$$

The clear knowledge of how the control signal $u(N)_p$ influence the plant outputs $X(N)_p$ will provide the required sign information. Therefore $X(N)_p / u(N)_p < 0$ leads to

$$\text{sign}(\partial X_p(N) / \partial u_p(N)) = -1 \quad (18)$$

and $\partial X_p(N) / \partial u_p(N) > 0$ leads to

$$\text{sign}(\partial X_p(N) / \partial u_p(N)) = 1 \quad (19)$$

Using Eq.(17) with the given differential signs provide in Eq.(18) and (19), the neural controller will effectively output control signals with the correct direction according to the plant output error $e(N)$.

2.4 The approximation of Jacobian

An accurate tracking response needs to increase the speed of convergence. However, for a single-input and single-output control system, the sensitivity of E_N with respect to the network output O_k can be approximated by a linear combination of the error and its differential according to the & adaptation law [8] shown as below

$$\frac{\partial E_N}{\partial O_k} = K_3 e + K_4 \frac{de}{dt} \quad (20)$$

where K_3 and K_4 are positive constants, so that Eq.8 takes the form

$$\delta_k = \frac{\partial E_N}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial X_p} \frac{\partial X_p}{\partial u_p} \frac{\partial u_p}{\partial O_n} \frac{\partial O_n}{\partial net_k} = \sum_n (K_3 e + K_4 \frac{de}{dt}) \frac{\partial O_n}{\partial net_k} \tag{21}$$

Example 1.

A direct neural controller applied to DC servo speed control system is shown in Fig. 8. Assume the voltage gain of servo amplifier is unity. The gain of speed sensor is 0.001V rpm , the first order dynamic model of DC servo motor is

$$M(S) = \frac{10000}{S + 1}$$

According to Eq. 23, the direct neural controller using & adaptation law with three layers and five hidden neurons shown in Fig. 9. is used to control and regulate the motor speed.

$$\frac{\partial E_N}{\partial O_k} = K_3 e + K_4 \frac{de}{dt} \tag{23}$$

According to Fig. 8, the neural control system without reference model is a self-tuning type adaptive control, so that $K_1 = K_3$ and $K_2 = K_4$ conditions can be applied. The $K_1 = 0.6$ and $K_2 = 0.05$ can be determined for input signals normalization. The learning rate $\eta = 0.1$, sampling time=0.0001s, $K_1 = K_3 = 0.6$, $K_2 = K_4 = 0.05$ and the step command of 1V(1000rpm) assigned for simulation phase, and the simulation results are shown in Fig. 10, Fig. 11, and Fig. 12. The simulation results show that the connective weights will be convergent. The time response for r, u shows that the network will keep an appropriate output voltage signal to overcome the speed (motional) voltage, which is generated by the rotating armature. Similarly, the neural controller can provide output to overcome the torque load and friction. This is similar to a PI controller, but the neural controller can enhance the adaptability and improve performance of control system.

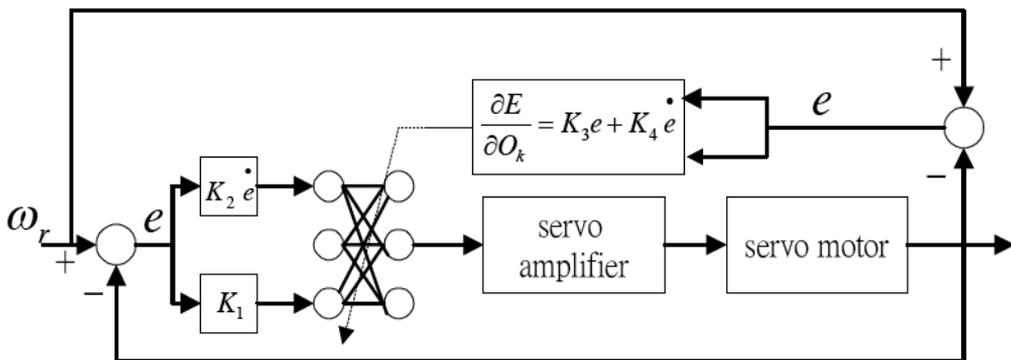


Fig. 8. The speed control system with direct neural controller

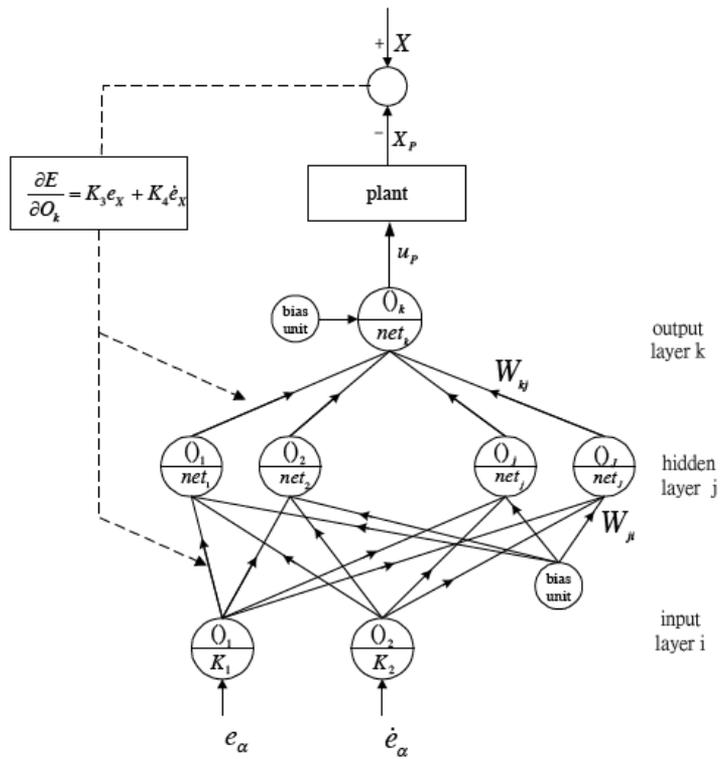


Fig. 9. The direct neural controller

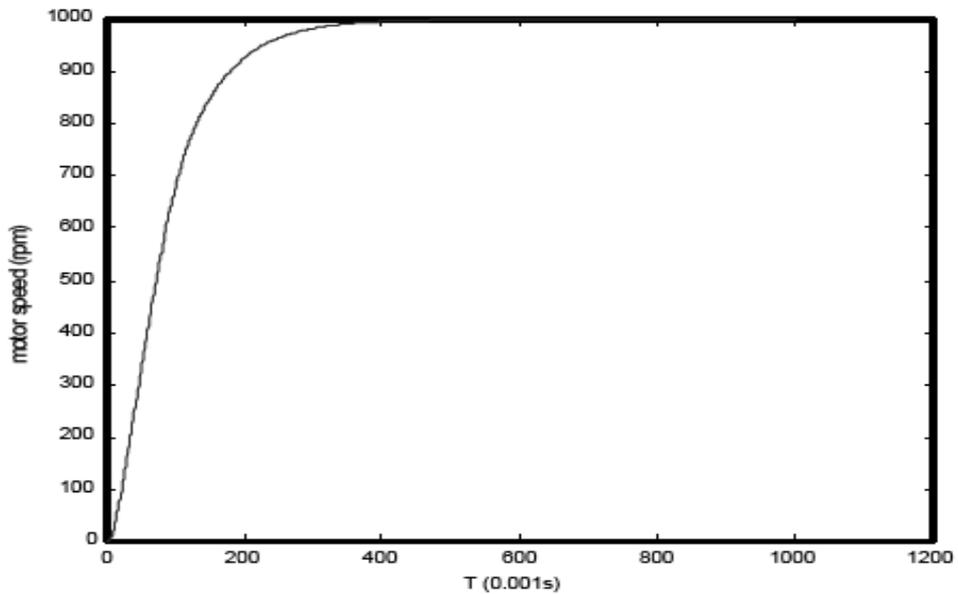


Fig. 10. Speed response for DC servo motor

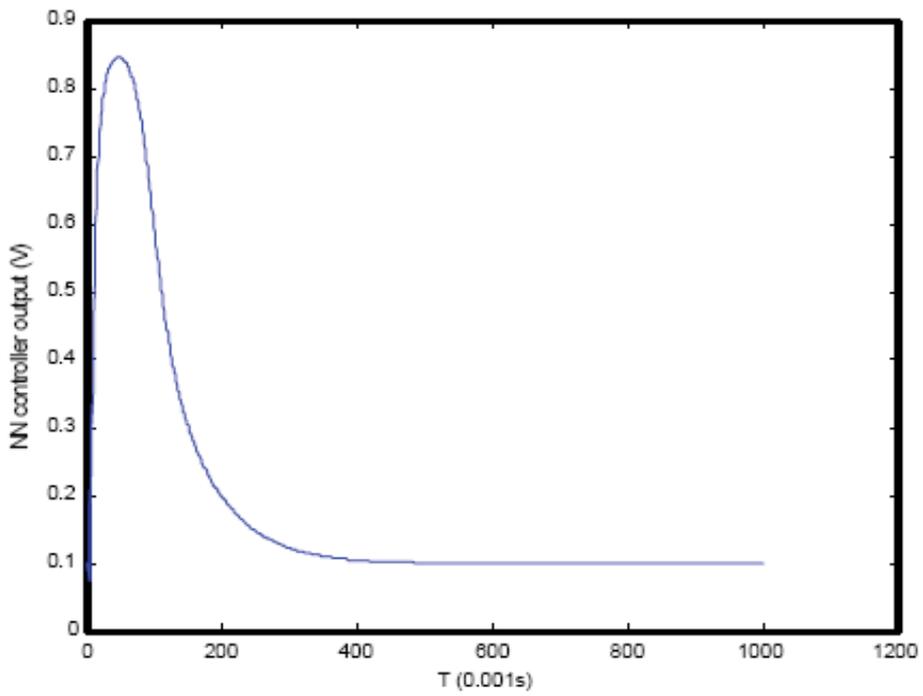


Fig. 11. The time response for control input

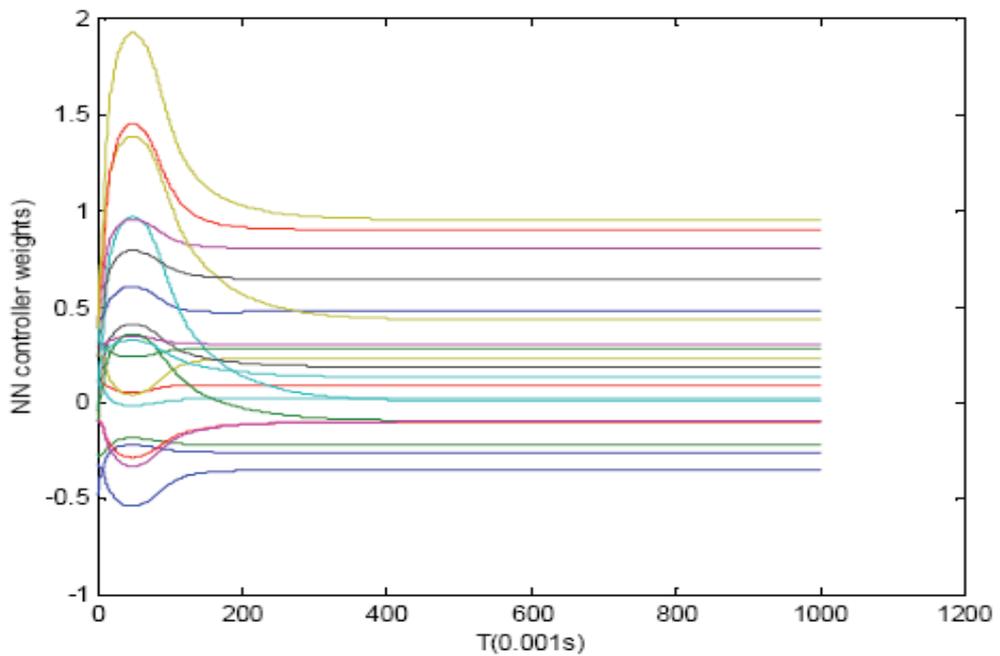


Fig. 12. All connective weights are convergent before 0.4s

The on-line trained neural controller using sign function approximation of Jacobian is also applied to this speed control system. The simulation results shown in Fig. 13, Fig. 14, and Fig. 15, which reveal that the on-line trained method takes more time for convergence.

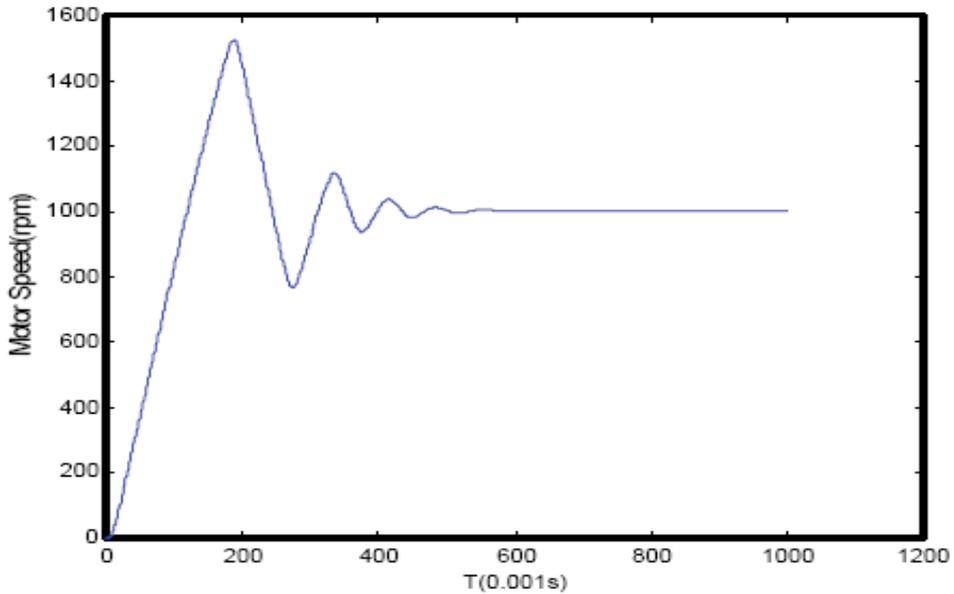


Fig. 13. Speed response for DC servo motor

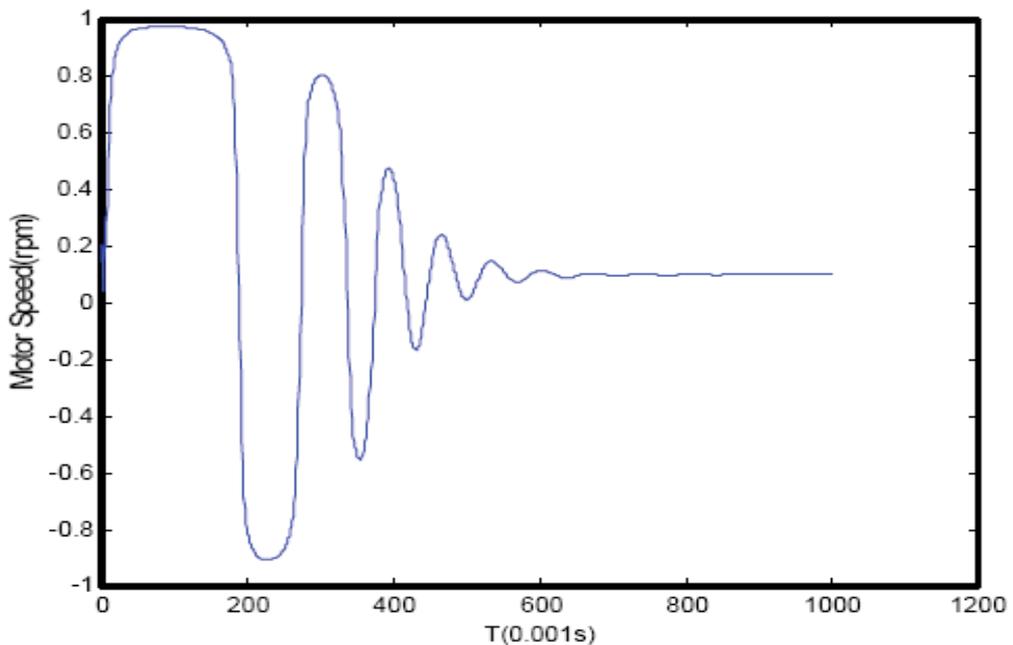


Fig. 14. The time response for control input

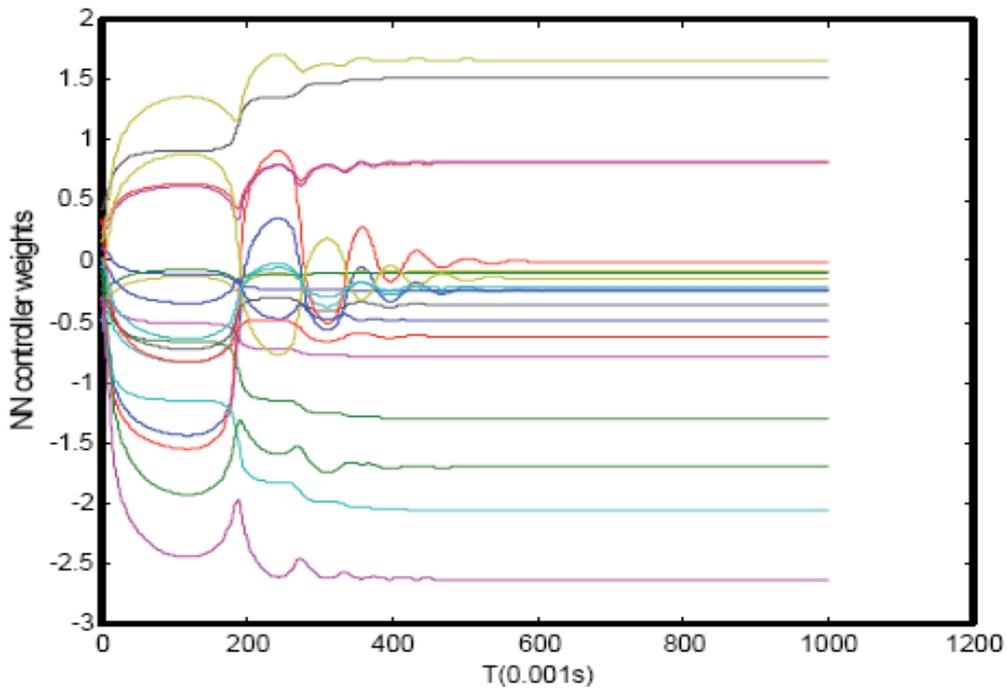


Fig. 15. All connective weights are convergent before 0.6s

The direct neural controllers using & adaptation law can provide better performance than using on-line trained method. The & adaptation law uses the error differential to increase the damping of the error convergent process, and improve the stability and convergent speed for weight update algorithm.

3. THE direct neural control applied to speed regulation for DC servo motor

The modern precise DC servo systems need to overcome the unknown nonlinear friction, parameters variations and torque load variations. It is reasonable to apply adaptive control to the DC servo system for speed control. But the conventional adaptive control techniques are usually based on the system model parameters. The unavailability of the accurate model parameters leads to a cumbersome design approach. The real-time implementation is difficult and sometimes not feasible because of using a large number of system parameters in these adaptive schemes. The proposed direct neural controllers can precisely regulate the speed for a DC servo motor, but don't have to the knowledge of system model parameters.

3.1 Description of the speed regulation system

The application of the direct neural controller for DC servo motor speed regulation is shown in Fig. 14, where v_r is the speed command and v is the actual output speed. The proposed neural network is treated as a speed regulator, which can keep the motor in constant speed against the disturbance.

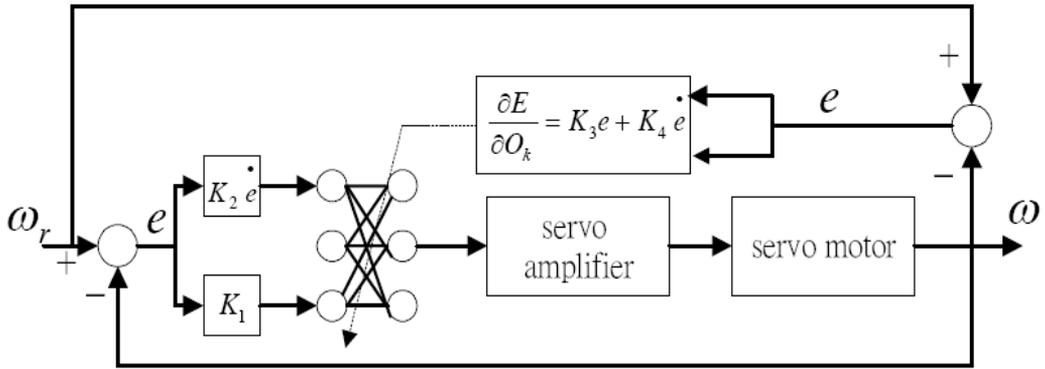


Fig. 14. The block diagram of speed control system

There are 5 hidden neurons in the proposed neural regulator. The proposed DNC is shown in Fig. 15 with a three layers neural network.

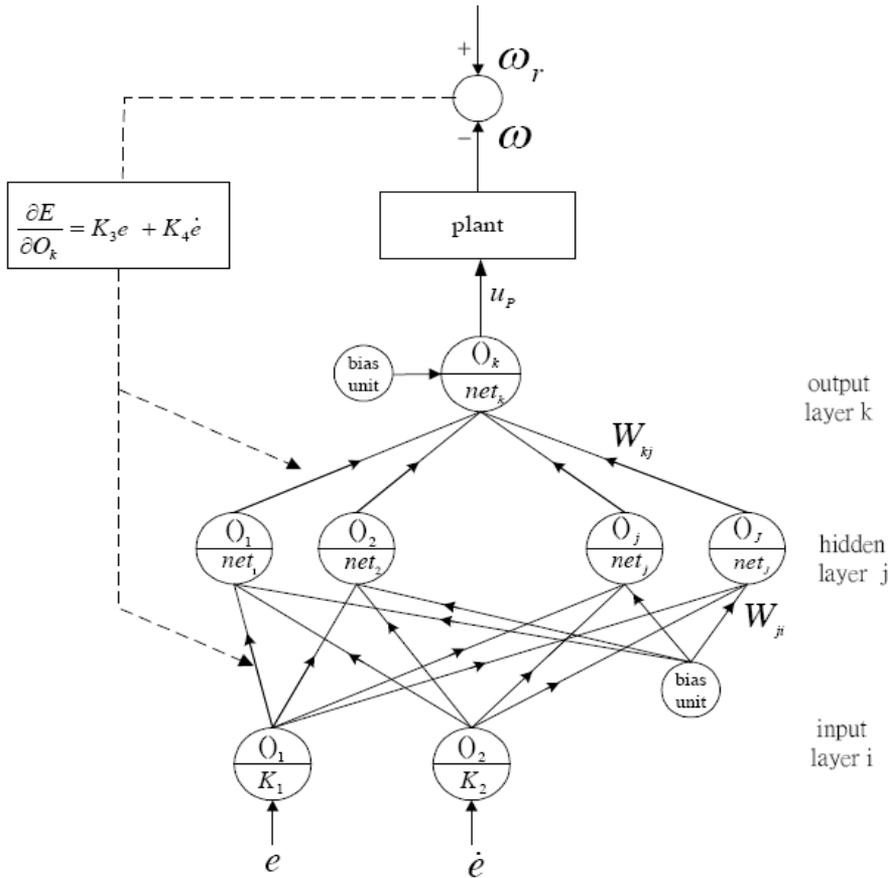


Fig. 15. The structure of proposed neural controller

The difference between command speed v_r and the actual output speed v is defined as error e . The error e and its differential \dot{e} are normalized between -1 and $+1$ in the input neurons before feeding to the hidden layer. In this study, the back propagation error term is approximated by the linear combination of error and error's differential. A tangent hyperbolic function is designed as the activation function of the nodes in the output and hidden layers. So that the net output in the output layer is bounded between -1 and $+1$, and converted into a bipolar analogous voltage signal through a D/A converter, then amplified by a servo-amplifier for enough current to drive the DC motor. A step speed command is assigned to be the reference command input in order to simulate the step speed response of a DC servo motor. The proposed three layers neural network, including the hidden layer (j), output layer (k) and input layer (i) as illustrated in Fig. 15. The input signals e and its differential \dot{e} are multiplied by the coefficients K_1 and K_2 , respectively, as the normalized signals O_i to hidden neuron. A tangent hyperbolic function is used as the activation function of the nodes in the hidden and output layers. The algorithms for weight update are described in previous section.

3.2 Dynamic Simulations

The block diagram of the DC servo motor speed control system with the proposed neural regulator is shown in Fig. 14, which consists of a 15W DC servo motor, an tachometer with a unit of $1/150.8$ V/rad/s, an 12 bits bipolar D/A converter with an output voltage range of $-5V$ to $+5V$ and a servo amplifier with voltage gain of 2.3. The parameters of DC servo motor are listed in Table 1.

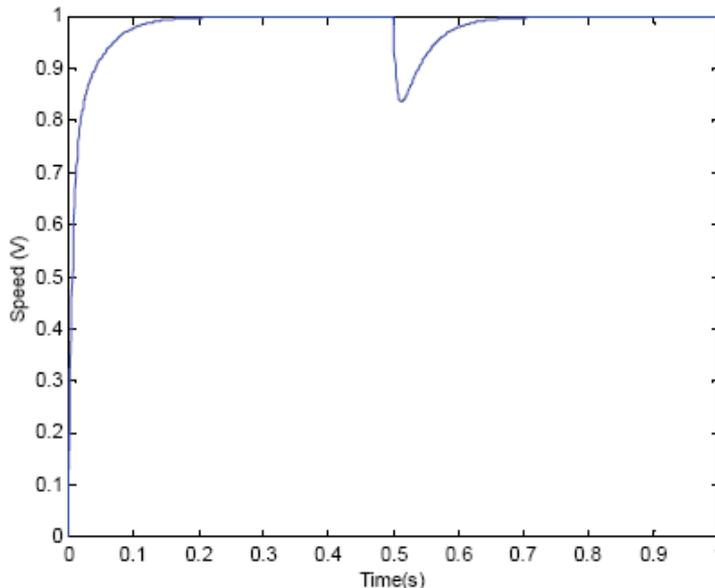
Motor resistance R_s	3.18Ω
Motor inductance L_a	$0.53mH$
Inertia of rotor J	$24.3 \times 10^{-7} kgm^2$
Torque constant K_T	$23mNm/A$
Back emf K_n	$0.00241V/rpm$

Table 1. The parameters of motor

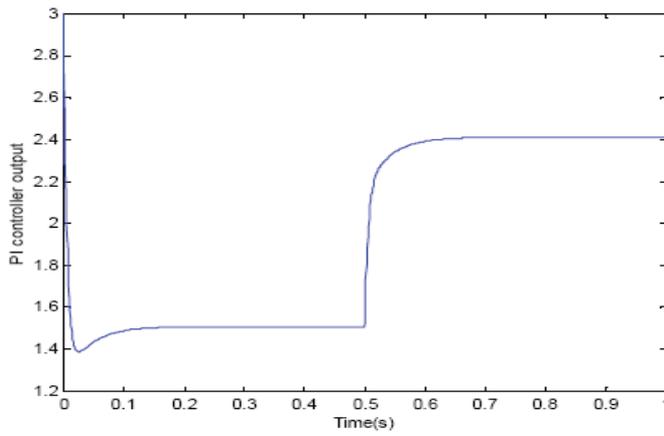
In the designed direct neural controller, the number of neurons is set to be 2, 5 and 1 for the input, hidden and output layers, respectively (see Fig.2). There is only one neuron in the output layer. The output signal of the direct neural controller will be between -1 and $+1$, which is converted into a bipolar analogous voltage signal by the D/A converter. The output of the D/A converter is between $+5V$ and $-5V$ corresponding to the output signal between $+1$ and -1 of the neural controller. It means the output of neural controller multiplied by a conversion gain of 5V. Then, the voltage signal is amplified by the servo amplifier to provide high current for driving the DC servo motor. The parameters K_1 and K_2 must be adjusted in order to normalize the input signals for the neural controller.

In this simulation, the parameters K_3 and K_4 can be determined, and $K_3 = K_1$ and $K_4 = K_2$ are assigned. In this simulations, a step signal of 1V corresponding to 150.8 rad/s is denoted as

the speed command, the sampling time is set to be 0.0001s, the learning rate η of the neural network is set to be 0.1 and the coefficient $\beta=0.5$ is assigned. Since the maximum error-voltage signal is 1V, the parameters K_1 and K_2 are assigned to be 0.6 and 0.01, respectively, in order to obtain an appropriate normalized input signals to the neural network. The parameters $K_3 = K_4 = 0.6$ and $K_5 = K_6 = 0.01$ are assigned for better convergent speed of the neural network. And a conventional PI controller with well tuning parameters applied to this speed regulate system is also simulated. Assumes a disturbance torque load of 0.015 Nm applies to this control system at $t=0.5$ s. The simulation results are shown in Fig. 16 and Fig. 17 4, where Fig. 16 (a) represents the speed response of the DC motor with PI controller, Fig. 16 (b) represents the output signal of the PI controller; Fig. 17 (a) represents the speed response of the DC motor with neural controller. Fig. 17 (b) represents the output signal of the neural controller. It exhibits a steady state error in the speed response is eliminated by the proposed neural regulator, which keeps appropriate voltage output as the inputs near 0. Fig. 17 (c) shows the convergent time of the connective weights is smaller than 100ms, and the speed response of the DC motor is stable. Consequently, the proposed neural speed regulator enhances the adaptability in speed control system. In addition, an extra attention should be taken on the disturbing torque load. The conventional PI controller does not have fast performance of speed regulation as the proposed neural speed regulator. The output of PI controller will saturate, if its performances are increased to near the neural regulator. Fig.3 (b) shows the maximum output of PI controller is close to 3V. Fig.4 (b) shows the maximum output of neural regulator is only 2.41V, and the speed regulation performance of neural regulator is better than that of the PI controller. The simulation results exhibit the neural regulator is available for the high-precision speed control systems. If the speed command is increased and over 1.66V, the parameters K_1 and K_2 need to be adjusted, the parameters K_3 and K_4 also need to be adjusted to appropriate values. Basically increasing the values of K_2 and K_4 will increase the damping effect of control system.

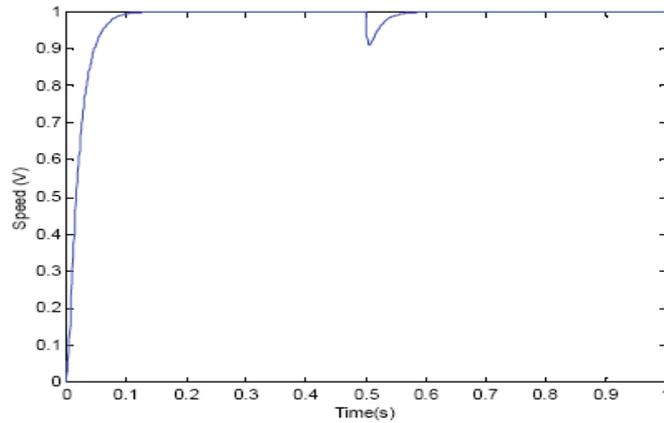


(a) Speed response

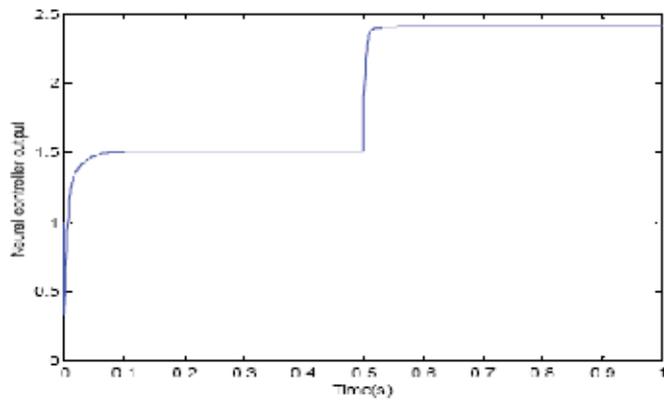


(b) Output of PI controller

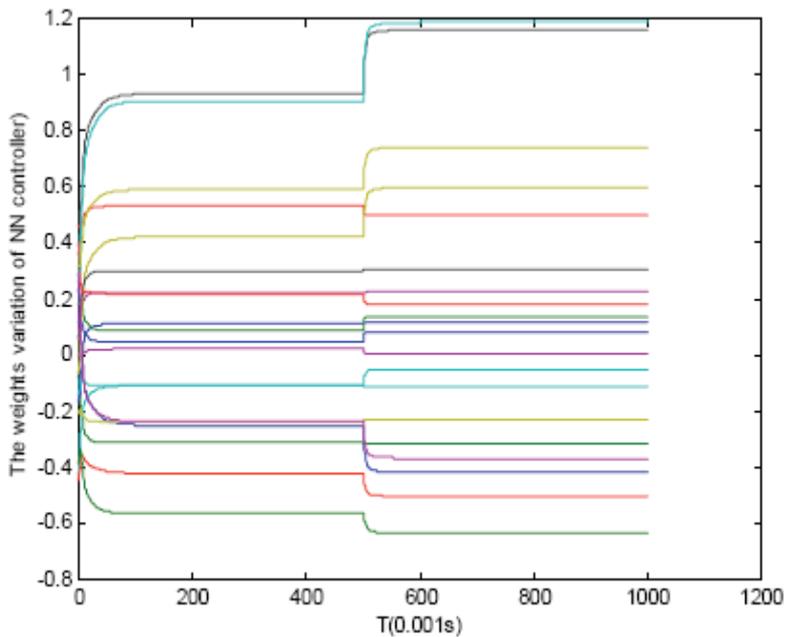
Fig. 16. Simulation results for speed regulation of DC servo motor with PI controller



(a) Speed response



(b) Output of neural regulator

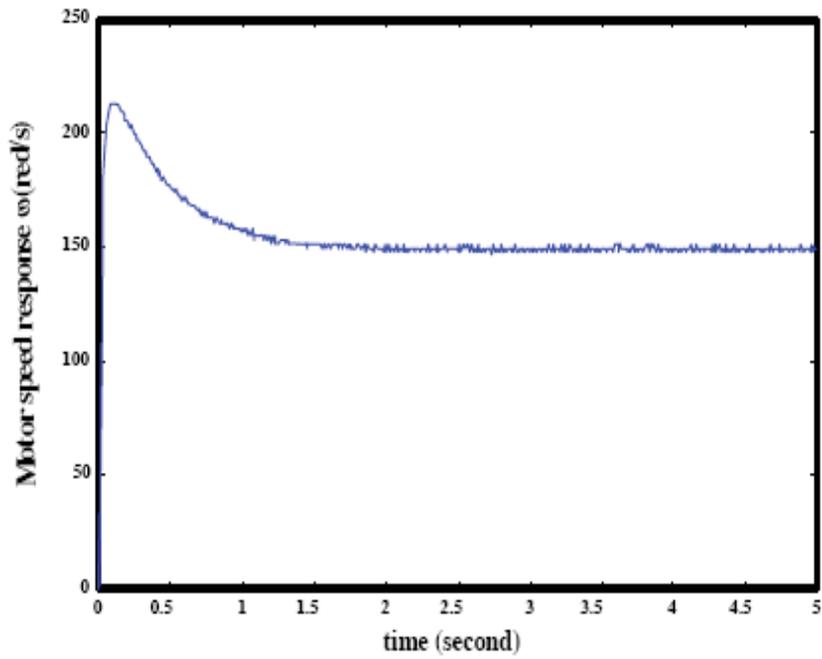


(c) The time responses for connective weights

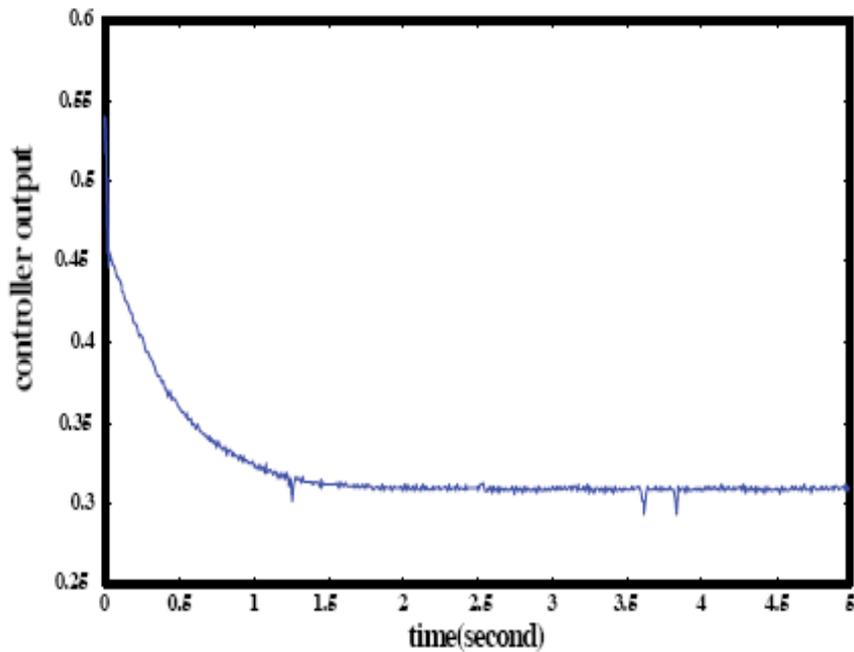
Fig. 17. Simulation results for speed regulator of DC servo motor with neural regulator

3.3 Experimental Results

The experimental apparatus consist of a 15W DC servo motor whose parameters shown in Table 1 are the same as that of simulation, an encoder with a unit of 0.01256 rad/pulse, an 12bits bipolar D/A converter with a voltage range of A5.13V to +5.13V and a servo amplifier with voltage gain of 2.3. In the designed direct neural controller, the number of neurons is set to be 2, 5 and 1 for the input, hidden and output layers, respectively. There is only one neuron in the output layer. The output signal of the direct neural controller will be between A1 and +1, which is converted into a bipolar analogous voltage signal by the D/A converter. The output of the D/A converter is between +4.847V and -4.847V corresponding to the output signal between +1 and -1 of the neural controller. Then, the voltage signal is amplified by the servo-amplifier to provide high current for driving the DC servo motor. Furthermore, the parameters K_1 and K_2 must be adjusted in order to normalize the input signals of the neural regulator. In this experiment, the parameters K_3 and K_4 depend on K_1 and K_2 , and $K_3 = K_1$ and $K_4 = K_2$ is assigned. A step signal of 120 pulses/0.01s (12000 pulses/s) corresponding to 150.8 rad/s is denoted as the speed command. The learning rate $\eta=0.3$, the coefficient $\beta=0.5$ and the sampling time of 0.01s are assigned. The parameter $K_1=0.003$ and $K_2=0.00003$ are defined to normalize the input signals of the neural controller. The relations $K_1 = K_3$ and $K_2 = K_4$ are applied in our experiments. The results are shown in Fig. 18. It shows that the speed response of a DC motor is stable and accurate but with overshoot. The experiment result with $K_2 = K_4 = 0.00004$ and $\eta=0.5$ is shown in Fig. 19. It shows that parameters K_2 and K_4 exhibit a damping effect evidenced by an overshoot in the speed response of a DC servo motor as $K_2 = K_4 = 0.0003$, and the overshoot decreased as $K_2 = K_4 = 0.00004$. The settling time is decreased as $K_2 = K_4 = 0.0004$.

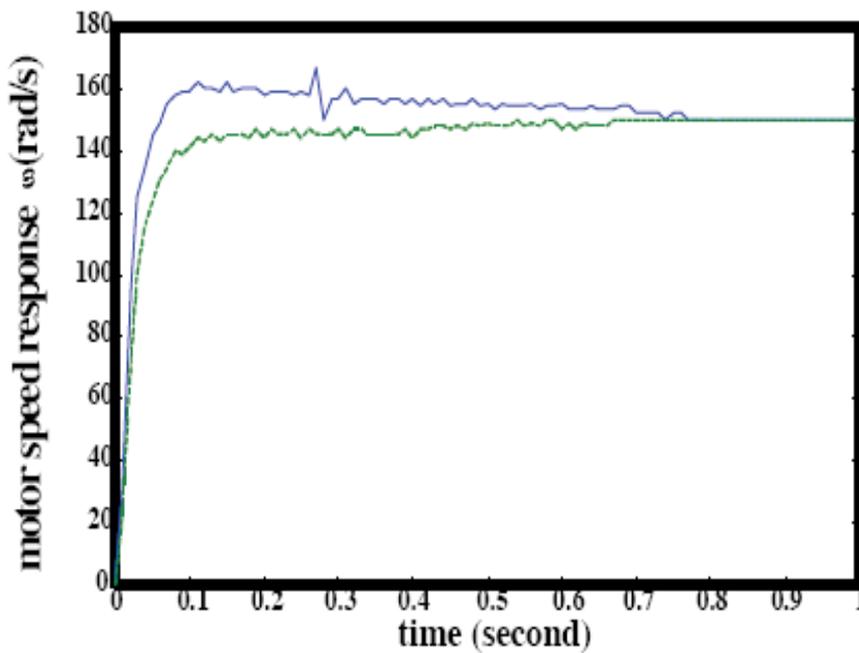


(a) Speed response of DC servo motor

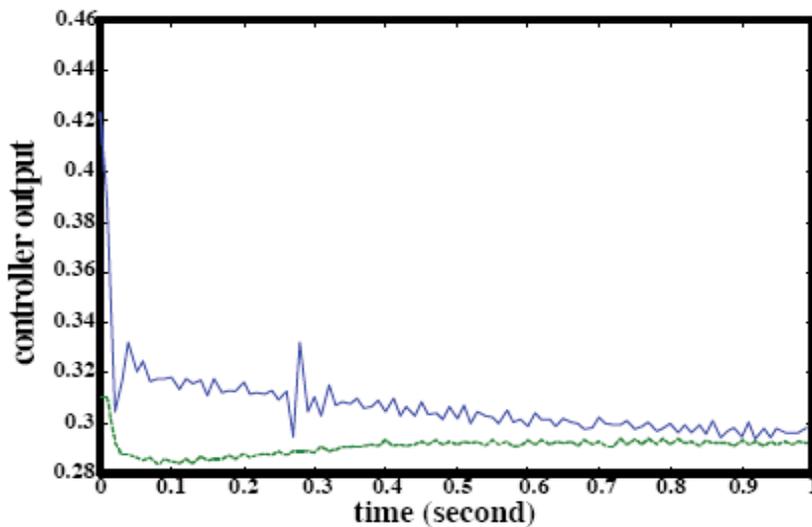


(b) The output of neural controller

Fig. 18. Experiment results (Sampling time=0.01s, $\eta=0.3$, $K_1 = K_3 = 0.003$, $K_2 = K_4 = 0.00003$)



(a) Speed response of DC servo motor

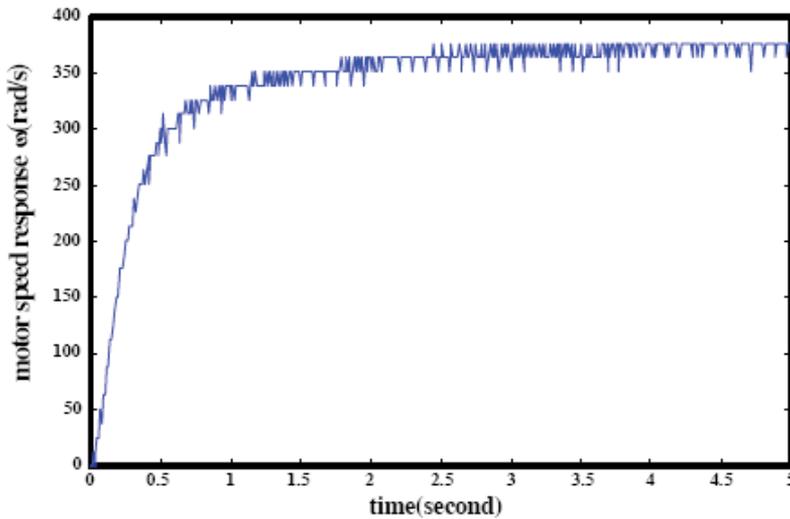


(b) The output of neural controller

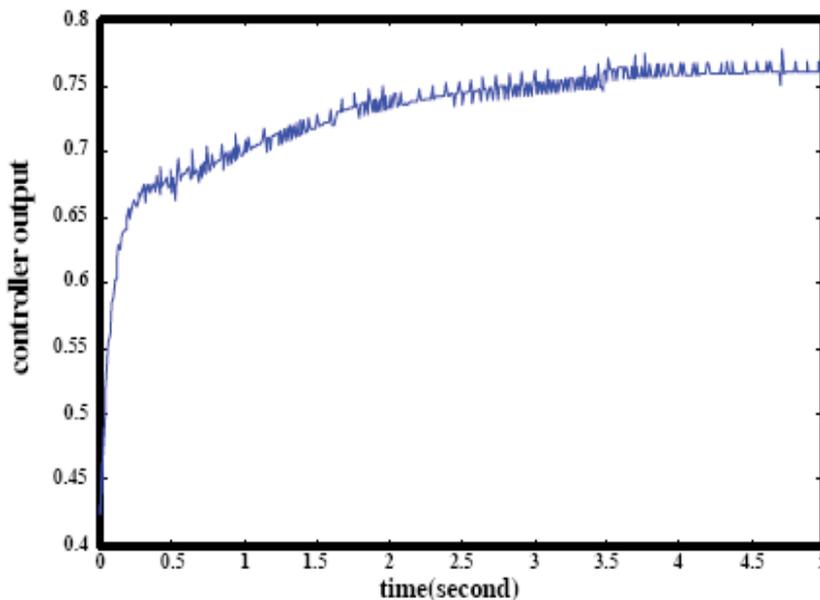
Fig. 19. Experiment results (Sampling time=0.01s, $K_1 = K_3 = 0.003, \eta = 0.5, K_2 = K_4 = 0.00003$: ----, $K_2 = K_4 = 0.00004$: -----)

It is a better way for neural controller that the sampling time is as small as possible, then choosing a correspondent small learning rate. It means the connective weights can have more update times within the same time interval. The smaller sampling interval leads to

more weight updates per second. It is helpful for convergence of on line learning. So that, a smaller sampling interval of 0.001s and the speed command of 30 pulses/ms (30,000 pulses/s) corresponding to 377rad/s are applied to this experiment, it means the connective weights can be updated 1000 times per second. The parameters $K_1 = K_3 = 0.003$ and $K_2 = K_4 = 0.00003$ are assigned for this experiment. Both of the learning rate of 0.3 and 0.5 are assigned, and the corresponding experiment results are shown in Fig. 20 and Fig. 21 respectively.

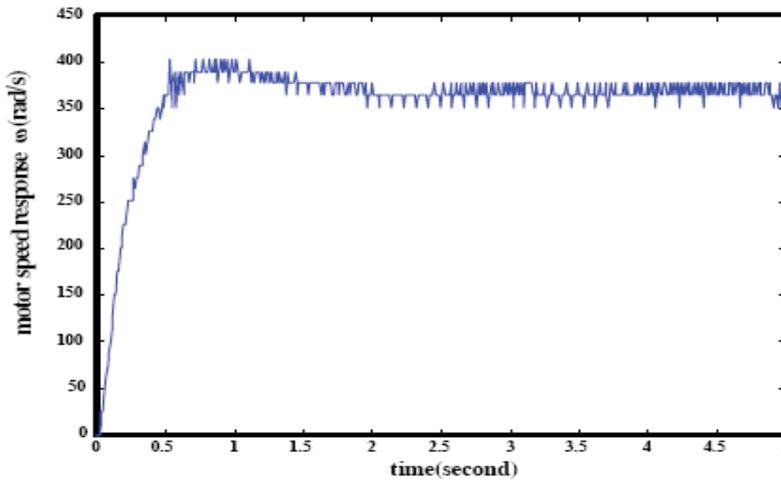


(a) Speed response of DC servo motor

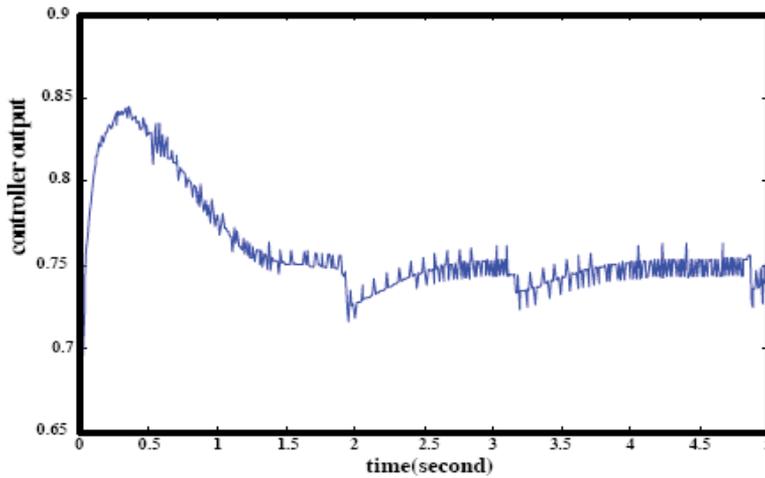


(b) The output of neural controller

Fig. 20. Experiment results (Sampling time=0.001s, $\eta=0.3$, $K_1 = K_3 = 0.03$, $K_2 = K_4 = 0.00003$)



(a) Speed response of DC servo motor



(b) The output of neural controller

Fig. 21. Experiment results (Sampling time=0.001s, $\eta=0.5$, $K_1 = K_3 = 0.03$, $K_2 = K_4 = 0.00003$)

Fig. 20 and Fig. 21 show the smaller sampling interval make the pulse number of one sampling interval become smaller, so that the speed error to speed command ratio will become larger. The speed error is between -1 and +1 pulse per sampling interval.

In Fig. 21, the speed response is still stable with $\eta = 0.5$, but more overshoot can be investigated; owing to the fact that more learning rate induces more neural controller output and get more overshoot. It can be investigated that the sampling time needs to be smaller, then choosing a correspondent small learning rate. It is proven that the speed response of a DC servo motor with the proposed direct neural controller is stable and accurate. The simulation and experimental results show the speed error comes from speed sensor characteristics, the measurement error is between -1 and +1 pulse per sampling interval. If the resolution of encoder is improved, the accuracy of the control system will be increased.

The speed error is in the interval of 1 pulses/0.01s as the sampling time of 0.01s, but it is in the interval of 1 pulses/0.001s as the sampling time of 0.001s. The step speed command is assigned as 120 pulses/0.01s (150.72rad/s) with the sampling interval of 0.01s, and the step speed command needs to be increased to 30pulses/ms (377rad/s) to keep the accuracy of the speed measurement. Furthermore, we have to notice the normalization of the input signals. From the experimental results, the input signals need to be normalized between +1 and A1. The learning rate should be determined properly depends on the sampling interval, the smaller sampling interval can match the smaller learn rate, and increase the stability of servo control system.

4. The Direct Neural Control Applied to Hydraulic Servo Control Systems

The electro-hydraulic servo systems are used in aircraft, industrial and robotic mechanisms. They are always used for servomechanism to transmit large specific powers with low control current and high precision. The electro-hydraulic servo system (EHSS) consists of hydraulic supply units, actuators and an electro-hydraulic servo valve (EHSV) with its servo driver. The EHSS is inherently nonlinear, time variant and usually operated with load disturbance. It is difficult to determine the parameters of dynamic model for an EHSS. Furthermore, the parameters are varied with temperature, external load and properties of oil etc. The modern precise hydraulic servo systems need to overcome the unknown nonlinear friction, parameters variations and load variations. It is reasonable for the EHSS to use a neural network based adaptive control to enhance the adaptability and achieve the specified performance.

4.1 Description of the electro-hydraulic servo control system

The EHSS is shown in Fig. 22 consists of hydraulic supply units, actuators and an electro-hydraulic servo valve (EHSV) with its servo driver. The EHSV is a two-stage electro hydraulic servo valve with force feedback. The actuators are hydraulic cylinders with double rods.

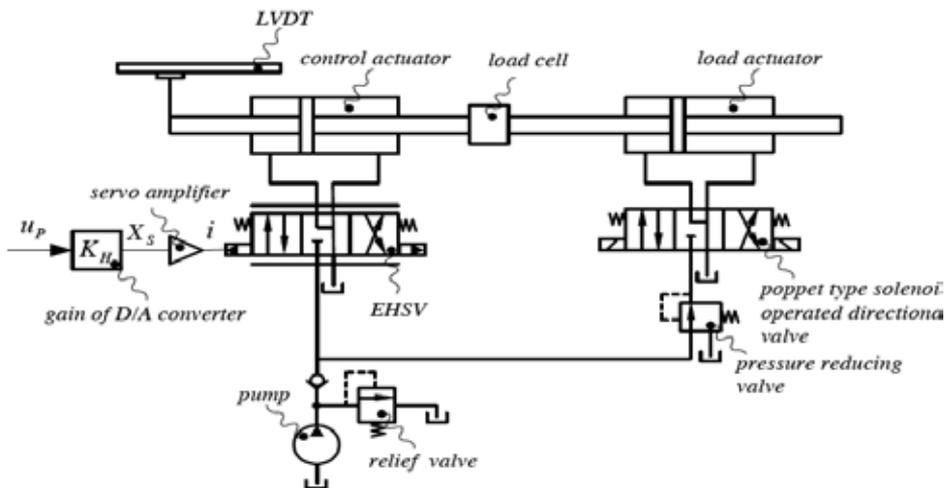


Fig. 22. The hydraulic circuit of EHSS

The application of the direct neural controller for EHSS is shown in Fig. 23, where y_r is the position command and y_p is the actual position response.

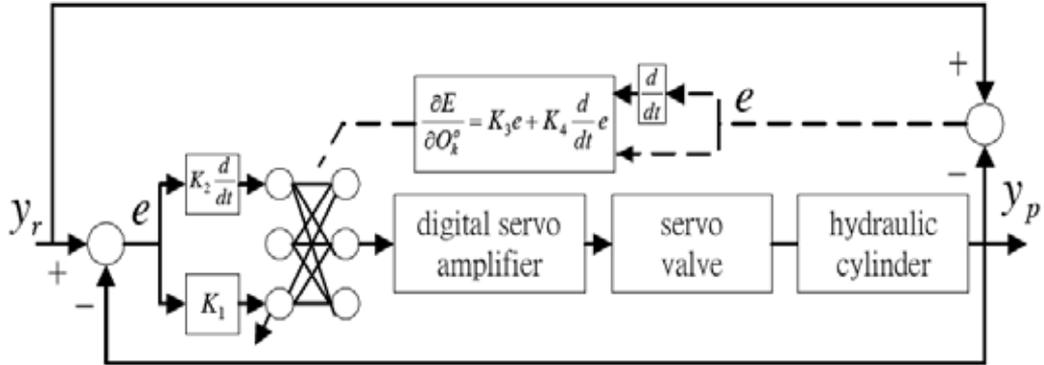


Fig. 23. The block diagram of EHSS control system

A. The simplified servo valve model

The EHSV is a two-stage electro hydraulic servo valve with force feedback. The dynamic of EHSV consists of inductance dynamic, torque motor dynamic and spool dynamic. The inductance and torque motor dynamics are much faster than spool dynamic, it means the major dynamic of EHSV determined by spool dynamic, so that the dynamic model of servo valve can be expressed as:

$$\frac{\Delta x_v}{\Delta e} = \frac{K_v}{1 + S/K_v} \tag{24}$$

Δx_v : The displacement of spool

Δe : The input voltage

B. The dynamic model of hydraulic cylinder

The EHSV is 4 ports with critical center, and used to drive the double rods hydraulic cylinder. The leakages of oil seals are omitted and the valve control cylinder dynamic model can be expressed as [8]:

$$X_p = \frac{\frac{kq}{A_p} x_v - \frac{k_c}{A_p^2} (1 + \frac{V_t}{4\beta_p k_c} s) F_L}{s(\frac{V_t M_t}{4\beta_p A_p^2} s^2 + (\frac{k_r M_t}{A_p^2} + \frac{B_p V_t}{4\beta_p A_p^2})s + (1 + \frac{B_p k_c}{A_p^2}))} \tag{25}$$

x_v : The displacement of spool

F_L : The load force

X_p : The piston displacement

C. Direct Neural Control System

There are 5 hidden neurons in the proposed neural controller. The proposed DNC is shown in Fig. 24 with a three layers neural network.

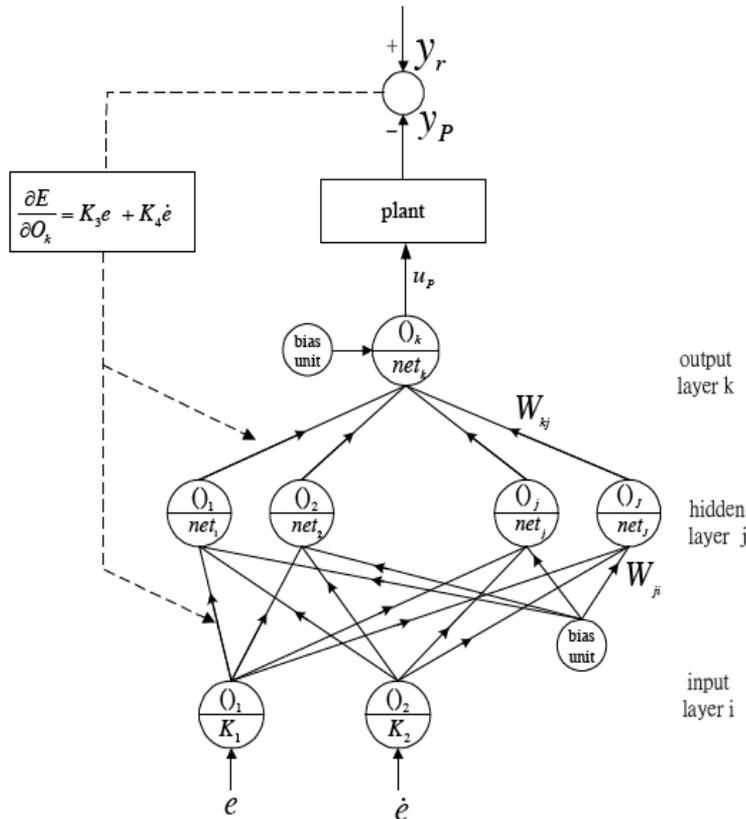


Fig. 24. The structure of proposed neural controller

The difference between command y_r and the actual output position response y_p is defined as error e . The error e and its differential \dot{e} are normalized between -1 and +1 in the input neurons before feeding to the hidden layer. In this study, the back propagation error term is approximated by the linear combination of error and error's differential. A tangent hyperbolic function is designed as the activation function of the nodes in the output and hidden layers. So that the net output in the output layer is bounded between -1 and +1, and converted into a bipolar analogous voltage signal through a D/A converter, then amplified by a servo-amplifier for enough current to drive the EHSV. A square command is assigned as the reference command in order to simulate the position response of the EHSS. The proposed three layers neural network, including the hidden layer (j), output layer (k) and input layer (i) as illustrated in Fig. 24. The input signals e and \dot{e} are normalized between -1 and +1, and defined as signals O_i feed to hidden neurons. A tangent hyperbolic function is used as the activation function of the nodes in the hidden and output layers. The net input to node j in the hidden layer is

$$net_j = \sum (W_{ij} \cdot O_i) + \theta_j \quad i=1,2,\dots,I, \quad j=1,2,\dots,J \quad (26)$$

the output of node j is

$$O_j = f(net_j) = \tanh(\beta \cdot net_j) \quad (27)$$

where $\beta > 0$, the net input to node k in the output layer is

$$net_k = \sum (W_{kj} \cdot O_j) + \theta_k \quad j=1,2,\dots,J, \quad k=1,2,\dots,K \quad (28)$$

the output of node k is

$$O_k = f(net_k) = \tanh(\beta \cdot net_k) \quad (29)$$

The output O_k of node k in the output layer is treated as the control input u_P of the system for a single-input and single-output system. As expressed equations, W_{ji} represent the connective weights between the input and hidden layers and W_{kj} represent the connective weights between the hidden and output layers θ_j and θ_k denote the bias of the hidden and output layers, respectively. The error energy function at the Nth sampling time is defined as

$$E_N = \frac{1}{2} (y_{rN} - y_{PN})^2 = \frac{1}{2} e_N^2 \quad (30)$$

where y_{rN} , y_{PN} and e_N denote the the reference command, the output of the plant and the error term at the Nth sampling time, respectively. The weights matrix is then updated during the time interval from N to N+1.

$$\Delta W_N = W_{N+1} - W_N = -\eta \frac{\partial E_N}{\partial W_N} + \alpha \cdot \Delta W_{N-1} \quad (31)$$

where η is denoted as learning rate and α is the momentum parameter. The gradient of E_N with respect to the weights W_{kj} is determined by

$$\frac{\partial E_N}{\partial W_N} = \frac{\partial E_N}{\partial net_k} \frac{\partial net_k}{\partial W_N} = \delta_k O_j \quad (32)$$

and δ_k is defined as

$$\begin{aligned}\delta_k &= \frac{\partial E_N}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial X_p} \frac{\partial X_p}{\partial u_p} \frac{\partial u_p}{\partial Q_k} \frac{\partial Q_k}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial Q_k} \frac{\partial Q_k}{\partial net_k} \\ &= \sum_n \frac{\partial E_N}{\partial Q_k} \beta(1 - O_k^2) \quad n=1,2,\dots,K\end{aligned}\quad (33)$$

where $\partial X_p / \partial u_p$ is difficult to be evaluated. The EHSS is a single-input and single-output control system (i.e., $n = 1$), in this study, the sensitivity of E_N with respect to the network output O_k is approximated by a linear combination of the error and its differential shown as:

$$\frac{\partial E_N}{\partial O_k} = K_3 e + K_4 \frac{de}{dt} \quad (34)$$

where K_3 and K_4 are positive constants. Similarly, the gradient of E_N with respect to the weights, W_i , is determined by

$$\frac{\partial E_N}{\partial W_{ji}} = \frac{\partial E_N}{\partial net_j} \frac{\partial net_j}{\partial W_{ji}} = \delta_j O_i \quad (35)$$

where

$$\begin{aligned}\delta_j &= \frac{\partial E_N}{\partial net_j} = \sum_m \frac{\partial E_N}{\partial net_k} \frac{\partial net_k}{\partial O_m} \frac{\partial O_m}{\partial net_j} \\ &= \sum_m \delta_k W_{km} \beta(1 - O_j^2) \quad m = 1, 2, \dots, J\end{aligned}\quad (36)$$

The weight-change equations on the output layer and the hidden layer are

$$\begin{aligned}\Delta W_{kj,N} &= -\eta \frac{\partial E_N}{\partial W_{kj,N}} + \alpha \cdot \Delta W_{kj,N-1} \\ &= -\eta \delta_k O_j + \alpha \cdot \Delta W_{kj,N-1}\end{aligned}\quad (37)$$

$$\begin{aligned}\Delta W_{ji,N} &= -\eta \frac{\partial E_N}{\partial W_{ji,N}} + \alpha \cdot \Delta W_{ji,N-1} \\ &= -\eta \delta_j O_i + \alpha \cdot \Delta W_{ji,N-1}\end{aligned}\quad (38)$$

where η is denoted as learning rate and α is the momentum parameter δ_j and δ_k can be evaluated from Eq.(34) and (31), The weights matrix are updated during the time interval from N to $N+1$:

$$W_{kj,N+1} = W_{kj,N} + \Delta W_{kj,N} \quad (39)$$

$$W_{ji,N+1} = W_{ji,N} + \Delta W_{ji,N} \quad (40)$$

4.2 Numerical Simulation

An EHSS shown as Fig.1 with a hydraulic double rod cylinder controlled by an EHSV is simulated. A LVDT of 1 V/m measured the position response of EHSS. The numerical simulations assume the supplied pressure $P_s = 70 \text{ Kg}_f/\text{cm}^2$, the servo amplifier voltage gain of 5, the maximum output voltage of 5V, servo valve coil resistance of 250 ohms, the current to voltage gain of servo valve coil of 4 mA/V (250 ohms load resistance), servo valve settling time $\approx 20\text{ms}$, the servo valve provides maximum output flow rate = 19.25 l/min under coil current of 20mA and ΔP of 70Kg_f/cm² condition. The spool displacement can be expressed by percentage (%), and then the model of servo valve can be built as

$$\frac{x_v(100\%)}{i \text{ (mA)}} = \frac{0.05}{S/200 + 1} \quad (41)$$

or

$$\frac{x_v(100\%)}{v \text{ (V)}} = \frac{0.2}{S/200 + 1} \quad (42)$$

The cylinder diameter =40mm, rod diameter=20mm, stroke=200mm, and the parameters of the EHSS listed as following:

$$A_p = 9.4248\text{cm}^2 = 0.00094248\text{m}^2$$

$$V_c = 188.5\text{cm}^3 = 0.0001885\text{m}^3$$

$$B_p = 40 \text{ N} \cdot \text{s}/\text{m}$$

$$k_c = 3.727(10^{-5}) \text{ m}^3/\text{MPa} \cdot \text{s}$$

$$M_s = 1\text{kgm}$$

$$k = 0 \text{ N}/\text{m}$$

$$\beta_s = 1000\text{MPa}$$

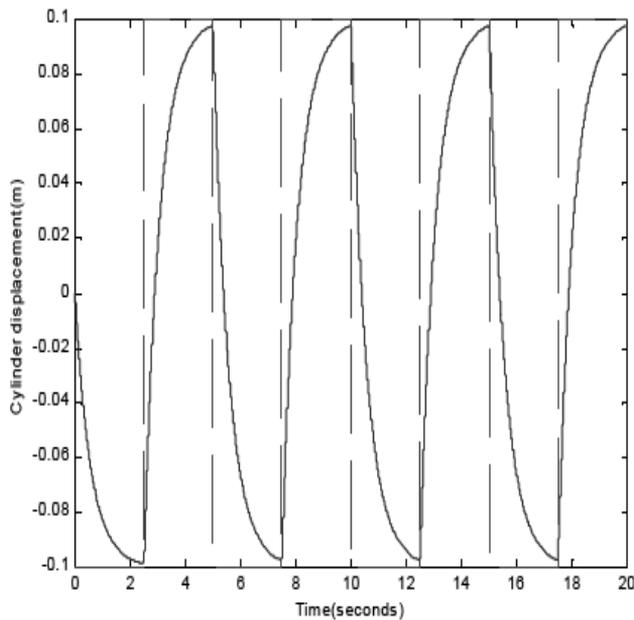
$$kq = 19.25 \text{ l}/\text{min} \text{ (at } \Delta P = 70.3 \text{ Kg}_f/\text{cm}^2 \text{)}$$

$$= 320.833\text{m}^3/\text{s} = 0.000320833 \text{ m}^3/\text{s}$$

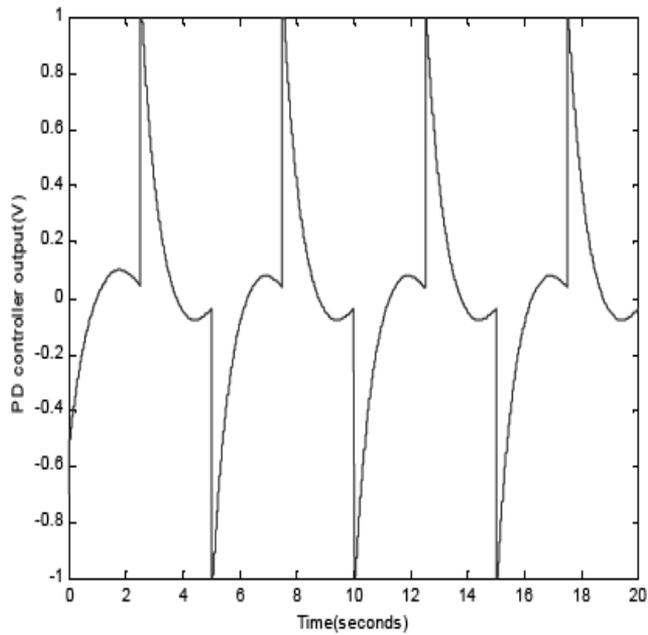
According to Eq(25), the no load transfer function is shown as

$$\frac{y_p}{x_i} = \frac{340414}{s(0.053s^2 + 44.122s + 1001678)} \tag{43}$$

The direct neural controller is applied to control the EHSS shown as Fig. 24, and the time responses for piston position are simulated. A tangent hyperbolic function is used as the activation function, so that the neural network controller output is between -1 . This is converted to be analog voltage between -) Volt by a D/A converter and amplified in current by a servo amplifier to drive the EHSV. The constants K_3 and K_4 are defined to be the parameters for the linear combination of error and its differential, which is used to approximate the BPE for weights update. A conventional PD controller with well-tuned parameters is also applied to the simulation stage as a comparison performance. The square signal with a period of 5 sec and amplitude of 0.1m is used as the command input. The simulation results for PD control is shown in Fig. 25 and for DNC is shown in Fig. 26. Fig. 26 reveals that the EHSS with DNC track the square command with sufficient convergent speed, and the tracking performance will become better and better by on-line trained. Fig. 27 shows the time response of piston displacement with 1200N force disturbance. Fig. 27 (a) shows the EHSS with PD controller is induced obvious overshoot by the external force disturbance, and Fig. 27 (b) shows the EHSS with the DNC can against the force disturbance with few overshoot. From the simulation results, we can conclude that the proposed DNC is available for position control of EHSS, and has favorable tracking characteristics by on-line trained with sufficient convergent speed.

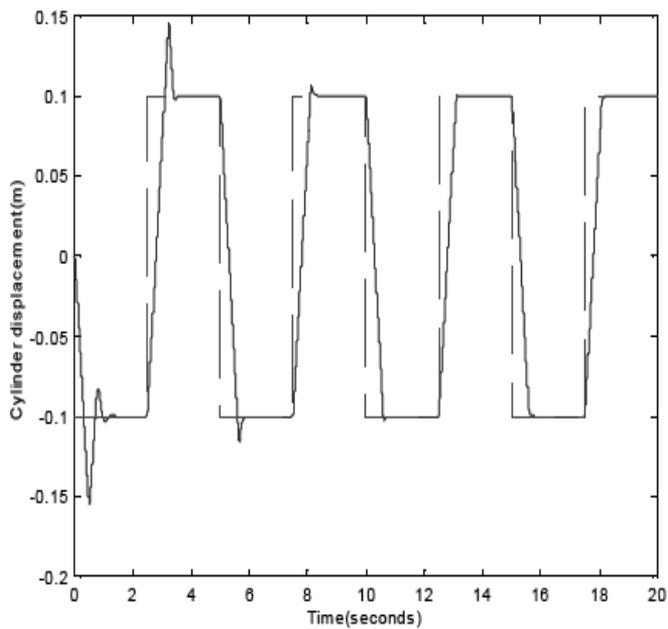


(a) Time response for piston displacement

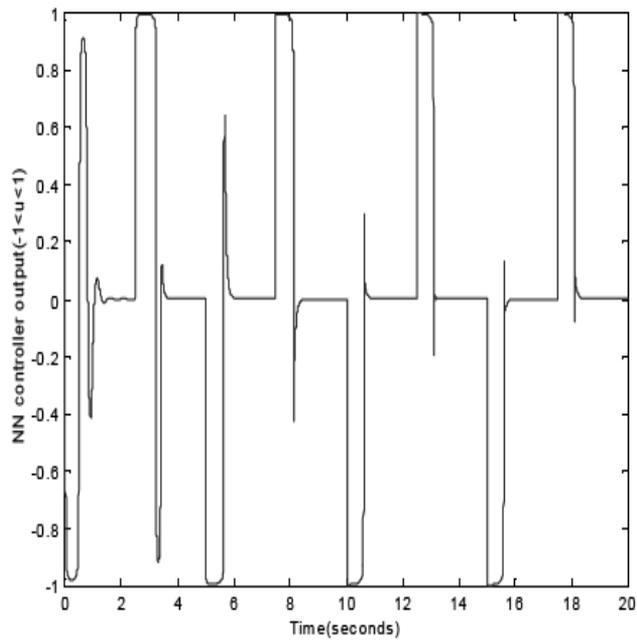


(b) Controller output

Fig. 25. The simulation results for EHSS with PD controller ($K_p=7$, $K_d=1$, Amplitude=0.1m and period=5 sec)

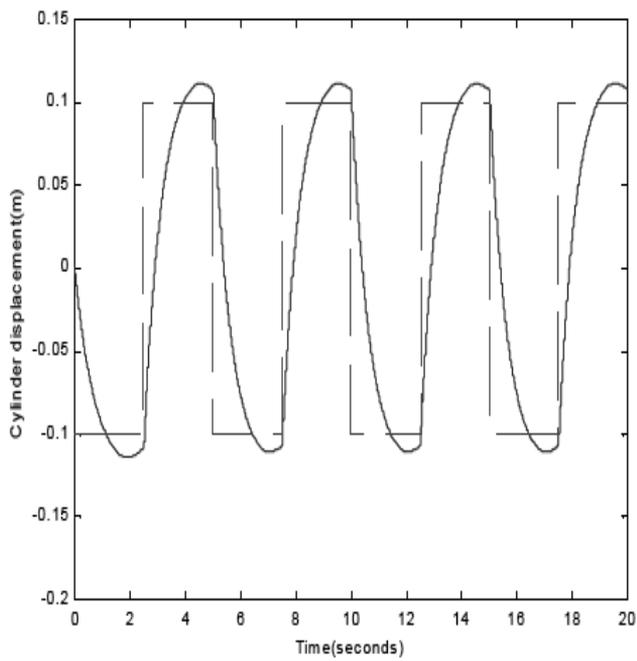


(a) Time response for piston displacement

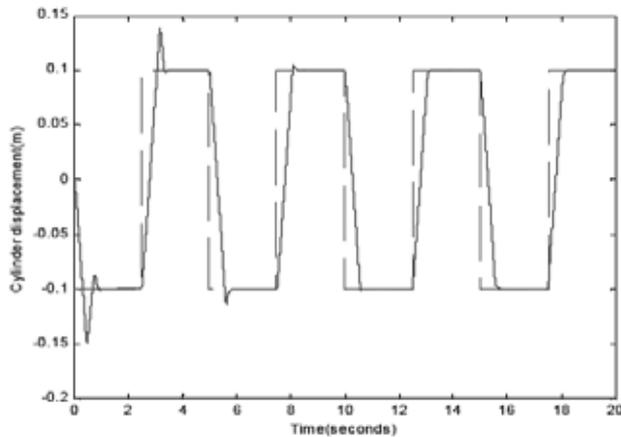


(b) Controller output

Fig. 26. The simulation results for EHSS with DNC (Amplitude=0.1m and period=5 sec)



(a) EHSS with PD controller

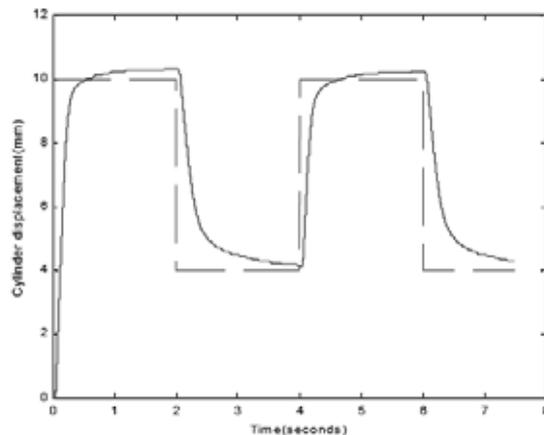


(b) EHSS with DNC

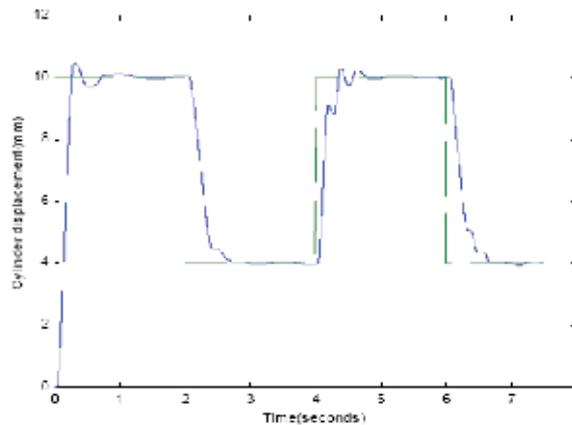
Fig. 27. Simulation results of position response with 1200N force disturbance

4.3. Experiment

The EHSS shown in Fig. 22 is established for our experiment. A hydraulic cylinder with 200mm stroke, 20mm rod diameter and a 40mm cylinder diameter is used as the system actuator. The Atchley JET-PIPE-206 servo valve is applied to control the piston position of hydraulic cylinder. The output range of the neural controller is between -1 , and converted to be the analog voltage between -5 Volt by a 12 bits bipolar DA /AD servo control interface, It is amplified in current by a servo amplifier to drive the EHSV. A crystal oscillation interrupt control interface provides an accurate 0.001 sec sample rate for real time control. A square signal with amplitude of 10mm and period of 4 sec is used as reference input. Fig. 28 shows the EHSS disturbed by external load force, which is induced by load actuator with operation pressure of 9 kg / cm^2 . Fig. 28 (a) shows the EHSS with PD controller is induced obvious overshoot by the external force disturbance, and Fig. 28 (b) shows the EHSS with the DNC can against the force disturbance with few overshoot. The experiment results show the proposed DNC is available for position control of EHSS.



(a) EHSS with PD controller



(b) EHSS with DNC

Fig. 28. Experiment results of position response with the load actuator pressure of 9 kg/cm^2

The proposed DNC is applied to control the piston position of a hydraulic cylinder in an EHSS., and the comparison of time responses for the PD control system is analyzed by simulation and experiment. The results show that the proposed DNC has favorable characteristic, even under external force load condition.

5. Conclusion

The conventional direct neural controller with simple structure can be implemented easily and save more CPU time. But the Jacobian of plant is always not easily available. The DNC using sign function for approximation of Jacobian is not sufficient to apply to servo control system. The & adaptation law can increase the convergent speed effutely, but the appropriate parameters always depend on try and error. It is not easy to evaluated the appropriate parameters. The proposed self tuning type adaptive control can easily determined the appropriate parameters. The DNC with the well-trained parameters will enhance adaptability and improve the performance of the nonlinear control system.

6 References

- D. Psaltis, A Sideris, and A. A. Yamamura (1988). A Multilayered Neural Network Controller, *IEEE Control System Magazine*, v.8, pp. 17-21.
- Y. Zhang, P. Sen, and G. E. Hearn (1995). An On-line Trained Adaptive Neural Controller, *IEEE Control System Magazine*, v.15, pp. 67-75.
- S. Weerasooriya and M. A. El-Sharkawi Hearn (1991). Identification and Control of a DC Motor Using Back-propagation Neural Networks, *IEEE Transactions on Energy Conversion*, v.6, pp. 663-669.
- A. Rubai and R. Kotaru (2000). Online Identification and Control of a DC Motor Using Learning Adaptation of Neural Networks, *IEEE Transactions on Industry Applications*, v.36, n.3.


```

        sys=mdlOutputs(t,x,u);
case {1,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 21;
sizes.NumOutputs = 21;
sizes.NumInputs = 5;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;
      rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;
      rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;
      rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;0.2];
%%%set the initial values for weights and states
%%%the initial values of weights randomly between -0.5 and +0.5
%%%the initial values of NN output assigned to be 0.2
str = [];
ts = [0 0];
function sys=mdlUpdate(t,x,u);
nv=0;
for j=1:5
    for i=1:3
        nv=nv+1;
        w1(j,i)=x(nv);
    end
end
k=1;
    for j=1:5
nv=nv+1;
w2(k,j)=x(nv);
    end
    for j=1:5
jv(j)=w1(j,:)*[u(1);u(2);u(3)]; %u(1)=K1*e ,u(2)=K2*de/dt
%u(3)=1 is bias unity
ipj(j)=tanh(0.5*jv(j)); %outputs of hidden layer
    end
kv(1)=w2(1,:)*ipj';
opk(1)=tanh(0.5*kv(1)); %output of output layer
    for j=1:5
dk=(u(4)+u(5))*0.5*(1-opk(1)*opk(1));
%%%delta adaptation law, dk means delta K,u(4)=K3*e ,u(5)=K4*de/dt
dw2(1,j)=0.1*dk*ipj(j); %dw2 is weight update quantity for W2
    end
    for j=1:5
sm=0;
sm=sm+dk*w2(1,j);
sm=sm*0.5*(1-ipj(j)*ipj(j));
dj(j)=sm; %back propogation, dj means delta J
    end
    for j=1:5
        for i=1:3

```

```
dw1(j,i)=0.1*dj(j)*u(i); %dw1 is weight update quantity for W1
end
end
for j=1:5
w2(1,j)=w2(1,j)+dw2(1,j); %weight update
for i=1:3
w1(j,i)=w1(j,i)+dw1(j,i); %weight update
end
end
nv=0;
for j=1:5
for i=1:3
nv=nv+1;
x(nv)=w1(j,i); %assign w1(1)-w1(15) to x(1)-x(15)
end
end
k=1;
for j=1:5
nv=nv+1;
x(nv)=w2(k,j); %assign w2(1)-w2(5) to x(16)-x(20)
end
x(21)=opk(1); %assign output of neural network to x(21)
sys=x; %Assign state variable x to sys
function sys=mdlOutputs(t,x,u)
for i=1:21
sys(i)=x(i);
end
```

Linear Programming in Database

Akira Kawaguchi and Andrew Nagel

*Department of Computer Science, The City College of New York, New York, New York
United States of America*

Keywords: linear programming, simplex method, revised simplex method, database, stored procedure.

Abstract

Linear programming is a powerful optimization technique and an important field in the areas of science, engineering, and business. Large-scale linear programming problems arise in many practical applications, and solving these problems requires an integration of data-analysis and data-manipulation capabilities. Database technology has become a central component of today's information systems. Almost every type of organization is now using database systems to store, manipulate, and retrieve data. Nevertheless, little attempt has been made to facilitate general linear programming solvers for database environments. Dozens of sophisticated tools and software libraries that implement linear programming models can be found. But, there is no database-embedded linear programming tool seamlessly and transparently utilized for database processing. The focus of the study in this chapter is to fill this technical gap between data analysis and data manipulation, by solving large-scale linear programming problems with applications built on the database environment. Specifically, this chapter studies the representation of the linear programming model in relational structures, as well as the computational method to solve the linear programming problems. We have developed a set of ready to use stored procedures to solve general linear programming problems. A stored procedure is a group of SQL statements, precompiled and physically stored within a database, thereby having complex logic run inside the database. We show versions of procedures in the open-source MySQL database and commercial Oracle database system. The experiments are performed with several benchmark problems extracted from the Netlib library. Foundations for and preliminary experimental results of this study are presented.*

1. Introduction

* This work has been partly supported by New York State Department of Transportation and New York City Department of Environment Protection.

Linear programming is a powerful technique for dealing with the problem of allocating limited resources among competing activities, as well as other problems having a similar mathematical formulation (Winston, 1994, Richard, 1991, Walsh, 1985). It has become an important field of optimization in the areas of science and engineering and has become a standard tool of great importance for numerous business and industrial organizations. In particular, large-scale linear programming problems arise in practical applications such as logistics for large spare-parts inventory, revenue management and dynamic pricing, finance, transportation and routing, network design, and chip design (Hillier and Lieberman, 2001).

While these problems inevitably involve the analysis of a large amount of data, there has been relatively little work addressing this in the database context. Little serious attempt has been made to facilitate data-driven analysis with data-oriented techniques. In today's marketplace, dozens of sophisticated tools and software libraries that implement linear programming models can be found. Nevertheless, these products do not work with database systems seamlessly. They rather require additional software layers built on top of databases to extract and transfer data in the databases. The focus of our study gathered here is to fill this technical gap between data analysis and data manipulation by solving large-scale linear programming problems with applications built on the database environment.

In mathematics, linear programming problems are optimization problems in which the objective function to characterize optimality of a problem and the constraints to express specific conditions for that problem are all *linear* (Hillier and Lieberman, 2001, Thomas H. Cormen and Stein, 2001). Two families of solution methods, so-called *simplex methods* (Dantzig, 1963) and *interior-point methods* (Karmarkar, 1984), are in wide use and available as computer programs today. Both methods progressively improve series of trial solutions by visiting edges of the feasible boundary or the points within the interior of the feasible region, until a solution is reached that satisfies the constraints and cannot be improved. In fact, it is known that large problem instances render even the best of codes nearly unusable (Winston, 1994). Furthermore, the program libraries available today are found outside the standard database environment, thus mandating the use of a special interface to interact with these tools for linear programming computations.

This chapter gives a detailed account of the methodology and technical issues related to general linear programming in the relational (or object-relational) database environment. Our goal is to find a suitable software platform for solving optimization problems on the extension of a large amount of information organized and structured in the relational databases. In principle, whenever data is available in a database, solving such problems should be done in a *database way*, that is, computations should be closed in the world of the database. There is a standard database language, ANSI SQL, for the manipulation of data in the database, which has grown to a level comparable to most ordinary programming or scripting languages. Eliminating reliance on a commercial linear programming package, thus eliminating the overhead of data transfer between database and package is what we hope to achieve.

There are also the issues of trade-off. A basic nature of linear programming is a collection of matrices defining a problem and a sequence of algebraic operations repeatedly applied to

these matrices, hence giving a perfect match for array-based programming in scientific computations. In general, the relational database is not designed for matrix operations like solving linear programming problems. Indeed, realizing matrix operations on top of the standard relational (or object-relational) structure is non-trivial. On the other hand, at the heart of the database system is the ability to effectively manage resources coupled with an efficient data access mechanism. The response to user is made by the best available sequence of operations, or so-called optimized queries, on the actual data. When handling extremely large matrices, the system probably gives a performance advantage over the unplanned or ad hoc execution of the program causing an insatiable use of virtual memory (thus causing thrashing) for the disposition of arrays.

In this chapter, implementation techniques and key issues for this development are studied extensively. A model suitable to capture the dynamics of linear programming computations is incorporated into the aimed development, by way of realizing a set of procedural interfaces that enables a standard database language to define problems within a database and to derive optimal solutions for those problems without requiring users to write detailed program statements. Specifically, we develop two sets of ready to use *stored procedures* to solve general linear programming problems. A stored procedure is a group of SQL statements, precompiled and physically stored within a database (Gulutzan and Pelzer, 1999, Gulutzan, 2007). It forms a logical unit to encapsulate a set of database operations, defined with an application program interface to perform a particular task, thereby having complex logic run inside the database. The exact implementation of a stored procedure varies from one database to another, but is supported by most major database vendors. To this end, we will show implementations using MySQL open-source database system and freely available Oracle Express Edition selected from the commercial domain. Our choice of these popular database environments is to justify the feasibility of concepts and to draw comparisons of their usability.

The rest of this chapter is organized as follows: Section 2 defines the linear programming model and introduces our approach to express the model in the relational database. Section 3 presents details of developed simulation system and experimental performance studies. Section 4 discusses related work, and Section 5 concludes our work gathered in this chapter.

2. Fundamentals

A linear programming problem consists of a collection of linear inequalities on a number of real variables and a fixed linear function to maximize or minimize. In this section, we summarize the principle technical issues in formulating the problem and some solution method in the relational database environment.

2.1 Linear Programming Principles

Consider the matrix notation expressed in the set of equations (1) below. The *standard form* of the linear programming problem is to maximize an objective function $Z = \mathbf{c}^T \mathbf{x}$, subject to the functional constraints of $\mathbf{Ax} \leq \mathbf{b}$ and non-negativity constraints of $\mathbf{x} \geq \mathbf{0}$, with $\mathbf{0}$ in this case being the n -dimensional zero column vector. A coefficient matrix \mathbf{A} and column vectors \mathbf{c} , \mathbf{b} , and \mathbf{x} are defined in the obvious manner such that each component of the column

vector \mathbf{Ax} is less than or equal to the corresponding component of the column vector \mathbf{b} . But all forms of linear programming problems arise in practice, not just ones in the standard form, and we must deal with issues such as minimization objectives, constraints of the form $\mathbf{Ax} \geq \mathbf{b}$ or $\mathbf{Ax} = \mathbf{b}$, variables ranging in negative values, and so on. Adjustments can be made to transform every non-standard problem into the standard form. So, we limit our discussion to the standard form of the problem.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}, \quad (1)$$

$$\mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

The goal is to find an optimal solution, that is, the most favorable values of the objective function among feasible ones for which all the constraints are satisfied. The *simplex method* (Dantzig, 1963) is an algebraic iterative procedure where each round of computation involves solving a system of equations to obtain a new trial solution for the optimality test. The simplex method relies on the mathematical property that the objective function's maximum must occur on a corner of the space bounded by the constraints of the feasible region.

To apply the simplex method, linear programming problems must be converted into a so-called *augmented form*, by introducing non-negative *slack variables* to replace non-equalities with equalities in the constraints. The problem can then be rewritten in the following form:

$$\mathbf{x}_s = \begin{bmatrix} x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{n+m} \end{bmatrix}, \quad [\mathbf{A} \quad \mathbf{I}] \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_s \end{bmatrix} = \mathbf{b}, \quad \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_s \end{bmatrix} \geq \mathbf{0}, \quad \begin{bmatrix} 1 & -\mathbf{c}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{I} \end{bmatrix} \begin{bmatrix} Z \\ \mathbf{x} \\ \mathbf{x}_s \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} \quad (2)$$

In equations (2) above, $\mathbf{x} \geq \mathbf{0}$, a column vector of slack variables $\mathbf{x}_s \geq \mathbf{0}$, and \mathbf{I} is the $m \times m$ identity matrix. Following the convention, the variables set to zero by the simplex method are called *nonbasic variables* and the others are called *basic variables*. If all of the basic variables are non-negative, the solution is called a basic feasible solution. Two basic feasible solutions are adjacent if all but one of their nonbasic variables are the same. The spirit of the simplex method utilizes a rule for generating from any given basic feasible solution a new one differing from the old in respect of just one variable.

Thus, moving from the current basic feasible solution to an adjacent one involves switching one variable from nonbasic to basic and vice versa for one other variable. This movement involves replacing one nonbasic variable (called *entering* basic variable) by a new one (called

leaving basic variable) and identifying the new basic feasible solution. The simplex algorithm is summarized as follows:

Simplex Method:

1. Initialization: transform the given problem into an augmented form, and select original variables to be the nonbasic variables (i.e., $\mathbf{x} = \mathbf{0}$), and slack variable to be the basic variables (i.e., $\mathbf{x}_s = \mathbf{b}$).
2. Optimality test: rewrite the objective function by shifting all the nonbasic variables to the right-hand side, and see if the sign of the coefficient of every nonbasic variable is positive, in which case the solution is optimal.
3. Iterative Step:
 - 3.1 Selecting an entering variable: as the nonbasic variable whose coefficient is largest in the rewritten objective function used in the optimality test.
 - 3.2 Selecting a leaving variable: as the basic variable that reaches zero first when the entering basic variable is increased, that is, the basic variable with the smallest upper bound.
 - 3.3 Compute a new basic feasible solution: by applying the Gauss-Jordan method of elimination, and apply the above optimality test.

2.2 Revised Simplex Method

The computation of the simplex method can be improved by reducing the number of arithmetic operations as well as the amount of round-off errors generated from these operations (Hillier and Lieberman, 2001, Richard, 1991, Walsh, 1985). Notice that n nonbasic variables from among the $n + m$ elements of $[\mathbf{x}^T, \mathbf{x}_s^T]^T$ are always set to zero. Thus, eliminating these n variables by equating them to zero leaves a set of m equations in m unknowns of the basic variables. The spirit of the *revised simplex method* (Hillier and Lieberman, 2001, Winston, 1994) is to preserve only the pieces of information relevant at each iteration, which consists of the coefficients of the nonbasic variables in the objective function, the coefficients of the entering basic variable in the other equations, and the right-hand side of the equations.

Specifically, consider the equations (3) below. The revised method attempts to derive a basic (square) matrix \mathbf{B} of size $m \times m$ by eliminating the columns corresponding to coefficients of nonbasic variables from $[\mathbf{A}, \mathbf{I}]$ in equations (2). Furthermore, let \mathbf{c}_B^T be the vector obtained by eliminating the coefficients of nonbasic variables from $[\mathbf{c}^T, \mathbf{0}^T]^T$ and reordering the elements to match the order of the basic variables. Then, the values of the basic variables become $\mathbf{B}^{-1}\mathbf{b}$ and $Z = \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{b}$. The equations (2) become equivalent with equations (3) after any iteration of the simplex method.

$$\begin{aligned}
 \mathbf{B} &= \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1m} \\ B_{21} & B_{22} & \cdots & B_{2m} \\ \vdots & \vdots & & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mm} \end{bmatrix}, \\
 \left[\begin{array}{ccc|c} 1 & \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{A} - \mathbf{c}^T & \mathbf{c}_B^T \mathbf{B}^{-1} & Z \\ 0 & \mathbf{B}^{-1} \mathbf{A} & \mathbf{B}^{-1} & \mathbf{x} \\ & & & \mathbf{x}_s \end{array} \right] &= \left[\begin{array}{c} \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{b} \\ \mathbf{B}^{-1} \mathbf{b} \end{array} \right] \tag{3}
 \end{aligned}$$

This means that only \mathbf{B}^{-1} needs to be derived to be able to calculate all the numbers used in the simplex method from the original parameters of \mathbf{A} , \mathbf{b} , \mathbf{c}_B —providing efficiency and numerical stability.

2.3 Relational Representation

A *relational model* provides a single way to represent data as a two-dimensional table or a *relation*. An n -ary relation being a subset of the Cartesian product of n domains has a collection of rows called *tuples*. Implementations of the simplex and revised simplex methods must locate the exact position of the values for the equations and variables of the linear programming problem to solve. However, the position of the tuples in the table is not relevant in the relational model. By definition, tuple ordering and matrix handling are beyond the standard relational features, and these are the most important issues that need to be addressed to implement the linear programming solver within the database using the simplex method. We will explore two distinct methods for representing matrices in the relational model.

Simplex calculations are most conveniently performed with the help of a series of tables known as *simplex tableaux* (Dantzig, 1963, Hillier and Lieberman, 2001). A simplex tableau is a table that contains all the information necessary to move from one iteration to another while performing the simplex method. Let \mathbf{x}_B be a column vector of m basic variables obtained by eliminating the nonbasic variables from \mathbf{x} and \mathbf{x}_S . Then, the initial tableau can be expressed as,

$$\begin{bmatrix} Z & 1 & -\mathbf{c}^T & \mathbf{0} & 0 \\ \mathbf{x}_B & \mathbf{0} & \mathbf{A} & \mathbf{I} & \mathbf{b} \end{bmatrix} \quad (4)$$

The algebraic treatment based on the revised simplex method (Hillier and Lieberman, 2001, William H. Press and Flannery, 2002) derives the values at any iteration of the simplex method as,

$$\begin{bmatrix} Z & 1 & \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{A} - \mathbf{c}^T & \mathbf{c}_B^T \mathbf{B}^{-1} & \mathbf{c}_B^T \mathbf{B}^{-1} \mathbf{b} \\ \mathbf{x}_B & \mathbf{0} & \mathbf{B}^{-1} \mathbf{A} & \mathbf{B}^{-1} & \mathbf{B}^{-1} \mathbf{b} \end{bmatrix} \quad (5)$$

For the matrices expressed (4) and (5), the first two column elements do not need to be stored in persistent memory. Thus, the simplex tableau can be a table using the rest of the three column elements in the relational model. Creating table instances as simplex tableaux is perhaps the most straightforward way. Indeed, our MySQL implementation in the next section uses this representation, in which a linear programming problem in the augmented form (equations (2)) can be seen as a relation:

$$\text{tableau}(\underline{\text{id}}, x_1, x_2, \dots, x_n, \text{rhs})$$

A variable of the constraints and the objective function becomes an attribute of the relation, together with the right hand side that becomes the rhs column on the table. The id column serves as a *key* that can uniquely determine every variable of the constraints and of the objective function in the tuple. A constraint of the linear programming problem in the augmented form is identified by a unique positive integer value ranging from 1 to n in the id column, where n is the number of constraints for the problem plus the objective function. Thus by applying relational operations, it is feasible to know the position of every constraint and variable for a linear programming problem, and to proceed with the matrix operations necessary to implement the simplex algorithm. See Figure 1 for the table instance populated with a simple example.

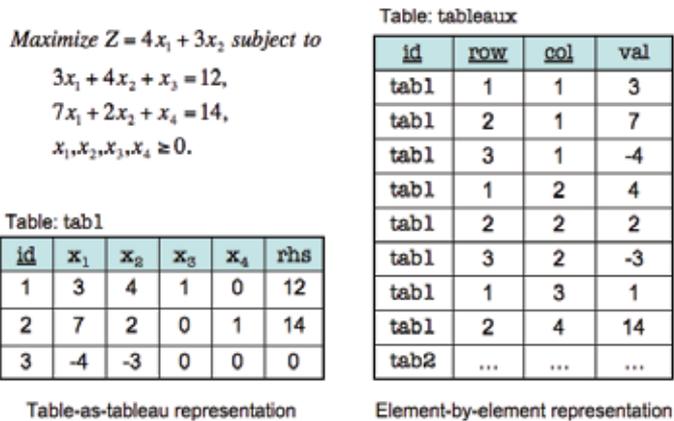


Fig. 1. Two representations of the initial simplex tableau in the relational database

The main drawback of this design is a fixed structure of table. An individual table needs to be created for each problem, and the cost of defining (and dropping) table becomes part of the process implementing the simplex method. The number of tables in the database will increase as the collection of problems to solve accumulates. This may cause administrative strain for database management. Subtle issues arise in the handling of large-scale problems. The table maps to the full instance of the matrix even if the problem has sparsely populated non-zero data. Thousands of zero values (or null values specific to database) held in a tuple pile up a significant amount of space. Besides, a tuple of a large number of non-zero values is problematic because the physical record holding such a tuple may not fit into a disk block. Accessing a spanned record over multiple disk blocks is time-consuming.

As an alternative to the table-as-tableaux structure, element-by-element representation can be considered. The simplex tableaux are decomposed to a collection of values, each of which is a tuple consisting of a tableau id, a row position, a column position, and a value in the specified position. This is to say that the table no longer possesses the shape of a tableau but has the information to locate every element in the tableau. Missing elements are zeros, thus space efficient for sparse contents. A single table can gather all the problem instances, in that the elements in the specific tableau are found by the use of the tableau id. The size of the tuple is small because there are only four attributes in the tuple. In return, there is a time

overhead for finding a specific element in the table. Our Oracle XE implementation detailed in the next section utilizes this representation.

3. System Development

The availability of real-time databases capable of accepting and solving linear programming problems helps us examine the effectiveness and practical usability in integrating linear programming tools into the database environment. Towards this end, a general linear programming solver is developed on top of the *de facto* standard database environment, with the combination of a PHP application for the front-end and a MySQL or Oracle application for the backend. Note that the implementation of this linear programming solver is strictly within the database technology, not relying on any outside programming language.

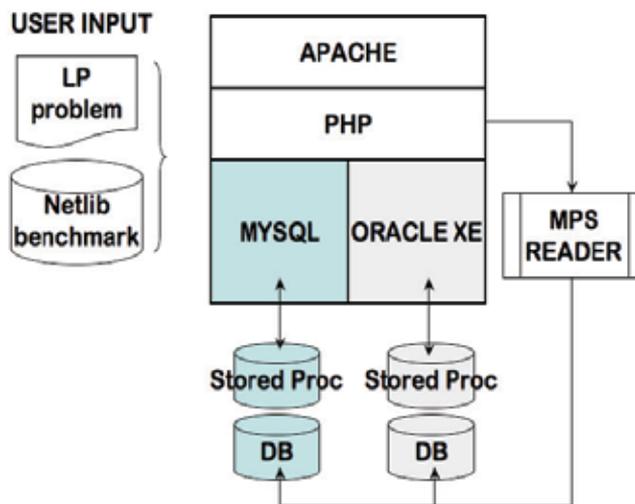


Fig. 2. Architecture of the implemented linear programming solver

The systems architecture is summarized in Figure 2. The PHP front-end enables the user to input the number of variables and number of constraints of the linear programming problem to solve. With these values, it generates a dynamic Web interface to accept the values of the objective function and the values of the constraining equations. The Web interface also allows the user to upload a file in a MPS (Mathematical Programming System) format that defines a linear programming problem. The MPS file format serves as a standard for describing and archiving linear programming and mixed integer programming problems (Organization, 2007). A special program is built to convert MPS data format into SQL statements for populating a linear programming instance. The main objective of this development is to obtain benchmark performances for large-scale linear programming problems.

The MySQL and Oracle back-ends perform iterative computations by the use of a set of stored procedures precompiled and integrated into the database structure. The systems encapsulate an API for processing a simplex method that requires the execution of several SQL queries to produce a solution. The input and output of the system are shown in Figure 3 in which each table of the right figure represents the tableau containing the values resulted from each iteration of the simplex method. The system presents successive transformations and optimal solution if it exists.

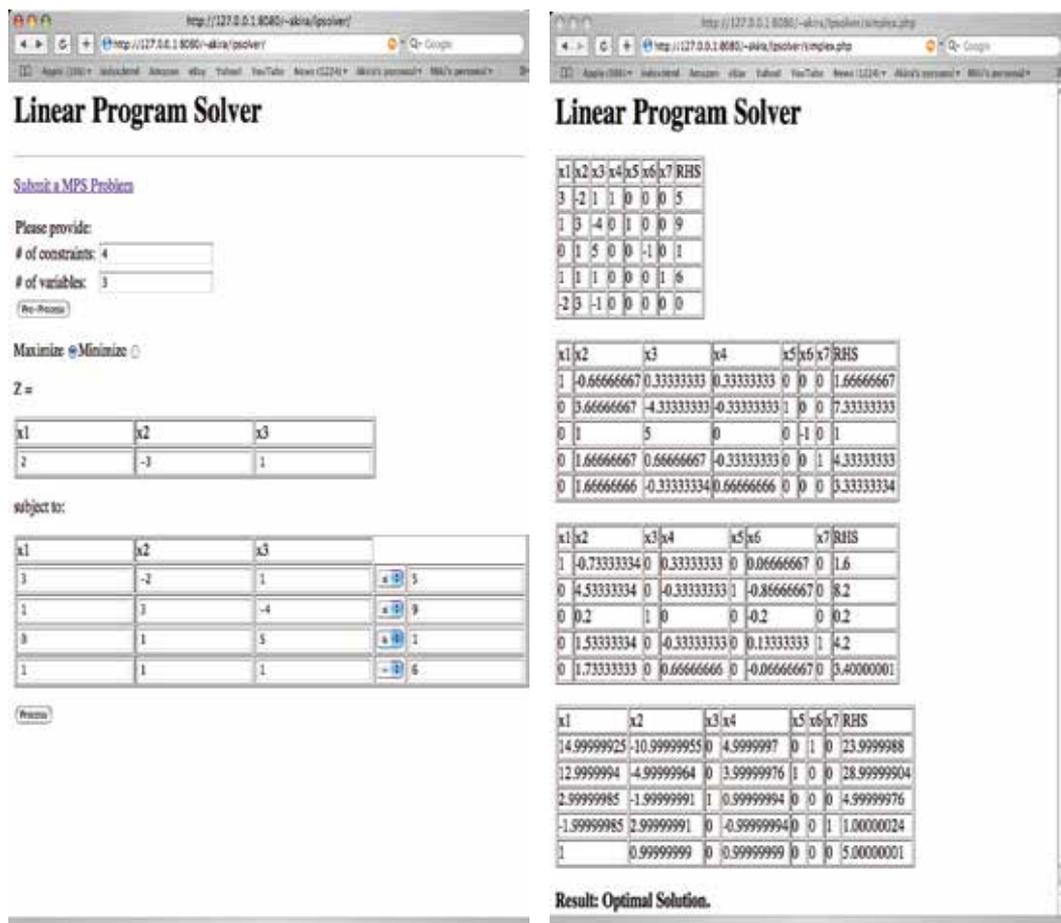


Fig. 3. Simplex method iterations and optimal solution

3.1 Stored Procedure Implementation MySQL

Stored procedures can have direct accesses to the data in the database, and run their steps directly and entirely within the database. The complex logic runs inside the database engine, thus faster in processing requests because numerous context switches and a great deal of network traffic can be eliminated. The database system only needs to send the final results back to the user, doing away with the overhead of communicating potentially large amounts of interim data back and forth (Gulutzan, 2007, Gulutzan and Pelzer, 1999).

Name	m	n	Nonzeros	Optimal value	Time	Standard deviation
ADLITTLE	57	97	465	2.2549496316E+05	1 min. 25 sec.	2.78 sec.
AFIRO	28	32	88	-464.7531428596	35 sec.	1.67 sec.
BLEND	75	83	521	-3.0812149846E+01	1 min. 5 sec.	3.20 sec.
BRANDY	22 1	24 9	2150	1.5185098965E+03	2 min. 50 sec.	4.25 sec.

Table 1. MySQL Experimental set and measured execution time

Stored procedures are supported by most DBMSs, but there is a fair amount of variation in their syntax and capabilities even their internal effects are almost invisible. Our development uses MySQL version 5.0.22 at the time of this writing (as for MySQL version 5, stored procedures are supported). The next code listing is the stored procedure used to create the table to store the linear programming problem to be solved by the application (Perez, 2007). The first part of the stored procedure consists of the prototype of the function and the declaration of the variables to be used in the procedure.

```

DELIMITER $$
DROP PROCEDURE IF EXISTS
  \lpsolver\.`createTable` $$
CREATE PROCEDURE \lpsolver\.`createTable`
  (constraints INT, variables INT)
BEGIN
  DECLARE i INT;
  DECLARE jiterator VARCHAR(50);
  DECLARE statement VARCHAR(1000);
  DROP TABLE IF EXISTS tableaux;

```

Because of the dynamic nature of the calculations for solving linear programming problems, our stored procedure relies on the extensive use of prepared SQL statements. In the next code block, the SQL statement to create a table is generated on the fly, based on the number of variables and constraints of the problem to solve. The generated procedure is then passed to the database for execution.

```

SET statement = 'CREATE TABLE
  tableaux(id INT(5) PRIMARY KEY, ' ;
SET i = 1;
table_loop:LOOP
  IF i > constraints + variables + 1 THEN
    LEAVE table_loop;
  END IF;
  SET jiterator = CONCAT('j', i);
  SET statement = CONCAT(statement,
    jiterator);
  SET statement = CONCAT(statement,
    ' DOUBLE DEFAULT 0');
  IF i <= constraints + variables THEN
    SET statement = CONCAT(statement, ', ');

```

```
END IF;
SET i = i + 1;
END LOOP table_loop;
SET statement = CONCAT(statement, ' ');
SET @sql_call = statement;
PREPARE s1 FROM @sql_call;
EXECUTE s1;
DEALLOCATE PREPARE s1;
END $$
DELIMITER;
```

3.2 Experimental Results MySQL

To see the effectiveness of the implementation, various linear programming problems were selected from commonly available Netlib linear programming library (Organization, 2007). As one case, see Table 1 for a sufficiently large problem set. The values m and n indicate the size, $m \times n$, of the coefficient matrix \mathbf{A} in equations (1), or equivalently, m is the number of constraints and n is the number of decision variables.

All experiments were performed by an Intel 586 based standalone machine with 1.2 GHz CPU and 512 MB memory that was running MySQL 5.0.22. The data values were extracted from Netlib MPS files to populate the problems into the database prior to run the simplex method. The time measured does not include this data preparation process, but only the execution of the stored procedure to produce a solution. The time listed in Table 1 is the average of ten executions of each problem. The results are based on the implementation of the revised simplex method contained in the stored procedures.

One limiting factor is the fact that MySQL allows to have up to 1000 columns on a table. Given that this implementation is based on mapping of a simplex tableau into a database relation, the number of variables plus the number of constraints cannot exceed the number of columns allowed for a MySQL table. This prohibited us from testing the problems in the Netlib library that exceed the column size of 1000. Finally, we observed one problem when trying to find optimal solutions for larger problems with higher numbers of columns, variables and zero elements. The computation never came to an end, indicating that the problem had become unbounded, which can be attributed to the tableau becoming ill-conditioned as a consequence of truncation errors resulted from repeated matrix operations (Kawaguchi and Perez, 2007).

3.3 Stored Procedure Implementation in Oracle XE

A Linear Programming Solver was implemented in Oracle XE stored procedures with a simple web interface built in PHP. Oracle XE is a free version of the Oracle database system subject to some restrictions. Notably, Oracle XE will only utilize a single processor, and total user data is limited to 4 GB. Still, stored procedures are entirely supported, as well as advanced indexing techniques, making Oracle XE an attractive alternative.

The web interface shown in Figure 4 provides for creation, editing, and display of large matrices, and allows the user to perform elementary matrix operations. The “Linear

Programming” menu provides options for uploading and parsing standard MPS files, for solving the problem automatically, and for viewing performance data. The “Work Tableau” option is shown, and provides an interface where the user can view the tableau, or any portion of it. They can choose an element to pivot on, or by clicking the “suggest” button, the column and row selected by the simplex method is high-lighted and displayed. This was useful for debugging, but also serves as a good educational tool since a student can go through the algorithm step by step.

Fig. 4. Web interface showing tableau with next iteration highlighted

The data model was also changed in this implementation to address the limitation of max number of columns allowed in a table. Rather than creating a table with enough columns to contain the simplex tableau, a matrix is represented by two tables, one describing the properties, and one containing the row position, column position, and value of each element. This has the added benefit of simplifying matrix operations. Since the tables described below hold any matrices stored in the system, creation, deletion, and alteration of matrices relies only on INSERT, DELETE, and UPDATE statements, with no need for ‘on the fly’ table creation or procedure compilation.

```
matrix_property(matrix_id, name, row_size, column_size)
matrix_values(matrix_id, row_position, column_position, value)
```

Although allowing for an indefinite number of columns in a stored matrix, this model introduces a problem in data look up. In the MySQL implementation, since each row of the

matrix was a tuple in the relation, once a row is retrieved by the database, every element in the row is either in main memory or efficiently buffered since the tuple is stored contiguously on disk. But in the Oracle implementation, getting each element in a row could require a new disk read. Fortunately, Oracle XE supports Hash Index for fast retrieval of tuples that share a common hash value. Since in the Oracle model, any two elements in the same row share the same `matrix_id` and `row_position`, we can build such an index.

In the Oracle XE environment, this is achieved by first creating a hash cluster (equivalent to hash buckets), then creating the table that is designated to be stored in the cluster according to a hash of at least one of its attributes.

```
CREATE CLUSTER Matrix_index (matid NUMBER, rowpos NUMBER) SIZE
512 SINGLE TABLE HASHKEYS 1000;

CREATE TABLE "MVALUE"
(
  "MATID" NUMBER NOT NULL ENABLE,
  "ROWPOS" NUMBER NOT NULL ENABLE,
  "COLPOS" NUMBER NOT NULL ENABLE,
  "CELL_VALUE" NUMBER(26,14) NOT NULL ENABLE,
  CONSTRAINT "MVALUE_UK1" UNIQUE ("MATID", "ROWPOS", "COLPOS")
  ENABLE,
  CONSTRAINT "MVALUE_FK" FOREIGN KEY ("MATID") REFERENCES
  "MPROPERTY" ("MATID") ON DELETE CASCADE ENABLE
)
CLUSTER "Matrix_index" ("MATID", "ROWPOS");
```

To actually benefit from this index, it is necessary to make use of Cursors in the stored procedures that operate on the matrices. Cursors are featured in many database systems, and provide an interface to declare complex SELECT statements and iterate over the results. By declaring cursors, rather than using a SELECT statement inside a FOR loop, the query optimizer is better able to take advantage of the hash index and retrieve entire rows of the matrix with minimal disk I/O. Implementing the solver in this way resulted in more than 50% performance increase, particularly as problem size was increased.

Name	<i>m</i>	<i>n</i>	Non zeros	Optimal Value (calculated)	Std. Err.	Iterations	Avg. Time (sec)	Std. Dev
ISRAEL	175	143	2358	-896641.4612	0.0004%	308	784	1.41
LOTFI	154	309	1086	-25.26470426	0.0000%	203	79.9	9.18
AFIRO	28	33	88	-464.7531429	0.0000%	11	0.4	0.52
SC105	106	104	281	-52.20206121	0.0000%	110	77.9	1.1
SC205	206	204	552	-52.20206121	0.0000%	257	702.2	2.9
ADLITTLE	57	98	465	225494.9384	0.0000%	146	38.1	1.73
BLEND	75	84	521	-30.82213945	0.0324%	118	41.2	0.63
BRANDY	221	250	2150	557.6518123	63.2764%	659	1433.8	7.11

Table 2. Oracle Experimental set and measured execution time

3.4 Experimental Results Oracle XE

As before, a set of linear programming problems was selected from the Netlib library (Organization, 2007). The results are presented in Table 2, where again, m is the number of constraints and n is the number of decision variables. The standard error is calculated based on the optimal value our solver returned compared to the optimal values published by Netlib. The Oracle experiments were run on an Intel D865GV board with 3 GHz Pentium 4 CPU and 2 GB memory running Oracle XE 10g. Other parameters and timing of the experiment are as described in 3.2.

While the model used in the Oracle implementation does allow for storage and simple manipulation of matrices larger than 1000×1000 , it did not solve all the problems experienced in the MySQL version. Truncation and rounding errors created deviance from the published optimal value, hence the inclusion of standard error in Table 2. For larger problems, this sometimes degenerated to an 'ill-conditioned' state, as with the MySQL implementation and the algorithm may not finish or may report incorrect results as with BRANDY.

Rounding errors were sometimes more of an issue in the Oracle implementation because it uses the Big M variant of the Simplex method when dealing with problems in non-standard form. Briefly, this involves introducing artificial variables to make each constraint feasible for the basic solution and penalizing those artificial variables with a large coefficient in the objective function, this penalty being the Big M. While this method is easy to implement, it requires more iterations, which introduces more potential for rounding/truncation errors.

4. Related Work

A vast amount of effort for the establishment of theory and practice is observed today. Certain special cases of linear programming, such as network flow problems and multi-commodity flow problems are considered important enough to have generated much research on specialized algorithms for their solution (Winston, 1994, Thomas H. Cormen and Stein, 2001, Hillier and Lieberman, 2001). A number of algorithms for other types of optimization problems work by solving linear programming problems as sub-problems. Historically, ideas from linear programming have inspired many of the central concepts of optimization theory, such as duality, decomposition, and the importance of convexity and its generalizations (Hillier and Lieberman, 2001).

There are approaches considered to fit a linear programming model, such as integer programming and nonlinear programming (Alexander, 1998, Richard, 1991, Hillier and Lieberman, 2001). But, our research focuses on the area of iterative methods for solving linear systems. Some of the most significant contributions and the chain of contributions building on each other are summarized in (Saad and van der Vorst, 2000), especially a survey of the transition from simplex methods to interior-point methods is presented in (Wang, 99). In terms of implementation techniques, the work of (Morgan, 1976, Shamir, 1987) provided us with introductory sources for reference. There are online materials such as (Optimization Technology Center and Laboratory, 2007, Organization, 2007) to help us understand the details and plan for experimental design.

The contents of this chapter are extended from the work gathered in (Kawaguchi and Perez, 2007), in which the experimental performance of MySQL implementation is shown. A more detailed implementation of MySQL stored procedures can be found in (Perez, 2007).

5. Conclusion

The subject of this research is to respond a lack of database tools for solving a linear programming problem defined within a database. We described the aim and approach for integrating a linear programming method into today's database system, with our goal in mind to establish a seamless and transparent interface between them. As demonstrated, this is feasible by the use of stored procedures, the emerging database programming standard that allows for complex logic to be embedded as an API in the database, thus simplifying data management and enhancing overall performance. As a summary, contributions of the discussions presented in this chapter are threefold: First, we present a detailed account on the methodology and technical issues to integrate a general linear programming method into relational databases. Second, we present the development as forms of stored procedures for today's representative database systems. Third, we present an experimental performance study based on a comprehensive system that implements all these concepts.

Our implementation of general linear programming solvers is on top of the PHP, MySQL, and Oracle software layers. The experiments with several benchmark problems extracted from Netlib library showed its correct optimal solutions and basic performance measures. However, due to the methods used, rounding errors were still an issue for large problems despite the system having the capacity to work with large matrices. We thus plan to continue this research in several directions. Although the Oracle system can work with large matrices, both implementations have too much rounding error to solve linear programming problems that would be considered large by commercial standards. This should be addressed by the implementation of a more robust method. Overall, the code must be optimized to reduce the execution time, which could also be improved by tuning the size and number of hash buckets in the index. We will perform more experiments to collect additional performance measures. Non-linear and other optimization methods should also be explored.

6. References

- Alexander, S. (1998). *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, NY.
- Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton University Press, Princeton, N.J.
- Gulutzan, P. (2007). *MySQL 5.0 New Features: Stored Procedures*. MySQL AB, <http://www.mysql.com>.
- Gulutzan, P. and Pelzer, T. (1999). *SQL-99 Complete, Really*. CMP Books.
- Hillier, F. S. and Lieberman, G. J. (2001). *Introduction to Operations Research*. McGraw-Hill, 8th edition.
- Karmarkar, N. K. (1984). A new polynomial-time algorithm for linear programming and extensions. *Combinatorica*, 4:373–395.

- Kawaguchi, A. and Perez, A. J. (2007). Linear programming for database environment. *ICINCO-ICSO 2007*: 186-191
- Morgan, S. S. (1976). A comparison of simplex method algorithms. Master's thesis, University of Florida.
- Optimization Technology Center, N. U. and Laboratory, A. N. (2007). The linear programming frequently asked questions.
- Organization, T. N. (2007). The netlib repository at utk and ornl.
- Perez, A. J. (2007). Linear programming for database environment. Master's thesis, City College of New York.
- Richard, B. D. (1991). *Introduction To Linear Programming: Applications and Extensions*. Marcel Dekker, New York, NY.
- Saad, Y. and van der Vorst, H. (2000). Iterative solution of linear systems in the 20-th century. *JCAM*.
- Shamir, R. (1987). The efficiency of the simplex method: a survey. *Manage. Sci.*, 33(3):301-334.
- Thomas H. Cormen, Charles E. Leiserson, R. L. R. and Stein, C. (2001). *Introduction to Algorithms, Chapter29: Linear Programming*. MIT Press and McGraw-Hill, 2nd edition.
- Walsh, G. R. (1985). *An Introduction to Linear Programming*. John Wiley & Sons, New York, NY.
- Wang, X. (99). From simplex methods to interior-point methods: A brief survey on linear programming algorithms.
- William H. Press, Saul A. Teukolsky, W. T. V. and Flannery, B. P. (2002). *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University.
- Winston, W. L. (1994). *Operations Research, Applications and Algorithms*. Duxbury Press.

Searching Model Structures Based on Marginal Model Structures

Sung-Ho Kim and Sangjin Lee

Department of Mathematical Sciences

Korea Advanced Institute of Science and Technology

Daejeon, 305-701, South Korea

1. Introduction

Graphs are used effectively in representing model structures in a variety of research fields such as statistics, artificial intelligence, data mining, biological science, medicine, decision science, educational science, etc. We use different forms of graphs according to the nature of the random variables involved. For instance, arrows are used when the relationship is asymmetric as when it is causal or temporal, and undirected edges are used when the relationship is associative.

When a random field is Markov with respect to a triangulated graph, i.e., a decomposable graph, which does not have a cycle of length 4 or larger, its corresponding probability model is expressed in a factorized form which facilitates computation over the probability distribution of the random field (Kemeny et al., 1976). This computational feasibility, among others, makes such a Markov random field a most favored random field. Literature is abundant in regard to the properties of the Markov random field which is Markov with respect to a decomposable graph (see Chapter 12 of Whittaker (1990) and Lauritzen (1996)). We call such a random field a decomposable graphical model.

There have been remarkable improvements in learning graphical models in the form of a Bayesian network (Pearl, 1986 & 1988; Heckerman et al., 1995; Friedman & Goldszmidt, 1998; Neil et al., 1999; Neapolitan, 2004) from data. This learning however is mainly instrumented by heuristic searching algorithms and the model searching is usually NP-hard [Chickering (1996)]. A good review is given in Cooper (1999) and Neapolitan (2004) on structural discovery of Bayesian or causal networks from data. Since a Bayesian network can be transformed into a decomposable graph [Lauritzen and Spiegelhalter (1988)], the method of model combination which is proposed in this paper would lead to an improvement in graphical modelling from data. This method would be useful when we don't have data which are large enough for the number of the random variables that are involved in the data. In this situation, it is desirable to develop marginal models of manageable sizes for subsets of variables and then search for a model for the whole set of variables based on the marginal models.

The main idea of the method to be proposed is similar to constraint-based learning as described in Neapolitan (2004) (also see Meek (1995) and Spirtes et al. (2000)) where we

construct a Bayesian network based on a list of constraints which are given in terms of conditional independence among a given set of random variables. But a noteworthy difference between the two is that, while the statements of conditional independencies are an extraction, as for the constraintbased learning, from the probability model of the whole set of the variables involved, the statements of conditional independencies for the method to be proposed are from the marginal probability models of the subsets of variables. This difference in how we extract the statements of conditional independence is the main source of the difference between the two methods.

In deriving the method of the paper, it is imperative that we make use of the relationship between the joint (as against marginal) model structure and its marginal model structure. Kim (2006) introduced a certain type of subgraph, called Markovian subgraph, and investigated its properties as a subgraph of a decomposable graph. Some of the properties play a crucial role in the process of constructing a decomposable graph based on a collection of its Markovian subgraphs. We will elaborate on this in later sections. Kim (2004) called our attention to the relationship between a set of probability models and a set of model structures and proved a theorem to the effect that we may deal with model structures of marginal models in search of the model structure of the joint probability model for the whole set of variables involved in data. In 1 this respect, we will use graphs to represent model structures and compare the joint model with its marginal models using graphs.

This paper consists of 8 sections. Section 2 introduces notations and graphical terminologies along with new concepts such as Markovian subgraph and Markovian subpath. A simple but motivational example is considered in Section 3 with some prelude remarks of the method to be proposed. Sections 4 and 5 then introduces theorems and a new type of graph that are instrumental for the model-combination. Section 6 describes the model-combining process and it is illustrated in section 7. The paper is concluded in section 8 with summarizing remarks.

2. Notation and preliminaries

We will consider only undirected graphs in the paper. We denote a graph by $\mathcal{G} = (V, E)$, where V is the set of the indexes of the variables involved in \mathcal{G} and E is a collection of ordered pairs, each pair representing that the nodes of the pair are connected by an edge. Since \mathcal{G} is undirected, that (u, v) is in E is the same as that (v, u) is in E . If $(u, v) \in E$, we say that u is a neighbor node of or adjacent to v or vice versa. We say that a set of nodes of \mathcal{G} forms a complete subgraph of \mathcal{G} if every pair of nodes in the set is adjacent to each other. If every node in A is adjacent to all the nodes in B , we will say that A is adjacent to B . A maximal complete subgraph is called a clique of \mathcal{G} , where the maximality is in the sense of set-inclusion. We denote by $C(\mathcal{G})$ the set of cliques of \mathcal{G} .

A path of length n is a sequence of nodes $u = v_0, \dots, v_n = v$ such that $(v_i, v_{i+1}) \in E$, $i = 0, 1, \dots, n - 1$ and $u \neq v$. If $u = v$, the path is called an n -cycle. If $u \neq v$ and u and v are connected by a path, we write $u \rightleftharpoons v$. We define the connectivity component of u as

$$[u] = \{v \in V; v \rightleftharpoons u\} \cup \{u\}.$$

So, we have

$$v \in [u] \iff u \rightleftharpoons v \iff u \in [v].$$

We say that a path, $v_1, \dots, v_n, v_1 \neq v_n$, is intersected by A if $A \cap \{v_1, \dots, v_n\} \neq \emptyset$ and neither of the end nodes of the path is in A . We say that nodes u and v are separated by A if all the paths from u and v are intersected by A . In the same context, we say that, for three disjoint sets A, B , and C , A is separated from B by C if all the paths from A to B are intersected by C and write $(A|C|B)_{\mathcal{G}}$. A non-empty set B is said to be intersected by A if B is partitioned into three sets B_1, B_2 , and $B \cap A$ and B_1 and B_2 are separated by A in \mathcal{G} . The complement of a set A is denoted by A^c and the cardinality of a set A by $|A|$.

For $A \subset V$, we define an *induced subgraph* of \mathcal{G} confined to A as $\mathcal{G}_A^{ind} = (A, E \cap (A \times A))$. We also define a graph, called a *Markovian subgraph* of \mathcal{G} confined to A , which is formed from \mathcal{G}_A^{ind} by completing the boundaries in \mathcal{G} of the connectivity components of the complement of A and denote it by \mathcal{G}_A . In other words, $\mathcal{G}_A = (A, E_A)$ where

$$E_A = (E \cap A \times A) \cup \{(u, v) \in A \times A; u \text{ and } v \text{ are not separated by } A \setminus \{u, v\} \text{ in } \mathcal{G}\}.$$

Let a path, π say, from u to v is a sequence of edges (u_i, u_{i+1}) with $u_0 = u$ and $u_k = v$. Then we will say that a sequence of edges $(u_{i_1}, u_{i_2}), \dots, (u_{i_r}, u_{i_{r+1}}), 0 \leq i_1 < i_2 < \dots < i_{r+1} \leq k$, is a Markovian subpath of π .

If $\mathcal{G} = (V, E)$, $\mathcal{G}' = (V, E')$, and $E' \subseteq E$, then we say that \mathcal{G}' is an edge-subgraph of \mathcal{G} and write $\mathcal{G}' \subseteq \mathcal{G}$. A subgraph of \mathcal{G} is either a Markovian subgraph, an induced subgraph, or an edge-subgraph of \mathcal{G} . If \mathcal{G}' is a subgraph of \mathcal{G} , we call \mathcal{G} a supergraph of \mathcal{G}' .

Although decomposable graphs are well known in literature, we define them here for completeness.

Definition 2.1. A triple (A, B, C) of disjoint, nonempty subsets of V is said to form a decomposition of \mathcal{G} if $V = A \cup B \cup C$ and the two conditions below both hold:

- (i) A and B are separated by C ;
- (ii) \mathcal{G}_C^{ind} is complete.

By recursively applying the notion of graph decomposition, we can define a decomposable graph.

Definition 2.2. \mathcal{G} is said to be decomposable if it is complete, or if there exists a decomposition (A, B, C) into decomposable subgraphs $\mathcal{G}_{A \cup C}^{ind}$ and $\mathcal{G}_{B \cup C}^{ind}$.

For a decomposable graph, we can find a sequence of cliques C_1, \dots, C_k of \mathcal{G} which satisfies the following condition

[see Proposition 2.17 of Lauritzen (1996)]: with $C_{(j)} = \cup_{i=1}^j C_i$ and $S_j = C_j \cap C_{(j-1)} \neq \emptyset$, for all $i > 1$, there is a $j < i$ such that $S_i \subseteq C_j$

By this condition for a sequence of cliques, we can see that S_j is expressed as an intersection of neighboring cliques of \mathcal{G} . If we denote the collection of these S_j 's by $\chi(\mathcal{G})$, we have, for a decomposable graph \mathcal{G} , that

$$\chi(\mathcal{G}) = \{a \cap b; a, b \in \mathcal{C}(\mathcal{G}), a \neq b\} \tag{1}$$

It is possible for some decomposable graph \mathcal{G} that there are sets, a and b , in $\chi(\mathcal{G})$ such that $a \subset b$.

The cliques are elementary graphical components and the S_j is obtained as intersection of neighboring cliques. So, we will call the S_j 's prime separators (PSs for short) of the decomposable graph \mathcal{G} . The PSs in a decomposable graph may be extended to separators of prime graphs in any undirected graph, where the prime graphs are defined as the maximal subgraphs without a complete separator in Cox and Wermuth (1999).

3. Simple example with remarks

Graph \mathcal{G} can be represented in the same way as a graphical log-linear model is represented in terms of generators [Fienberg (1980)]. If \mathcal{G} consists of cliques C_1, \dots, C_r , we will write

$$\mathcal{G} = [C_1] \cdots [C_r].$$

For instance, if \mathcal{G} is of five nodes and $C_1 = \{1, 2\}$, $C_2 = \{2, 3\}$, $C_3 = \{3, 4, 5\}$, then $\mathcal{G} = [12][23][345]$. In this context, the terms *graph* and *model structure* are used in the same sense.

Suppose that we are given a pair of simple graphical models where one model is of random variables X_1, X_2, X_3 with their inter-relationship that X_1 is independent of X_3 conditional on X_2 and the other is of X_1, X_2, X_4 with their inter-relationship that X_1 is independent of X_4 conditional on X_2 . From this pair, we can imagine a model structure for the four variables X_1, \dots, X_4 . The two inter-relationships are pictured at the left end of Figure 1. The graph at the top of the two at the left is represented by $[12][23]$ and the one at the bottom by $[12][24]$. X_1 and X_2 are shared in both models, and assuming that none of the four variables are marginally independent of the others, we can see that the following joint models have the marginals, $[12][23]$ and $[12][24]$:

$$[12][24][23], [12][24][34], [12][23][34], [12][234], \tag{2}$$

which are displayed in graph in Figure 1. Note that the first three of these four models are submodels or edge-subgraphs of the last one.

It is important to note that some variable(s) are independent of the others, conditional on X_2 in the pair of marginals, and in all the models in (2). That conditional independence takes place conditional on the same variable in the marginal models and also in the joint models underlies the main theme of the method to be proposed in the paper.

In addressing the issue of combining graphical model structures, we can not help using independence graphs and related theories to derive desired results with more clarity and refinement. The conditional independence embedded in a distribution can be expressed to some level of satisfaction by a graph in the form of graph-separateness [see, for example, the separation theorem in p. 67, Whittaker (1990)]. We instrument the notion of conditional independence with some particular sets of random variables in a model, where the sets form a basis of the model structure so that the Markov property among the variables of the model may be preserved between the joint model and its marginals. The sets are



Fig. 1. Two marginal models on the left and the four joint models on the right

prime separators. In the simple example, X_2 forms the basis. Without the variable, X_2 , the conditional independence disappears.

It is shown that if we are given a graphical model with its independence graph, \mathcal{G} , and some of its marginal models, then under the decomposability assumption of the model we can find a graph, say \mathcal{H} , which is not smaller than \mathcal{G} and in which the graph-separateness in the given marginal models is preserved (Theorem 4.3). This graph-separateness is substantiated by the prime separators which are found in the graphs of the marginal models. In combining marginal models into \mathcal{H} , we see to it that these prime separators appear as the only prime separators in \mathcal{H} . This is reflected in the model-combining procedure described in Section 6.

4. Theorems useful for model-combination

Let $\mathcal{G} = (V, E)$ be the graph of a decomposable model and let V_1, V_2, \dots, V_m be subsets of V . The m Markovian subgraphs, $\mathcal{G}_{V_1}, \mathcal{G}_{V_2}, \dots, \mathcal{G}_{V_m}$, may be regarded as the structures of m marginal models of the decomposable model, \mathcal{G} . For simplicity, we write $\mathcal{G}_i = \mathcal{G}_{V_i}$.

Definition 4.1. Suppose there are m Markovian subgraphs, $\mathcal{G}_1, \dots, \mathcal{G}_m$. Then we say that graph \mathcal{H} of a set of variables V is a combined model structure (CMS) corresponding to $\mathcal{G}_1, \dots, \mathcal{G}_m$, if the following conditions hold:

- (i) $\cup_{i=1}^m V_i = V$.
- (ii) $\mathcal{H}_{V_i} = \mathcal{G}_i$, for $i = 1, \dots, m$. That is, \mathcal{G}_i are Markovian subgraphs of \mathcal{H} .

We will call \mathcal{H} a maximal CMS corresponding to $\mathcal{G}_1, \dots, \mathcal{G}_m$ if adding any edge to \mathcal{H} invalidates condition (ii) for at least one $i = 1, \dots, m$. Since \mathcal{H} depends on $\mathcal{G}_1, \dots, \mathcal{G}_m$, we denote the collection of the maximal CMSs by $\Omega(\mathcal{G}_1, \dots, \mathcal{G}_m)$.

According to this definition, a CMS is a Markovian supergraph of each $\mathcal{G}_i, i = 1, \dots, m$. There may be many CMSs that are obtained from a collection of Markovian subgraphs as we saw in (2).

In the theorem below, $\mathcal{C}_{\mathcal{G}}(A)$ is the collection of the cliques which include nodes of A in the graph \mathcal{G} . The proof is intuitive. The symbol, $\langle \cdot | \cdot | \cdot \rangle$, follows Pearl (1988), and for three disjoint sets, A, B , and C , $\langle A|C|B \rangle_{\mathcal{G}}$ means that A is separated from B by C in \mathcal{G} .

Theorem 4.2. Let $\mathcal{G}' = (V', E')$ be a Markovian subgraph of \mathcal{G} and suppose that, for three disjoint subsets A, B, C of V' , $\langle A|B|C \rangle_{\mathcal{G}'}$. Then

- (i) $\langle A|B|C \rangle_{\mathcal{G}}$;
- (ii) For $W \in \mathcal{C}_{\mathcal{G}}(A)$ and $W' \in \mathcal{C}_{\mathcal{G}}(C)$, $\langle W|B|W' \rangle_{\mathcal{G}}$.

Proof. Since

$$\langle A|B|C \rangle_{\mathcal{G}'}, \tag{3}$$

there is no path in \mathcal{G}' between A and C that bypasses B . If (i) does not hold, it is obvious that (3) does not hold either. Now suppose that result (ii) does not hold. Then there must be a path from a node in A to a node in C bypassing B . This implies negation of the condition (3) by the definition of the Markovian subgraph. Therefore, result (ii) must hold.

Recall that if $\mathcal{G}_i, i = 1, 2, \dots, m$, are Markovian subgraphs of \mathcal{G} , then \mathcal{G} is a CMS. For a given set S of Markovian subgraphs, there may be many maximal CMSs, and they are related with S through PSs as in the theorem below.

Theorem 4.3. Let there be Markovian subgraphs $\mathcal{G}_i, i = 1, 2, \dots, m$, of a decomposable graph \mathcal{G} . Then

(i)
$$\cup_{i=1}^m \chi(\mathcal{G}_i) \subseteq \chi(\mathcal{G});$$

(ii) for any maximal CMS \mathcal{H} ,
$$\cup_{i=1}^m \chi(\mathcal{G}_i) = \chi(\mathcal{H}).$$

Proof. See Kim (2006).

For a given set of Markovian subgraphs, we can readily obtain the set of PSs under the decomposability assumption. By (1), we can find $\chi(\mathcal{G})$ for any decomposable graph \mathcal{G} simply by taking all the intersections of the cliques of the graph. An apparent feature of a maximal CMS in contrast to a CMS is stated in Theorem 4.3. Note that, in this theorem, \mathcal{G} is a CMS of $\mathcal{G}_i, i = 1, 2, \dots, m$.

Another important merit of a PS is that if a set of nodes is a PS in a Markovian subgraph, then it is not intersected in any other Markovian subgraphs.

Theorem 4.4. Let \mathcal{G} be a decomposable graph and \mathcal{G}_1 and \mathcal{G}_2 be Markovian subgraphs of \mathcal{G} . Suppose that a set $C \in \chi(\mathcal{G}_1)$ and that $C \subseteq V_2$. Then C is not intersected in \mathcal{G}_2 by any other subset of V_2 .

Proof. Suppose that there are two nodes u and v in C that are separated in \mathcal{G}_2 by a set S . Then, by Theorem 4.2, we have $\langle u|S|v \rangle_{\mathcal{G}}$. Since $C \in \chi(\mathcal{G}_1)$ and \mathcal{G}_1 is decomposable, C is an intersection of some neighboring cliques of \mathcal{G}_1 by equation (1). So, S can not be a subset of V_1 but a proper subset of S can be. This means that there are at least one pair of nodes, v_1 and v_2 , in \mathcal{G}_1 such that all the paths between the two nodes are intersected by C in \mathcal{G}_1 , with v_1 appearing in one of the neighboring cliques and v_2 in another.

Since v_1 and v_2 are in neighboring cliques, each node in C is on a path from v_1 to v_2 in \mathcal{G}_1 . From $\langle u|S|v \rangle_{\mathcal{G}}$, it follows that there is an l -cycle ($l \geq 4$) that passes through the nodes u, v, v_1 , and v_2 in \mathcal{G} . This contradicts the assumption that \mathcal{G} is decomposable. Therefore, there can not be such a separator S in \mathcal{G}_2 .

Among the above three theorems, Theorem 4.3 plays a key role in the method of model-combination and the other two are employed in adding and removing edges during the combining process.

5. Graph of prime separators

In this section, we will introduce a graph of PSs which consists of PSs and edges connecting them. The graph is the same as the undirected graphs that are considered so far in this paper, the nodes being replaced with PSs. Given a decomposable graph \mathcal{G} , the graph of the PSs of \mathcal{G} is defined as follows:

Let $A = \cup_{a \in \chi(\mathcal{G})} a$. Then the graph of the prime separators (GOPS for short) of \mathcal{G} is obtained from \mathcal{G}_A by replacing every PS and all the edges between every pair of neighboring PSs in \mathcal{G}_A with a node and an edge, respectively.

For example, there are three PSs, $\{3, 4\}$, $\{3, 5\}$, and $\{4, 8\}$, in graph \mathcal{G}_1 in Figure 8. Then none of the PSs is conditionally independent of any other among the three PSs. We represent this phenomenon with the graph at the top-left corner in Figure 9, where the GOPS's are the graphs of the line (as against dotted) ovals only. The xGOPS's (short for "expanded GOPS") as appearing in the figure are defined in Section 6 and used in model combining.

We can see conditional independence among the PSs, $\{13, 14\}$, $\{10, 13\}$, $\{10, 19\}$, and $\{10, 21\}$, in graph \mathcal{G}_3 in Figure 8. This conditional independence is depicted in GOPS₃ in Figure 9. As connoted in GOPS₁ in Figure 9, a GOPS may contain a clique of more than 2 PSs, but it cannot contain a cycle of length 4 or larger if the PSs are from a decomposable graph.

Let \mathcal{G} be a Markovian subgraph of \mathcal{G} and suppose that, for three PSs, A, B , and C , of \mathcal{G} , $A \setminus C$ and $B \setminus C$ are separated by C in \mathcal{G} . Then, by Theorem 4.2, the same is true in \mathcal{G} .

For three sets, A, B , and C , of PSs of a graph \mathcal{G} , if A and B are separated by C , then we have that

$$(\cup_{a \in A} a) \cap (\cup_{b \in B} b) \subseteq (\cup_{c \in C} c) \tag{4}$$

When A,B, and C are all singletons of PSs, the set-inclusion is expressed as

$$A \cap B \subseteq C \quad (5)$$

This is analogous to the set-inclusion relationship among cliques in a junction tree of a decomposable graph (Lauritzen (1996)). A junction tree is a tree-like graph of cliques and intersection of them, where the intersection of neighboring cliques lies on the path which connects the neighboring cliques. As for a junction tree, the sets in (5) are either cliques or intersection of cliques. In the context of a junction tree, the property as expressed in (5) is called the junction property. We will call the property expressed in (4) *PS junction* property, where ‘PS’ is from ‘prime separator.’

The GOPS and the junction tree are different in the following two senses: First, the basic elements are PSs in the GOPS while they are cliques in the junction tree; secondly, the GOPS is an undirected graph of PSs while the junction tree is a tree-like graph of cliques. Some PSs may form a clique in an undirected graph as in graphs \mathcal{G}_1 and \mathcal{G}_4 in Figure 8. This is why GOPS may not necessarily be tree-like graphs. So, two PSs may be separated by a set of PSs. But, since all the PSs in a decomposable graph \mathcal{G} are obtained from the intersections of neighboring cliques in \mathcal{G} , the GOPS of \mathcal{G} is the same as the junction tree of \mathcal{G} with the clique-nodes removed from the junction tree. Whether \mathcal{G} is decomposable or not, expression (4) holds in general.

6. Description of model-combining procedure

We will call a node a *PS node* if it is contained in a PS, and a *non-PS node* otherwise. Theorem 4.4 implies that if, for a given Markovian subgraph \mathcal{G}' , s is the set of the PSs each of which is a neighbor to a PS node v in \mathcal{G}' , then s will also be the set of the neighboring PSs of any PS, say a , such that $v \in a$, in the Markovian subgraph which is obtained by adding the PS, a , to \mathcal{G}' . This is useful in locating PSs for model-combination since PS nodes of a PS always form a complete subgraph.

Other useful nodes in model-combination are the non-PS nodes that are shared by multiple Markovian subgraphs. A simple illustration of the usefulness is given in expression (2). The Markovian subgraphs in Figure 1 share node 1, which determines the meeting points of the subgraphs when they are combined into the maximal CMS, [12][234]. Whether they are PS nodes or not, a set of nodes which are shared by a pair of Markovian subgraphs become meeting points of the subgraphs in the combining process. The shared nodes restrict the possible locations of the PS nodes that are not shared by both of the subgraphs. We will call by *xGOPS* a GOPS which is expanded with the nodes that are shared with other subgraphs. However we will not distinguish the two and use the terminology “GOPS” when confusion is not likely.

A rule of thumb of model-combination is that we connect two nodes each from different Markovian subgraphs in a given set, say \mathcal{M} , of Markovian subgraphs if the two nodes are not separated by any other nodes in \mathcal{M} . We will formally describe this condition below:

[Separateness condition] Let \mathcal{M} be a set of Markovian subgraphs of \mathcal{G} and \mathcal{H} a maximal CMS of \mathcal{M} . If two nodes are in a graph in \mathcal{M} and they are not adjacent in the graph, then neither are they in \mathcal{H} . Otherwise, adjacency of the nodes in \mathcal{H} is determined by checking separateness of the nodes in \mathcal{M} .

Suppose that \mathcal{M} consists of m Markovian subgraphs, $\mathcal{G}_1, \dots, \mathcal{G}_m$, of \mathcal{G} and we denote by a^i a PS of \mathcal{G}_i . We can then combine the models of \mathcal{M} as follows.

Step 1. We arrange the subgraphs into $\mathcal{G}_{i_1}, \dots, \mathcal{G}_{i_m}$ such that $|V_{i_j} \cap V_{i_{j+1}}| \geq |V_{i_{j+1}} \cap V_{i_{j+2}}|$ for $j = 1, 2, \dots, m - 2$. For convenience, let $i_j = j, j = 1, 2, \dots, m$. We set $\eta_1 = \{\mathcal{G}_1\}$.

Step 2a. We first put an edge between every pair of PSs, a^1 and a^2 , if

$$a^1 \cap a^2 \neq \emptyset,$$

in such a way that the separateness condition is satisfied with regard to \mathcal{M} . We denote the resulting GOPS by H .

Step 2b. Once the node-sharing PSs are all considered in Step 2a, we need to consider all the PSs a^1 and a^2 such that

$$a^1 \cap (\cup_{a \in \chi(\mathcal{G}_2)} a) = \emptyset \text{ and } a^2 \cap (\cup_{a \in \chi(\mathcal{G}_1)} a) = \emptyset \tag{6}$$

and put edges between $a^i, i = 1, 2$, and every PS in \mathcal{G}_{3-i} that is acceptable under the separateness condition, in addition to the GOPS which is obtained in Step 2a. For example, for each a^1 satisfying (6), we add edges to \mathcal{H} between the a^1 and every possible PS in \mathcal{G}_2 under the separateness condition, and similarly for each of a^2 that satisfy (6). We denote the result of the combination by η_2 .

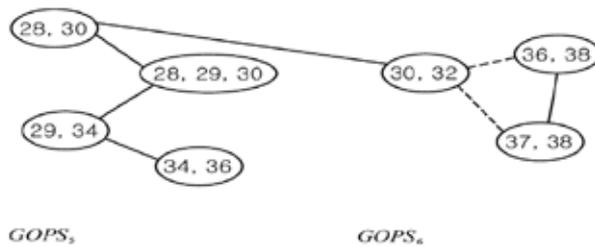


Fig. 2. A graphic display of part of Step 2a corresponding to that the PS of $GOPS_5, \{28, 30\}$, and the PS of $GOPS_6, \{30, 32\}$, share node 30 and that $\{28, 30\}$ is adjacent to $\{29, 31, 32, 34\}$ and separated from $\{35, 36, 37, 38\}$ by $\{29, 31, 32, 34\}$. The non-adjacent connectedness is expressed by dashed lines.



Fig. 3. Step 2a in progress from Figure 2 as for the PS pairs, {28, 29, 30} and {30, 32} and {34, 36} and {36, 38}.

Step 3. Let η_i be the GOPS obtained from the preceding step. Note that η_i can be a set of GOPS's. For each GOPS \mathcal{H} in η_i , we combine \mathcal{H} with \mathcal{G}_{i+1} as in Step 2, where we replace \mathcal{G}_1 and \mathcal{G}_2 with \mathcal{H} and \mathcal{G}_{i+1} , respectively. We repeat this combination with \mathcal{G}_{i+1} for all the graphs \mathcal{H} in η_i , which results in the set, η_{i+1} , of newly combined graphs.

Step 4. If $i + 1 = m$, then stop the process. Otherwise, repeat Step 3.

We will call this process Markovian combination of model structures or MCMoSt for short. The process is summarized in flowcharts in Figures 5 and 6; the former is of the main body of the process and the latter is of checking for the separateness condition. For a brief illustration of the MCMoSt, we will consider the two marginal graphs, \mathcal{G}_5 and \mathcal{G}_6 in Figure 8. This example has only two graphs, so we may skip Step 1.

Figure 9 shows the GOPS_s of two marginal graphs \mathcal{G}_5 and \mathcal{G}_6 . As for \mathcal{G}_5 , the set of PS_s in GOPS₁ is {{28, 30}, {28, 29, 30}, {29, 34}, {34, 36}} and it is {{30, 32}, {36, 38}, {37, 38}} for \mathcal{G}_6 . The PS of GOPS₅, {28, 30}, and the PS of GOPS₆, {30, 32}, share node 30. So we put an edge between the two PS's. In \mathcal{G}_5 , {28, 30} is adjacent to {29, 31, 32, 34} and is separated from {35, 36, 37, 38} by {29, 31, 32, 34}. This separateness must be preserved, by Theorem 4.2, in the combined model of \mathcal{G}_5 and \mathcal{G}_6 . We represent this non-adjacent connectedness by dashed lines in Figure 2.

The other PSs that share nodes between \mathcal{G}_5 and \mathcal{G}_6 are the pair of {28, 29, 30} and {30, 32} and the pair of {34, 36} and {36, 38}. We put edges between the PSs in each of these pairs and then check the separateness condition. In \mathcal{G}_5 , {37, 38} is separated from {28, 29, 30} by {31, 32, 34, 35, 36}, which is satisfied in the graph in Figure 3. This is the result of Step 2a.

In Step 2b, we can see that the PS, {37, 38}, of \mathcal{G}_6 is disjoint with all the PS's of \mathcal{G}_5 . In \mathcal{G}_5 , we see that {34, 36} separates {37, 38} from the remaining six nodes in \mathcal{G}_5 . Thus we put an edge between {34, 36} and {37, 38} only. This ends up with the combined GOPS in Figure 4.

In combining a pair of graphs, \mathcal{G}_1 and \mathcal{G}_2 say, suppose that an edge is added between a PS, a^1 , in \mathcal{G}_1 and another PS, a^2 , in \mathcal{G}_2 and let $N_i, i = 1, 2$, be the set of the PSs which are adjacent to a^i in \mathcal{G}_i . Then, under the decomposability assumption and the separateness condition, further edge-additions are possible between the PSs in the $(\{a^i\} \cup N_i)$'s only.

An example of this is given in Section 7.

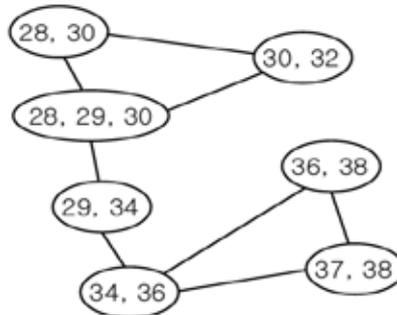


Fig. 4. Step 2b as continued from Figure 3.

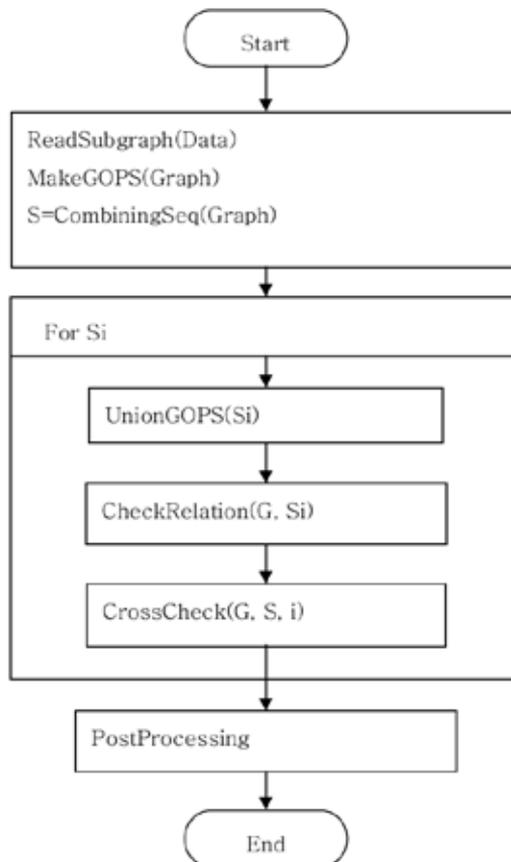


Fig. 5. A flowchart of the model-combining process, MCMoSt. In this chart, S is a sequence of marginal models to be combined; *UnionGOPS* just puts the two graphs to be combined together; *CheckRelation* checks if the separateness condition is satisfied between nodes and/or PSs; *CrossCheck* checks if the combined graph preserves the PSs of the two graphs.

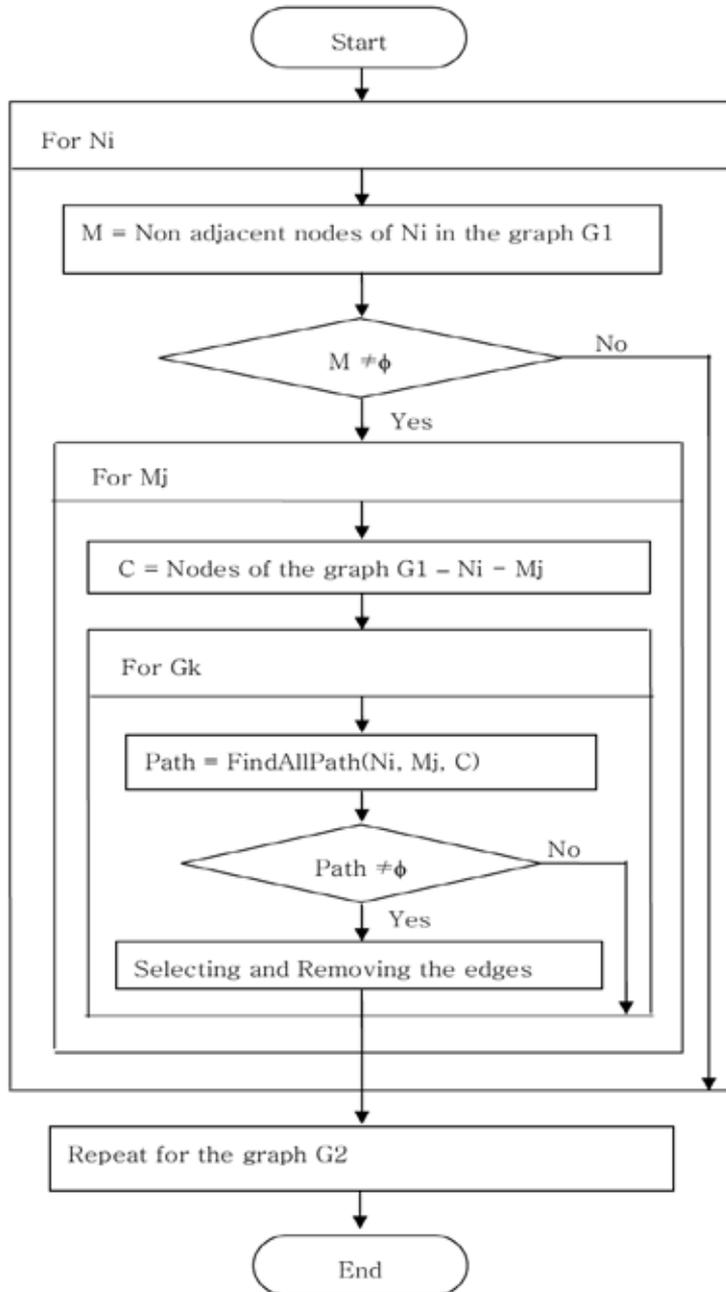
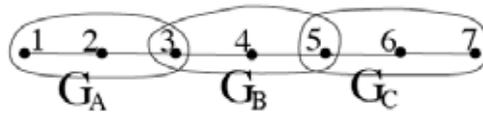


Fig. 6. A flowchart of the process *CheckRelation* which is a main part of MCMoSt. In this chart, we assume combining two graphs, G_1 and G_2 say. *FindAllPath(A, B, C)* finds paths between A and B that are blocked by C ; *Selecting and Removing the edges* means that, for each of the paths which are found in *FindAllPath*, the edges to be removed are selected and removed.

6.1 Time complexity of the procedure

Let $\mathcal{V} = \{V_1, V_2, \dots, V_m\}$. For a given set of \mathcal{G}_A 's, $A \in \mathcal{V}$, we denote by $E^s(\mathcal{V})$ the set of the pairs, u and v , for which there is at least one \mathcal{G}_A such that $\{u, v\} \subseteq A$ and they are not adjacent in \mathcal{G}_A , denote by $E^a(\mathcal{V})$ the set of the pairs, u and v , for which there is at least one \mathcal{G}_A such that $\{u, v\} \subseteq A$ and they are adjacent in \mathcal{G}_A , and let $E^{rem}(\mathcal{V}) = \{\{u, v\} \subseteq V; u \neq v\} \setminus (E^s(\mathcal{V}) \cup E^a(\mathcal{V}))$. For example, in the graph below, $V = \{1, 2, \dots, 7\}$, $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, $C = \{5, 6, 7\}$, $\mathcal{V} = \{A, B, C\}$, $E^s(\mathcal{V}) = \{\{1, 3\}, \{3, 5\}, \{5, 7\}\}$, $E^a(\mathcal{V}) = \{\{i, i + 1\}, i = 1, 2, \dots, 6\}$ and $E^{rem}(\mathcal{V}) = \{\{i, j\}, 1 \leq i < j \leq 7\} \setminus (E^s(\mathcal{V}) \cup E^a(\mathcal{V}))$.



The computing time of MCMoSt depends upon the sizes of the sets such as E^a and E^{rem} of the graphs in \mathcal{M} . A main part of the algorithm is designed for searching for all the possible edges between nodes under the condition that the pairs of nodes in E^s are separated. We use the depth-first search method (Tarjan, 1972) in Step 2 of the combination process to check the separateness between nodes. Suppose we combine \mathcal{G}_1 and \mathcal{G}_2 into a graph \mathcal{H} and obtain E^s , E^a and E^{rem} from \mathcal{G}_1 and \mathcal{G}_2 . Then we search for all the possible edges between nodes in such a way that, if there is a path, π' , in \mathcal{G}_1 or \mathcal{G}_2 which contains u and v on itself and there is a path, π , in \mathcal{H} which also contains u and v on itself, then π' is a Markovian subpath of π .

For two graphs, \mathcal{G}_1 and \mathcal{G}_2 , let $|V_i| = n_i$ with $i = 1, 2$, $|V_1 \cap V_2| = n_{12}$ and $\tilde{n}_i = n_i - n_{12}$. It is well known that the time complexity of the depth-first search method for a graph $\mathcal{G} = (V, E)$ is of order $O(|V| + |E|)$. So the time complexity for the combination is of order $\tilde{n}_1 \tilde{n}_1^2 O(\tilde{n}_2 + \tilde{e}_2) + \tilde{n}_2^2 O(\tilde{n}_1 + \tilde{e}_1)$ where \tilde{e}_i is the number of edges in the induced subgraph of \mathcal{G}_i on $V_i \setminus V_{3-i}$. As a matter of fact, when we use GOPS's instead of graphs of nodes, the time complexity reduces by a considerable amount. For instance, we can see in Figure 9 that the six GOPS's are composed of 3, 3, 5, 5, 6, 3 PS's, respectively, while the marginal graphs are of ten nodes each. MCMoSt uses PS's and the nodes that are shared between graphs rather than nodes only.

7. Illustration

In this section, we suppose that we are given six marginal models as in Figure 8 each of which is Markovian subgraphs of the graph in Figure 7. As a matter of fact the six marginal models were obtained through a statistical analysis. We first generated data from the model in Figure 7 assuming that all the 40 variables are binary. We then chose six subsets of variables in such a way that the variables are more highly associated within subsets than between them. The six marginal models in Figure 8 were obtained through a statistical analysis of contingency table data. A detailed description of this is given in Kim (2005).

Since our interest is in the model-combining method, we will refrain from any further discussion on this statistical issue.

For notational convenience, we will denote a PS by $c(\cdot)$. For instance, $c(1, 2)$ denotes a PS consisting of nodes 1 and 2. From \mathcal{G}_1 , we have $\chi(\mathcal{G}_1) = \{c(3, 4), c(4, 8), c(3, 5)\}$. If we regard the three PSs as random variables, these PSs are associated. In the same context, we can represent the conditional independence relationship among the PSs via an independence graph based on the corresponding marginal models \mathcal{G}_i . The GOPS's are displayed for each marginal model in Figure 9 along with the nodes which are shared among the marginal models. We will call a variable whose corresponding node is a PS-node a PS variable and similarly for a non-PS variable. Since every non-PS variable is

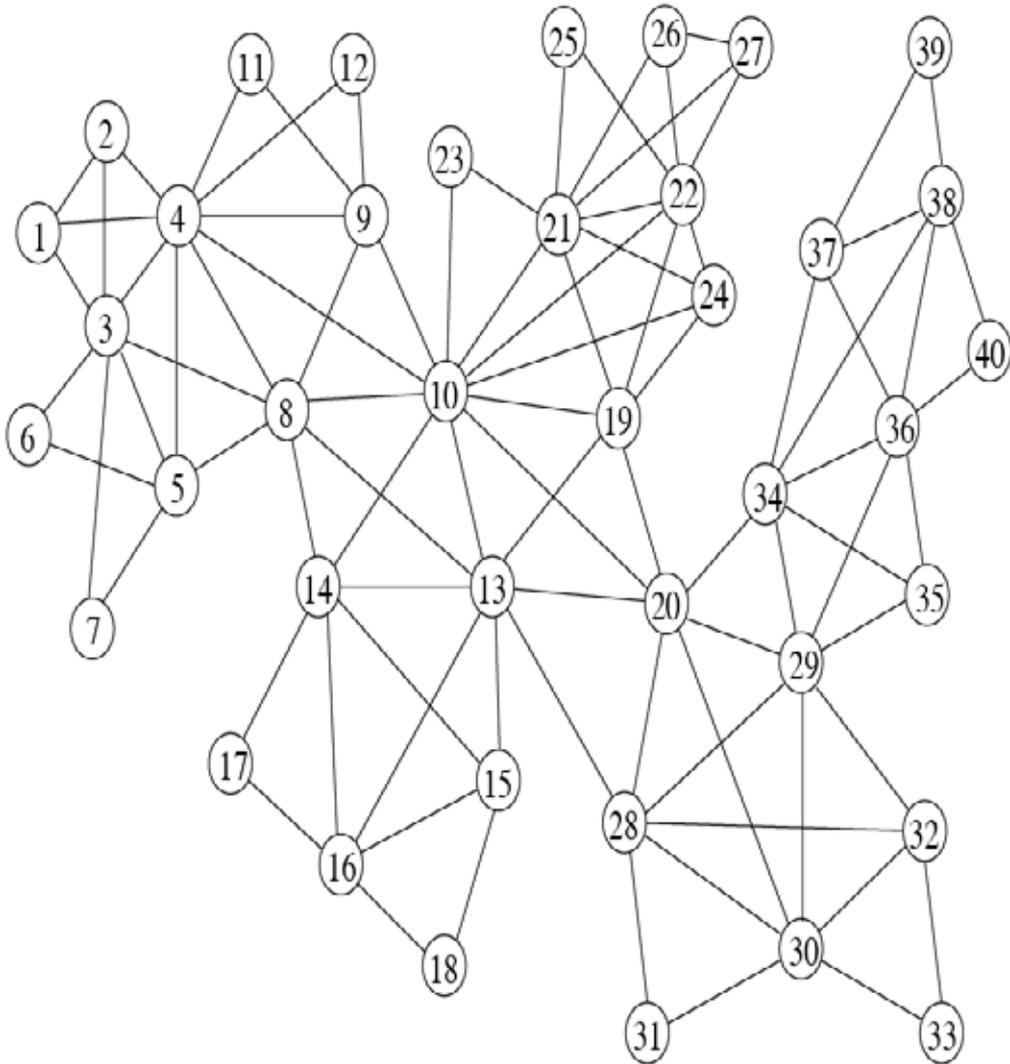


Fig. 7. A model of the 40 variables that are used for illustration

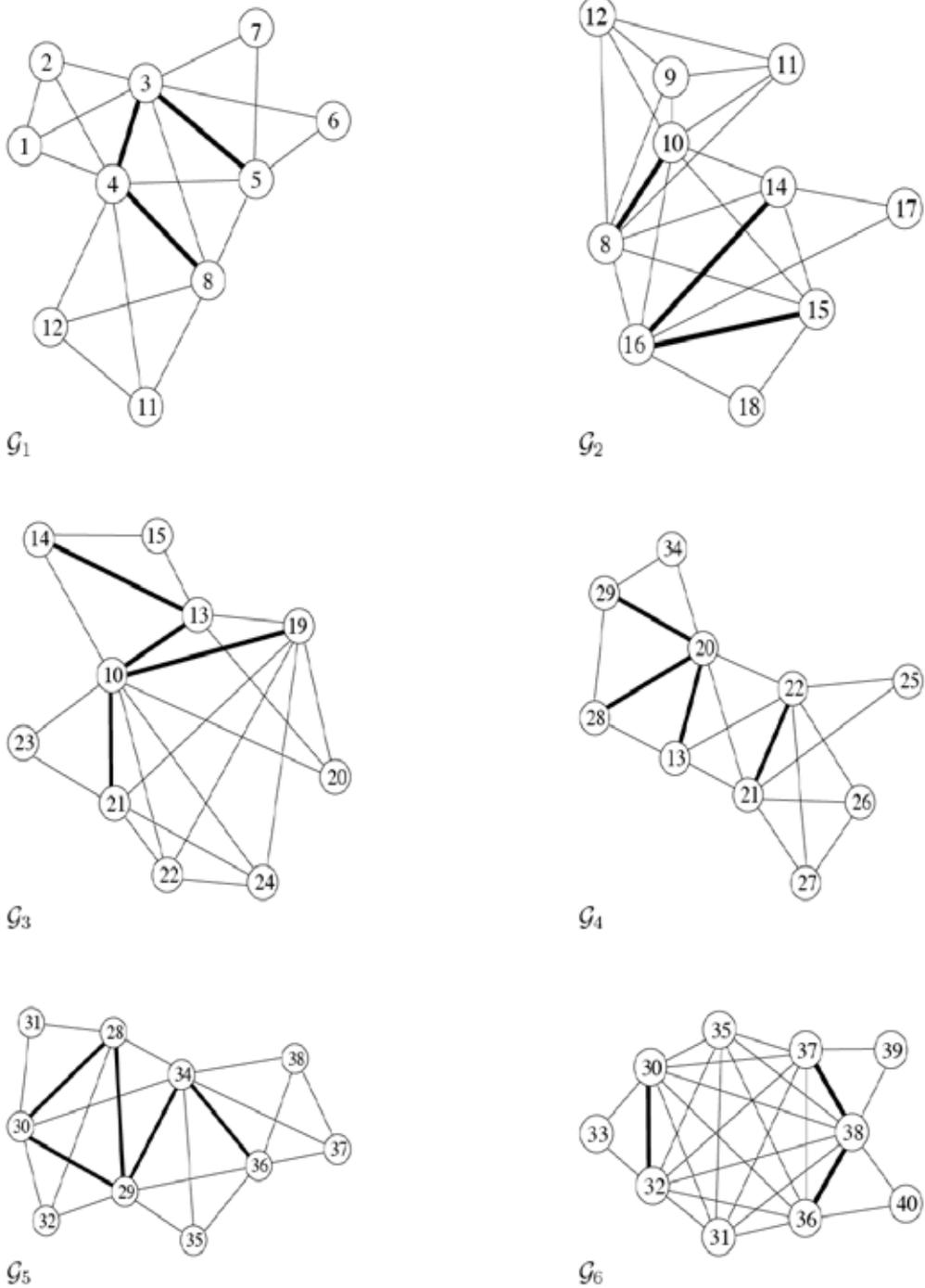


Fig. 8. Six marginal models of the model in Figure 7. PSs are represented by thick lines. See Figure 9 for the PSs of the six marginal models.

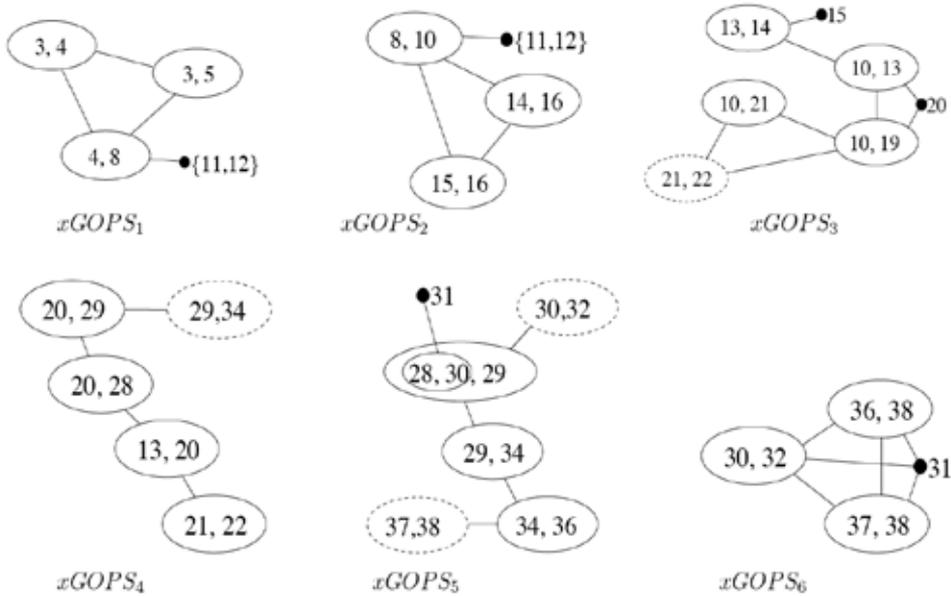


Fig. 9. The GOPS's and $xGOPS$'s of the six marginal models in Figure 8. $GOPS_i$'s are the graphs of the line ovals only. $xGOPS_i$ is the independence graph of the PSs of \mathcal{G}_i and the nodes which are shared by \mathcal{G}_i with other marginal models. The oval nested in another oval in $xGOPS_5$ means that the PS, $c(28, 30)$, is a subset of the PS, $c(28, 29, 30)$. $c(28, 30)$ is a neighbor of node 31 in the graph. Dotted ovals mean that the corresponding set of nodes is a PS in some other marginal models.

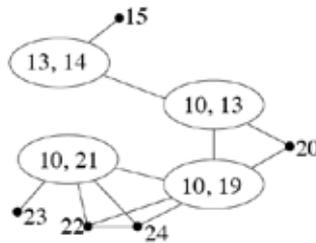


Fig. 10. The graph obtained by linking non-PS variables (bullets) to the PS's of \mathcal{G}_3 in Figure 8.

Separated from other variables by its neighbor PSs, we can represent \mathcal{G}_i by linking each non-PS variable to its neighbor PSs. For example, the graph in Figure 10, which is obtained by adding non-PS variables of \mathcal{G}_3 to the graph, $GOPS_3$, of the PSs of \mathcal{G}_3 . Since every non-PS node has a unique set of neighboring PSs, a graph such as that in Figure 10 is determined uniquely.

According to Theorem 4.3 (i), all the PSs that appear in a marginal model of \mathcal{G} are found in $\chi(\mathcal{G})$. This means that we must make sure that all the PSs of the marginal models appear as PSs in \mathcal{G} . This fact is instrumental to constructing the independence graph of the PSs of the marginal models.

In section 3, we considered a simple problem of model combination. Although the example is very simple, we can see therein that the shared variables, X_1 and X_2 , are like road signs in constructing a model structure of the variables that are involved in the variable-sharing marginal models. As more variables are shared between a pair of marginal models, the possible locations of each variable are more limited and thus model construction for the variables that are involved in either of the two marginal models becomes easier. The variable-sharing between V_i and V_{i+1} , $i = 1, 2, \dots, 5$, is as follows:
 $|V_1 \cap V_2| = 3$, $|V_2 \cap V_3| = 3$, $|V_3 \cap V_4| = 4$, $|V_4 \cap V_5| = 3$, $|V_5 \cap V_6| = 7$.
 $|V_i \cap V_j| = 0$ when $|i - j| > 1$. So, it is desirable that we begin combining marginal models from the pair of \mathcal{G}_5 and \mathcal{G}_6 ,

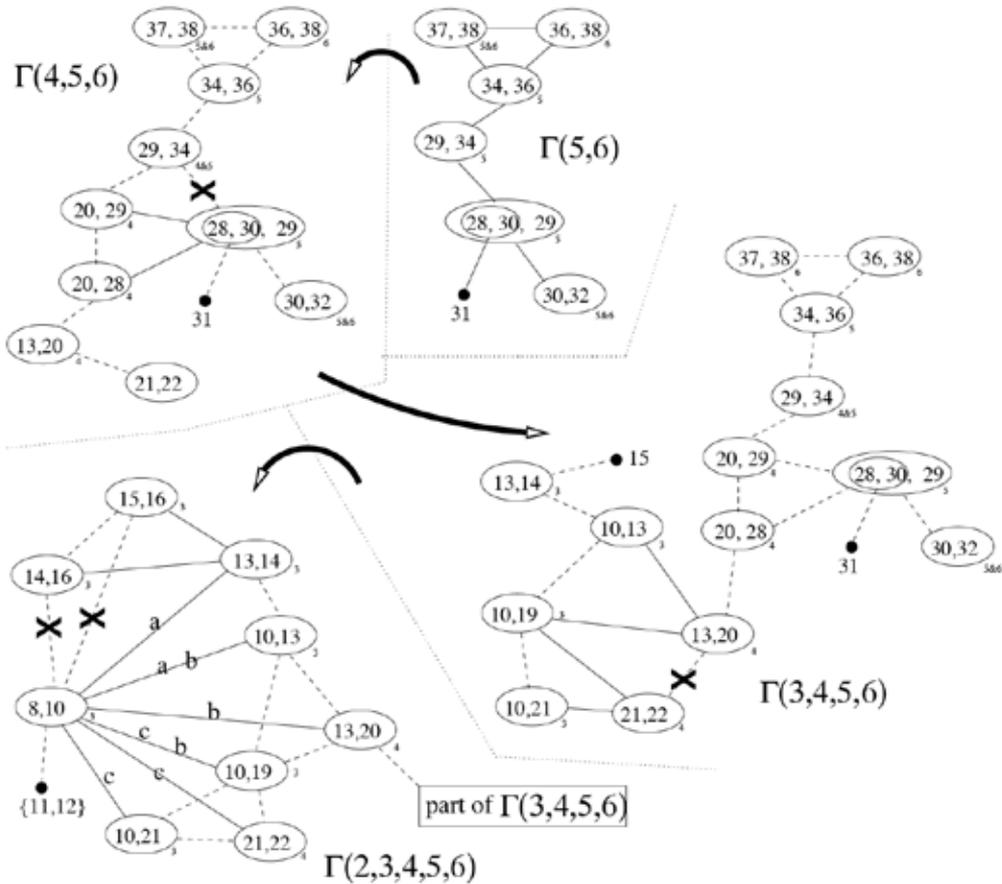


Fig. 11. A model-combining process of marginal models \mathcal{G}_i . $\Gamma(A)$ denotes a maximal independence graph of the PSs of \mathcal{G}_i , $i \in A$. The small numbers at the bottom-right of the ovals are the marginal model labels to which the corresponding PSs belong. $\Gamma(2, 3, 4, 5, 6)$ is of three α GOPS's which are determined by three different groups of edges that are labelled by a, b, and c at the bottom-left corner. Line edges are used when they are newly added; dotted edges for existing edges; and X-marked dotted edges for the existing edges to be removed.

and then keep combining marginal models in the order of $\mathcal{G}_4, \mathcal{G}_3, \mathcal{G}_2, \mathcal{G}_1$.

Figure 11 shows part of the model-combining process of the xGOPS's. The graph in the top-right is the result of combining \mathcal{G}_5 and \mathcal{G}_6 , which is the same as the graph in Figure 3.

For a given collection of the GOPSs of $\mathcal{G}_i, i \in A$, there can be more than one maximal independence graph of the PSs, and we will denote the set of maximal independence graphs by $\Gamma(A)$. When $|\Gamma(A)| = 1$, the independence graph itself will be represented by $\Gamma(A)$.

For notational convenience, we will use the symbol \oplus to denote an outcome of model-combination. For example, we write $\mathcal{G} \oplus \mathcal{G}'$ to express the outcome of combining the two graphs, \mathcal{G} and \mathcal{G}' . Since $\Gamma(5, 6)$ is a single graph, we may express the next combination by $\Gamma(5, 6) \oplus \text{xGOPS}_4$. By applying the same method for the combination, we also obtain a unique graph as in the top-left corner of Figure 11, and denote it by $\Gamma(4, 5, 6)$. Note that the PS, $c(29, 34)$, is shared by both of $\Gamma(5, 6)$ and xGOPS_4 . Since $c(20, 28)$ and $c(20, 29)$ share nodes with $c(28, 29, 30)$, we put edges between each of the former two PSs and the last PS. This edge addition conflicts with the separateness of $c(29, 24)$ from $c(20, 28)$. Considering that node 28 is shared by $c(20, 28)$ and $c(28, 29, 30)$ only (since $c(28, 30)$ is contained in $c(28, 29, 30)$, it is ignored here), we can see that the edge between $c(29, 34)$ and $c(28, 29, 30)$ must be deleted. Note that the three PSs, $c(29, 34)$, $c(20, 29)$, and $c(28, 29, 30)$ are on a path and satisfy the PS junction property (4).

$\Gamma(4, 5, 6) \oplus \text{xGOPS}_3 = \Gamma(3, 4, 5, 6)$ is also a single graph as in Figure 11. Since neither of $c(10, 21) \cap c(21, 22)$ and $c(10, 13) \cap c(13, 20)$ is empty, line edges are put between the node-sharing PSs. Then the separateness condition is violated since $c(10, 21)$ and $c(10, 13)$ are no longer separated by $c(10, 19)$, and the three PSs, $c(10, 13)$, $c(10, 19)$, and $c(10, 21)$, share node 10. Thus the edge between $c(21, 22)$ and $c(13, 20)$ is deleted. After this, we check if there are any PSs pertaining to Stage 2 of section 6, which end up with the addition of edges between $c(10, 19)$ and $c(21, 22)$ and between $c(10, 19)$ and $c(13, 20)$.

So far the combination process has produced single graphs. But $\Gamma(3, 4, 5, 6) \oplus \text{xGOPS}_2$ is a set of three graphs. $c(14, 16)$ and $c(13, 14)$ share node 14, and so we put a line edge between them. $c(8, 10)$ shares node 10 with

$$c(10, 13), c(10, 19), \text{ and } c(10, 21), \quad (7)$$

so we can put two line edges, between $c(8, 10)$ and each of the first two of the PSs in (7), and another set of two line edges, between $c(8, 10)$ and each of the last two of the PSs in (7). Note that any of these edge additions violates the separateness of $c(13, 14)$ from $c(10, 19)$ and $c(10, 21)$. So the edges between $c(8, 10)$ and $c(14, 16)$ and between $c(8, 10)$ and $c(15, 16)$ are deleted. Implementing Stage 2 then ends up with the addition of edges between $c(15, 16)$ and $c(13, 14)$ and possibly between $c(8, 10)$ and $c(13, 14)$, where the latter edge can be added along with the edge between $c(8, 10)$ and $c(10, 13)$ out of the three edges between $c(8, 10)$ and each of the PSs in (7). In other words, $c(8, 10)$ can form a clique with $c(10, 13)$ and $c(13, 14)$, with $c(10, 13)$, $c(13, 20)$, and $c(10, 19)$, or with $c(10, 19)$, $c(10, 21)$, and $c(21, 22)$ in an xGOPS in $\Gamma(3, 4, 5, 6) \oplus \text{xGOPS}_2$. This is depicted at the bottom-left corner of Figure 11.

However, the set $c(8, 10) \cup c(13, 14) \cup c(10, 13) \cup c(10, 19) \cup c(10, 21) = \{8, 10, 13, 14, 19, 21\}$ shares node 8 with V_1 , nodes 8, 10, 14 with V_2 , nodes 10, 13, 14 with V_3 , and nodes

13, 21 with V_4 . So it is more likely that $c(8, 10)$ and $c(13, 14)$ belong to the same clique. This is because the grouping of the variables was made so that the variables are more highly associated with each other within the subsets than between them. Based on this observation, we chose the xGOPS in which $c(8, 10)$ forms a clique with $c(10, 13)$ and $c(13, 14)$. We denote this graph by $\Gamma^a(2, 3, 4, 5, 6)$.

We can apply the same argument in combining $\Gamma^a(2, 3, 4, 5, 6)$ with \mathcal{G}_1 , which ends up with the GOPS in Figure 12. The bullets in the figure represent the non-PS variables, which are connected to their neighboring PSs. Note that those neighboring PSs are classified as such mostly due to the non-PS variables. For example, $\{37, 38\}$ is a PS separating node 39 from $V_6 \setminus \{37, 38, 39\}$ in \mathcal{G}_6 .

A PS is itself a complete subgraph and so is a clique of PSs. So we can easily transform the graph in Figure 12 into the undirected graph in Figure 13. This is the maximal CMS of the six marginal models as listed in Figure 8. The model in Figure 7 is fully recovered in the maximal CMS except the 5 thick edges appearing in Figure 13. These additional edges were created because X_4 were missing in \mathcal{G}_2 . If X_4 had been added to \mathcal{G}_2 , then $X_{\{4,9\}}$ would have separated X_{11} , X_{12} , and $X_{\{8,10\}}$ from each other, making those additional edges unnecessary. This phenomenon of additional edges leads

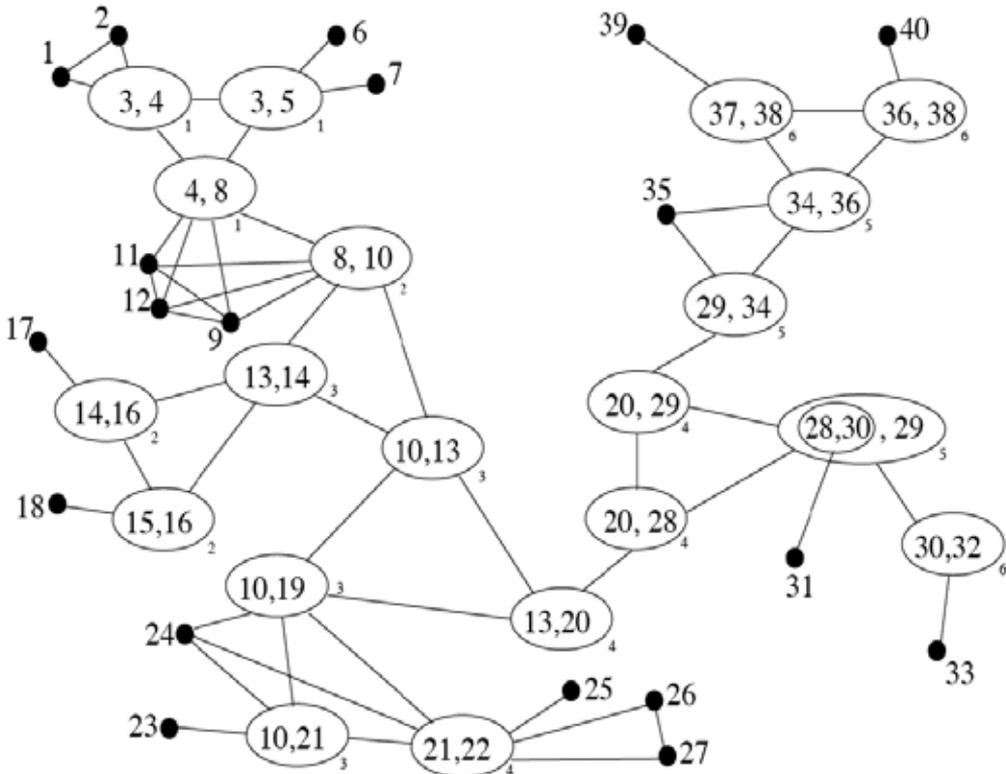


Fig. 12. An independence graph of PSs and non-PS variables. The PSs are in ovals and the dots are for the non-PS variables, and the small numbers at the bottom-right of the ovals are the marginal model labels of which the ovals are PSs.

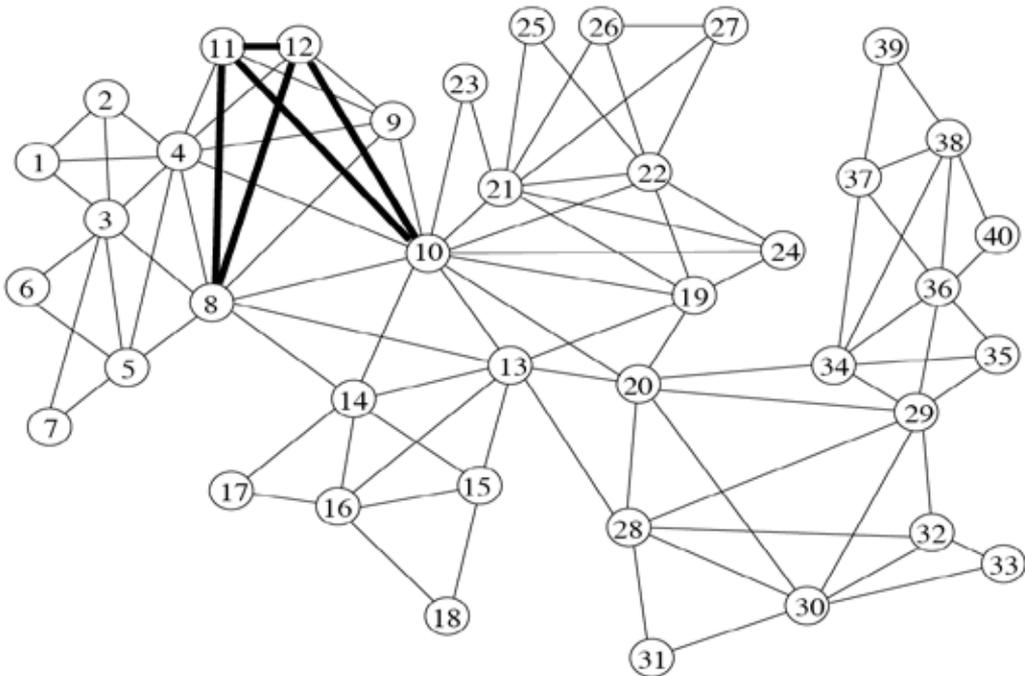


Fig. 13. The combined model structure which is obtained from the independence graph in Figure 12. The thick edges are additional to the model in Figure 7. us to recommend that the variables be grouped into marginal models so that the association between variables is higher within a marginal model than between marginal models.

8. Conclusion

In Section 7, we considered a model, \mathcal{G} , of 40 variables and six marginal models of it. The marginal models have their model structures given in decomposable graphs which are actually Markovian subgraphs of the graph \mathcal{G} . In this context, marginal model and Markovian subgraph may be regarded as synonyms as long as the joint model has a model structure which can be represented via an undirected graph.

In combining marginal models, it is important to make use of the locations of the variables that are shared by the marginal models to be combined. While we use GOPS's of marginal models to construct another GOPS, the locations of the non-PS nodes that are shared by the marginal models to be combined are as important as the PSs in the marginal models. The PS junction property (4) and the separateness condition are instrumental for locating PSs in model-combination. When $|\mathcal{G}_i \oplus \mathcal{G}_j| > 1$, that is, a multiple number of maximal CMS's are obtained, it is desirable that we look for or develop more marginal models from data in order to minimize the resultant maximal CMS's. For instance, the PS $\{8, 10\}$ of \mathcal{G}_2 in Figure 8 can be connected to the combined graph Γ (3, 4, 5, 6) by three different sets of edges as shown at the bottom-left corner of Figure 11. Denote the three different sets of corresponding nodes by A,B, and C. Then if we could develop a model for the set of

variables, $\{8, 10\} \cup A \cup B \cup C$, then the model would help us in choosing one of the three different sets of edges.

In selecting the subsets of variables, it is important that the variables that are highly associated belong to the same set. In other words, when the joint model is graphical with an undirected graph as its model structure, variables that appear as neighbors in the graph are desired to belong to a subset of variables. Otherwise, the model-combination may end up with an unnecessarily large graph. An example is demonstrated by thick edges in Figure 13. The thick edges would not have appeared, if X_9 had been included in \mathcal{G}_1 or X_4 in \mathcal{G}_2 . In this regard, subset selection for marginal modeling is crucial for a successful model-combination.

As mentioned in Section 1, several heuristic searching methods are developed for learning Bayesian networks from data. Since they deal with the whole set of variables involved in data, we can easily run into a sparse data problem for large scale modeling, not to mention the time complexity burden. The marginal-model based approach as proposed in this 15 paper may not suffer from the sparse data problem. Furthermore, in applying our method, the marginal models don't have to be based on observed data only. They may be based on expert opinions, since the method deals with model structures only.

Although the model combination is carried out under the decomposability assumption, we can deal with the marginal models of a graphical model, which are not decomposable, by transforming their model structures into decomposable (i.e., triangulated) graphs. The combined model will then be larger than expected as a trade-off of the graph triangulation made on the marginal models.

9. References

- Chickering, D. (1996). Learning Bayesian networks is NP-complete, In: *Learning from Data*, D. Fisher and H. Lenz (Ed.), 121-130, Springer-Verlag.
- Cooper, G.F.(1999). An overview of the representation and discovery of causal relationships using Bayesian networks, In: *Computation, Causation, and Discovery*, C. Glymour and G.F. Cooper (Ed.), 3-62, The MIT Press, Cambridge, MA.
- Cox, D.R. and Wermuth, N. (1999). Likelihood factorizations for mixed discrete and continuous variables, *Scand. J. Statist.*, 26, 209-220.
- Fienberg, S.E. (1980). *The Analysis of Cross-Classified Categorical Data*, The MIT Press, Cambridge, MA.
- Friedman, N. and Goldszmidt, M. (1998). Learning Bayesian networks with local structure, In: *Learning in Graphical Models*, M.I. Jordan, (Ed.), 421-460, Kluwer, Dordrecht, Netherlands.
- Heckerman, D., Geiger, D. and Chickering, D.M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data, *Machine Learning*, 20, 197-243.
- Kemeny, J.G., Speed, T.P., Knapp, A.W.(1976). *Denumerable Markov Chains*, 2nd edition, Springer, New York, NY.
- Kim, S.-H. (2004). Combining decomposable model-structures, Research Report 04-15, Division of Applied Mathematics, KAIST, Daejeon, S. Korea.
- Kim, S.-H. (2005). Log-linear modelling for contingency tables by using marginal model structures, *Research Report 05-12*, Division of Applied Mathematics, KAIST, Daejeon, S. Korea.

- Kim, S.-H. (2006). Properties of Markovian subgraphs of a decomposable graph, *Lecture Notes in AI*, 4293, 15-26.
- Lauritzen, S.L. (1996). *Graphical Models*. Oxford: Oxford University Press.
- Lauritzen, S.L. and Spiegelhalter, D.J. (1988). Local computations with probabilities on graphical structures and their application to expert systems, *J. R. Statist. Soc. B*, 50, 2, 157-224.
- Meek, C. (1995). Causal influence and causal explanation with background knowledge, In: *Uncertainty in Artificial Intelligence: Proceedings of the Eleventh Conference*, P. Besnard and S. Hanks (Ed.), 403-410, Morgan Kaufmann, San Mateo, CA.
- Neapolitan, R.E. (2004). *Learning Bayesian Networks*, Pearson Prentice Hall, Upper Saddle
- Neil, J.R., Wallace, C.S. and Korb, K.B. (1999). Learning Bayesian networks with restricted causal interactions, In: *Uncertainty in Artificial Intelligence*, K.B. Laskey and H. Prade, (Ed.), 486-493, Morgan Kaufmann Publishers, San Francisco, CA.
- Pearl, J. (1986). Fusion, propagation and structuring in belief networks, *Artificial Intelligence*, 29, 241-288.
- Pearl, J. (1988). *Probabilistic Reasoning In Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA.
- Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search*, 2nd ed. Cambridge, MA: MIT Press.
- Tarjan, R. E. (1972). Depth-first search and linear graph algorithms. *SIAM J. Comput.* 1(2), 146-160.
- Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*, Wiley, New York, NY.

Active Vibration Control of a Smart Beam by Using a Spatial Approach

Ömer Faruk KIRCALI^{1,2}, Yavuz YAMAN²,
Volkan NALBANTOĞLU² and Melin ŞAHİN²

¹*STM Savunma Teknolojileri Muhendislik ve Ticaret A.S., Ankara, TURKIYE*

²*Department of Aerospace Engineering, Middle East Technical University, Ankara,
TURKIYE*

1. Introduction

The vibration control is an important and rapidly developing field for lightweight flexible aerospace structures. Those structures may be damaged or become ineffective under the undesired vibrational loads they constantly experience. Hence, they require effective control mechanism to attenuate the vibration levels in order to preserve the structural integrity. The usage of smart materials, as actuators and/or sensors, has become promising research and application area that gives the opportunity to accomplish the reduction of vibration of flexible structures and proves to be an effective active control mechanism.

For the last few decades there has been an extensive research about the piezoelectric materials because of the capability of being used both as actuators and sensors. The smart structure is a structure that can sense external disturbance and respond to that in real time to fulfil operational requirements. Smart structures consist of passive structure, highly distributed active devices called smart materials/elements and processor networks. The smart materials are primarily used as sensors and/or actuators and are either embedded or attached to an existing passive structure (Çalışkan, 2002). Today, the main and maybe the most widespread application area of piezoelectric materials is using them as collocated actuator and sensor pair for active vibration control purposes (Prasad, 1998).

Active vibration control of a smart structure starts with an accurate mathematical model of the structure. Modeling smart structures may require the modeling of both passive structure and the active parts. Crawley and de Luis (1989), by neglecting the mass of active elements, presented an analytical modeling technique to show that the piezoelectric actuators can be used to suppress some modes of vibration of a cantilevered beam. Similar approach was carried out on thin plates by Dimitridis et al (1991). Although neglecting the mass and stiffness properties of the smart materials compared to the passive structure is generally acceptable, the modeling of a smart structure mainly involves the force and moment descriptions generated by the smart materials. Sample modeling studies are proposed by

several researchers such as Pota et al. (1992), Halim (2002b) The governing differential equations of motion of the smart structures can then be solved by analytical methods, such as modal analysis, assumed-modes method, Galerkin's method or finite element method (Meirovitch, 1986).

Since it is not so easy to consider all non-uniformities in structural properties of a smart structure, the analytical modeling techniques such as finite element model, modal analysis or assumed modes method allow one to obtain system model including only the approximate information of optimal placement of piezoelectric patches, natural frequencies and mode shapes of the structure except damping (Çalışkan, 2002 and Halim, 2002b). In order to improve the model, Nalbantoğlu (1998) and Nalbantoğlu et al. (2003) showed that experimental system identification techniques can be applied on flexible structures and they may help one to identify the system more accurately.

Due to having a large number of resonant modes, the high frequency characteristics of a flexible structure generally cause problems in identifying the system method. Since, usually the first few vibrational modes are taken into account in the controller design, the reduction of the model is often required to obtain the finite-dimensional system model (Hughes, 1981; Balas, 1995 and Moheimani, 1997). General approach for reducing the order of the model is the direct model reduction. However, removing the higher modes directly from the system model perturbs zeros of the system (Clark, 1997). Minimizing the effect of model reduction and correcting the system model is possible by adding a feedthrough, or correction, term including some of the removed modes, to the truncated model (Clark, 1997; Moheimani, 2000d and Moheimani, 2000c). Halim (2002b) proposed an optimal expression for feedthrough term in case of undamped and damped system models.

Various control techniques have been used as active control strategy like optimal control (Hanagoud, 1992), LQG control (Bai, 1996) and robust control using H_∞ (Nalbantoğlu, 1998; Yaman, 2001 and Ülker, 2003) or H_2 control framework (Halim, 2002c). The H_∞ control design technique for robust control phenomena has been developed by many researchers for various application areas including the vibration control (Zames, 1981; Francis, 1984; Doyle, 1989 and Lenz, 1993). Yaman et al. (2001, 2003b) showed the effect of H_∞ based controller on suppressing the vibrations of a smart beam due its first two flexural modes. They also extended their studies to a smart plate (2002a, 2002b). Ülker (2003) showed that, besides the H_∞ control technique, μ -synthesis based controllers can also be used to suppress vibrations of smart structures. In all those works on flexible structures, the general control strategy focused on analyzing the vibrations at specific locations over the structure and minimizing them. However, that kind of pointwise controller design ignores the effect of vibration at the rest of the body and a successful vibration reduction over entire structure can not always be accounted for.

Moheimani and Fu (1998c) introduced spatial H_2 norm, which is a measured performance over spatial domain, for spatially distributed systems in order to meet the need of spatial vibration control. Besides, Moheimani et al. (1997, 1998a) proposed spatial H_∞ norm

concept and simulation based results of spatial vibration control of a cantilevered beam were presented. Moheimani et al. (1998b, 1999) carried out the spatial approach on feedforward and feedback controller design, and presented illustrative results. They also showed that spatial H_∞ controllers could be obtained from standard H_∞ controller design techniques. Although the simulations demonstrated successful results on minimizing the vibrations over entire beam, implementation of that kind of controllers was not guaranteed on real world systems. Halim (2002b, 2002c) studied the implementation of spatial H_2 controllers on active vibration control of a simply-supported beam experimentally and presented successful results. He continued to work on simply-supported beams about implementation of spatial H_∞ controller and obtained successful experimental results (Halim, 2002a). Further experimental studies were performed on active vibration control of a simply-supported piezoelectric laminate plate by Lee (2005). Lee also attenuated acoustic noise due to structural vibration.

The current chapter aims to summarize the studies of modelling and spatial control of a cantilevered beam (Kırcalı et al. 2008, 2007, 2006a and 2006b).

2. Assumed-Modes Modeling of the Smart Beam

Consider the cantilevered smart beam model used in the study which is depicted in Fig.1. The smart beam consists of a passive aluminum beam (507mmx51mmx2mm) with eight symmetrically surface bonded SensorTech BM500 type PZT (Lead-Zirconate-Titanate) patches (25mmx20mmx0.5mm), which are used as the actuators. Note that, in this study, the group of PZT patches on one side of the beam is considered as if it is a single patch. The beginning and end locations of the PZT patches along the length of the beam away from the fixed end are denoted as r_1 and r_2 , and the patches are assumed to be optimally placed (Çalışkan, 2002). The subscripts b , p and sb indicate the passive beam, PZT patches and smart beam respectively. Analytical modeling of the smart beam is performed by assumed-modes method, which represents the deflection of the beam by means of a series solution:

$$y(r, t) = \sum_{i=1}^N \phi_i(r) q_i(t) \quad (1)$$

where ϕ_i are admissible functions which satisfy the geometric boundary conditions of the passive beam, q_i are time-dependent generalized coordinates, r is the longitudinal axis and t is time. Assumed-modes method uses this solution to obtain approximate system model of the structure with the help of energy expressions (Mason, 1981). The kinetic and potential energies of the smart beam can be determined to be (Kırcalı, 2006a):

$$T_{sb}(t) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \left\{ \int_0^{L_b} \rho_b A_b \phi_i \phi_j dr + 2 \int_{r_1}^{r_2} \rho_p A_p \phi_i \phi_j dr \right\} \dot{q}_i \dot{q}_j \quad (2)$$

$$V_{sb}(t) = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \left\{ \int_0^{L_b} E_b I_b \phi_i'' \phi_j'' dr + 2 \int_{r_1}^{r_2} E_p I_p \phi_i'' \phi_j'' dr \right\} q_i q_j \quad (3)$$

The total viscous damping force of the smart beam can similarly be obtained as (Kırcalı, 2006a):

$$F_{sb} = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \left\{ \int_0^{L_b} (2\xi_i \omega_i) \rho_b A_b \phi_i \phi_j dr + 2 \int_{r_1}^{r_2} (2\xi_i \omega_i) \rho_p A_p \phi_i \phi_j dr \right\} \dot{q}_i \dot{q}_j \quad (4)$$

where the beam's density, Young's modulus of elasticity, second moment of area and cross sectional area are defined as ρ_b , E_b , I_b , and A_b respectively. Also note that subscript i and j yield number of eigenvalues, ξ_i is the viscous damping coefficient of the i^{th} mode and ω_i represents the i^{th} natural frequency of the beam.

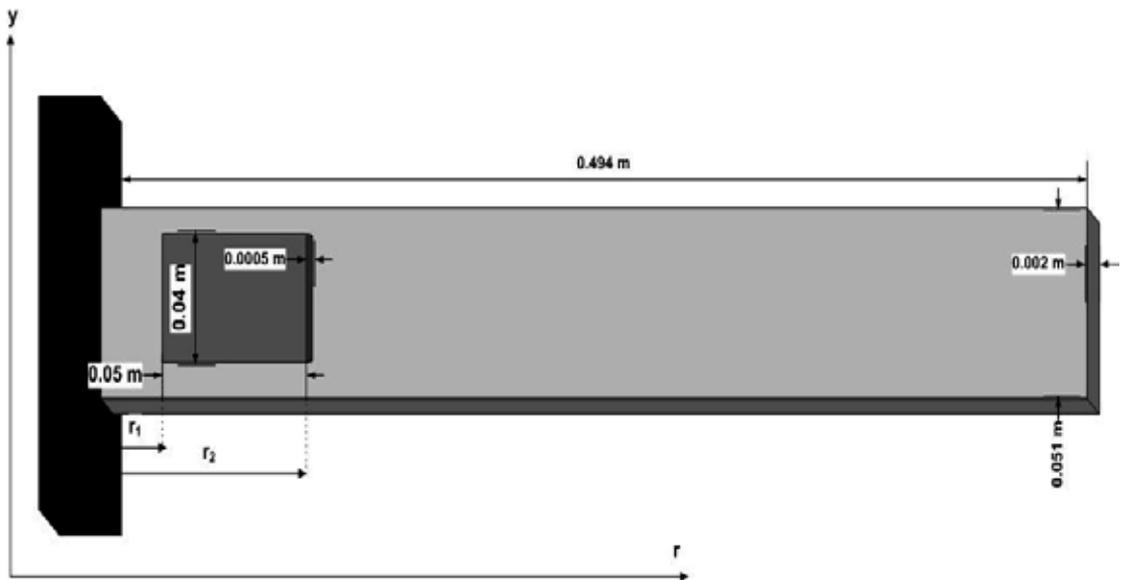


Fig. 1. The smart beam model used in the study

The PZT patches are placed in a collocated manner and the voltage is applied in order to create a bimorph configuration (PZT patches bonded to opposite faces of the beam have opposite polarity), the resulting effect on the beam becomes equivalent to that of a bending moment. This case is shown in Fig.2:

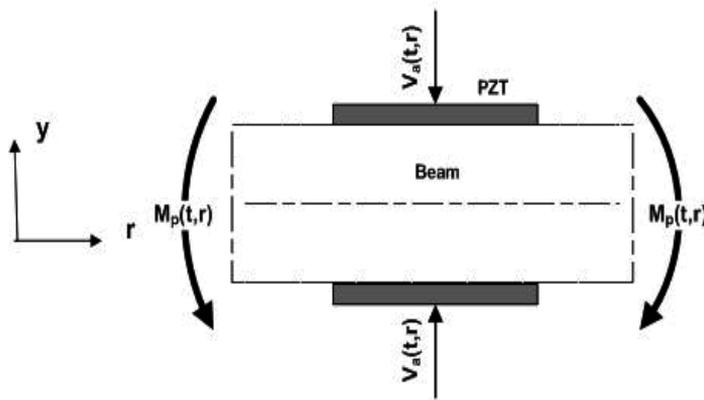


Fig. 2. Inducing bending moment by applying voltage to PZT patches

Here $M_p(t, r)$ denotes the bending moment and $V_a(t, r)$ is the applied voltage. When the voltage is applied on a PZT patch, a piezoelectric strain ε_p is introduced in the patch (Baz, 1988):

$$\varepsilon_p(t, r) = \frac{d_{31}}{t_p} V_a(t, r) \quad (5)$$

The longitudinal stress consequently generates a bending moment about the neutral axis of the system, as:

$$M_p(t, r) = C_p V_a(t, r) \quad (6)$$

where C_p is a geometric constant due to bending moment, and expressed as:

$$C_p = E_p d_{31} w_p (t_p + t_b) \quad (7)$$

As a consequence, the transfer function, $G_N(s, r)$, from the input voltage to the beam deflection in the frequency domain, including N number of eigenfunctions, is obtained as:

$$G_N(s, r) = \sum_{i=1}^N \frac{\bar{P}_i \phi_i(r)}{s^2 + 2\xi_i \omega_i s + \omega_i^2} \quad (8)$$

where

$$\bar{P}_i = \frac{C_p [\phi'_i(r_2) - \phi'_i(r_1)]}{\{\rho AL^3\}_{sb}} \quad (9)$$

and

$$\{\rho AL^3\}_{sb} = \rho_b A_b L_b^3 + 2\rho_p A_p L_p^3 \quad (10)$$

The detailed derivation of equation (8) can be found in (Kırcalı, 2006a). In this study, the assumed modes (i.e. the admissible functions) of the fixed-free smart beam are taken as the eigenfunctions of the fixed-free passive beam:

$$\phi_i(r) = L_b \left\{ \cosh \beta_i r - \cos \beta_i r - \frac{\cos \beta_i L_b + \cosh \beta_i L_b}{\sin \beta_i L_b + \sinh \beta_i L_b} (\sinh \beta_i r - \sin \beta_i r) \right\} \quad (11)$$

3. Model Correction and Spatial Identification of the Smart Beam

Assumed-modes method uses admissible functions in order to model the dynamics of the system, but ignores the nonuniform mass and stiffness distributions. If one uses a large number of admissible functions, or more general if their number goes to infinity, the model will be exactly the same as the original one. However, using infinite number of admissible functions is not convenient to apply for real structures at least for huge amount of computing requirements. Therefore, it is generally believed that the utilization of sufficiently large number of admissible functions will be enough to increase the accuracy of the approximate system model (Hughes, 1987).

Including large number of admissible functions leads to not only a more accurate but also a high order approximate system model. Since the order of an H_∞ controller depends on the system order, such a higher order model yields an excessive order controller which may not be possibly implemented. However, the controller design techniques generally focus on a particular bandwidth which includes only a few vibration modes of the system. In this respect, the reduction of the order of the model is required.

One of the most popular techniques for reducing the order of the system model is the direct model reduction, which simplifies the system model by directly truncating the higher modes of frequency range of interest. However, removing the higher modes may perturb the zeros of the system which will affect the closed-loop performance and stability (Clark, 1997). One particular approach to compensate the error of the model truncation was

presented by Moheimani (Moheimani, 2000a) which considers adding a correction term that minimizes the weighted spatial H_2 norm of the truncation error. The additional correction term had a good improvement on low frequency dynamics of the truncated model. Moheimani (2000d) and Moheimani et al. (2000c) developed their corresponding approach to the spatial models which are obtained by different analytical methods. Moheimani (2006b) presented an application of the model correction technique on a simply-supported piezoelectric laminate beam experimentally. However, in all those studies, the damping in the system was neglected. Halim (2002b) improved the model correction approach with damping effect in the system. This section will give a brief explanation of the model correction technique with damping effect based on those previous works (Moheimani, 2000a, 2000c and 2000d) and for more detailed explanation the reader is advised to refer to the reference (Moheimani, 2003).

Recall the transfer function of the system from system input to the beam deflection including N number of modes given in equation (8). The spatial system model expression includes N number of resonant modes assuming that N is sufficiently large. The controller design however interests in the first few vibration modes of the system, say M number of lowest modes. So the truncated model including first M number of modes can be expressed as:

$$G_M(s, r) = \sum_{i=1}^M \frac{\bar{P}_i \phi_i(r)}{s^2 + 2\xi_i \omega_i s + \omega_i^2} \quad (12)$$

where $M \ll N$. This truncation may cause error due to the removed modes which can be expressed as an error system model, $E(s, r)$:

$$\begin{aligned} E(s, r) &= G_N(s, r) - G_M(s, r) \\ &= \sum_{i=M+1}^N \frac{\bar{P}_i \phi_i(r)}{s^2 + 2\xi_i \omega_i s + \omega_i^2} \end{aligned} \quad (13)$$

In order to compensate the model truncation error, a correction term should be added to the truncated model (Halim, 2002b):

$$G_C(s, r) = G_M(s, r) + K(r) \quad (14)$$

where $G_C(s, r)$ and $K(r)$ are the corrected transfer function and correction term, respectively.

The correction term $K(r)$ involves the effects of removed modes of the system on the frequency range of interest, and can be expressed as:

$$K(r) = \sum_{i=M+1}^N \phi_i(r)k_i \quad (15)$$

where k_i is a constant term. The reasonable value of k_i should be determined by keeping the difference between $G_N(s, r)$ and $G_C(s, r)$ to be minimum, i.e. corrected system model should approach more to the higher ordered one given in equation (8). Moheimani (2000a) represents this condition by a cost function, J , which describes that the spatial H_2 norm of the difference between $G_N(s, r)$ and $G_C(s, r)$ should be minimized:

$$J = \langle\langle W(s, r) \{G_N(s, r) - G_C(s, r)\} \rangle\rangle_2^2 \quad (16)$$

The notation $\langle\langle \dots \rangle\rangle_2^2$ represents the spatial H_2 norm of a system where spatial norm definitions are given in (Moheimani, 2003). $W(s, r)$ is an ideal low-pass weighting function distributed spatially over the entire domain R with its cut-off frequency ω_c chosen to lie within the interval (ω_M, ω_{M+1}) (Moheimani, 2000a). That is:

$$|W(j\omega, r)| = \begin{cases} 1 & -\omega_c < \omega < \omega_{c+1}, r \in R \\ 0 & \text{elsewhere} \end{cases} \quad (17)$$

and $\omega_c \in (\omega_M, \omega_{M+1})$

where ω_M and ω_{M+1} are the natural frequencies associated with mode number M and $M + 1$, respectively. Halim (2002b) showed that, by taking the derivative of cost function J with respect to k_i and using the orthogonality of eigenfunctions, the general optimal value of the correction term, so called k_i^{opt} , for the spatial model of resonant systems, including the damping effect, can be shown to be:

$$k_i^{opt} = \frac{1}{4\omega_c\omega_i} \frac{1}{\sqrt{1-\xi_i^2}} \ln \left\{ \frac{\omega_c^2 + 2\omega_c\omega_i\sqrt{1-\xi_i^2} + \omega_i^2}{\omega_c^2 - 2\omega_c\omega_i\sqrt{1-\xi_i^2} + \omega_i^2} \right\} \bar{P}_i \quad (18)$$

An interesting result of equation (18) is that, if damping coefficient is selected as zero for each mode, i.e. undamped system, the resultant correction term is equivalent to those given

in references (Moheimani, 2000a, 2000c and 2000d) for an undamped system. Therefore, equation (18) can be represented as not only the optimal but also the general expression of the correction term.

So, following the necessary mathematical manipulations, one will obtain the corrected system model including the effect of out-of-range modes as:

$$G_C(s, r) = \sum_{i=1}^M \frac{\bar{P}_i \phi_i(r)}{s^2 + 2\xi_i \omega_i s + \omega_i^2} + \sum_{i=M+1}^N \phi_i(r) \left\{ \frac{1}{4\omega_c \omega_i} \frac{1}{\sqrt{1-\xi_i^2}} \ln \left\{ \frac{\omega_c^2 + 2\omega_c \omega_i \sqrt{1-\xi_i^2} + \omega_i^2}{\omega_c^2 - 2\omega_c \omega_i \sqrt{1-\xi_i^2} + \omega_i^2} \right\} \bar{P}_i \right\} \quad (19)$$

Consider the cantilevered smart beam depicted in Fig.1 with the structural properties given at Table 1. The beginning and end locations of the PZT patches $r_1 = 0.027m$ and $r_2 = 0.077m$ away from the fixed end, respectively. Note that, although the actual length of the passive beam is 507mm, the effective length, or span, reduces to 494mm due to the clamping in the fixture.

	Aluminum Beam	Passive	PZT
Length	$L_b = 0.494m$		$L_p = 0.05m$
Width	$w_b = 0.051m$		$w_p = 0.04m$
Thickness	$t_b = 0.002m$		$t_p = 0.0005m$
Density	$\rho_b = 2710kg/m^3$		$\rho_p = 7650kg/m^3$
Young's Modulus	$E_b = 69GPa$		$E_p = 64.52GPa$
Cross-sectional Area	$A_b = 1.02 \times 10^{-4} m^2$		$A_p = 0.2 \times 10^{-4} m^2$
Second Moment of Area	$I_b = 3.4 \times 10^{-11} m^4$		$I_p = 6.33 \times 10^{-11} m^4$
Piezoelectric charge constant	-		$d_{31} = -175 \times 10^{-12} m/V$

Table 1. Properties of the Smart Beam

The system model given in equation (8) includes N number of modes of the smart beam, where as N gets larger, the model becomes more accurate. In this study, first 50 flexural resonance modes are included into the model (i.e. $N=50$) and the resultant model is called *the full order model*:

$$G_{50}(s, r) = \sum_{i=1}^{50} \frac{\bar{P}_i \phi_i(r)}{s^2 + 2\xi_i \omega_i s + \omega_i^2} \quad (20)$$

However, the control design criterion of this study is to suppress only the first two flexural modes of the smart beam. Hence, the full order model is directly truncated to a lower order model, including only the first two flexural modes, and the resultant model is called *the truncated model*:

$$G_2(s, r) = \sum_{i=1}^2 \frac{\bar{P}_i \phi_i(r)}{s^2 + 2\xi_i \omega_i s + \omega_i^2} \quad (21)$$

As previously explained, the direct model truncation may cause the zeros of the system to perturb, which consequently affect the closed-loop performance and stability of the system considered (Clark, 1997). For this reason, the general correction term, given in equation (18), is added to the truncated model and the resultant model is called *the corrected model*:

$$G_C(s, r) = \sum_{i=1}^2 \frac{\bar{P}_i \phi_i(r)}{s^2 + 2\xi_i \omega_i s + \omega_i^2} + \sum_{i=3}^{50} \phi_i(r) \left\{ \frac{1}{4\omega_c \omega_i} \frac{1}{\sqrt{1-\xi_i^2}} \ln \left\{ \frac{\omega_c^2 + 2\omega_c \omega_i \sqrt{1-\xi_i^2} + \omega_i^2}{\omega_c^2 - 2\omega_c \omega_i \sqrt{1-\xi_i^2} + \omega_i^2} \right\} \bar{P}_i \right\} \quad (22)$$

where the cut-off frequency, based on the selection criteria given in equation (17), is taken as:

$$\omega_c = (\omega_2 + \omega_3) / 2 \quad (23)$$

The assumed-modes method gives the first three resonant frequencies of the smart beam as shown in Table 2. Hence, the cut-off frequency becomes 79.539 Hz. The performance of model correction for various system models obtained from different measurement points along the beam is shown in Fig.3 and Fig.4.

Resonant Frequencies	Value (Hz)
ω_1	6.680
ω_2	41.865
ω_3	117.214

Table 2. First three resonant frequencies of the smart beam

The error between full order model-truncated model, and the error between full order model-corrected model, so called the error system models \bar{E}_{F-T} and \bar{E}_{F-C} , allow one to see the effect of model correction more comprehensively.

$$\bar{E}_{F-T} = G_N(s, r) - G_M(s, r) \quad (24)$$

$$\bar{E}_{F-C} = G_N(s, r) - G_C(s, r) \quad (25)$$

The frequency responses of the error system models are shown in Fig.5 and Fig.6. One can easily notice from the aforementioned figures that, the error between the full order and corrected models is less than the error between the full order and truncated ones in a wide range of the interested frequency bandwidth. That is, the model correction minimizes error considerably and makes the truncated model approach close to the full order one. The error between the full order and corrected models is smaller at low frequencies and around 50 Hz it reaches a minimum value. As a result, model correction reduces the overall error due to model truncation, as desired.

In this study, the experimental system models based on displacement measurements were obtained by nonparametric identification. The smart beam was excited by piezoelectric patches with sinusoidal chirp signal of amplitude 5V within bandwidth of 0.1-60 Hz, which covers the first two flexural modes of the smart beam. The response of the smart beam was acquired via laser displacement sensor from specified measurement points. Since the patches are relatively thin compared to the passive aluminum beam, the system was considered as 1-D single input multi output system, where all the vibration modes are flexural modes. The open loop experimental setup is shown in Fig.7.

In order to have more accurate information about spatial characteristics of the smart beam, 17 different measurement points, shown in Fig.8, were specified. They are defined at 0.03m intervals from tip to the root of the smart beam.

The smart beam was actuated by applying voltage to the piezoelectric patches and the transverse displacements were measured at those locations. Since the smart beam is a spatially distributed system, that analysis resulted in 17 different single input single output system models where all the models were supposed to share the same poles. That kind of

analysis yields to determine uncertainty of resonance frequencies due to experimental approach. Besides, comparison of the analytical and experimental system models obtained for each measurement points was used to determine modal damping ratios and the uncertainties on them. That is the reason why measurement from multiple locations was employed. The rest of this section presents the comparison of the analytical and experimental system models to determine modal damping ratios and clarify the uncertainties on natural frequencies and modal damping ratios.

Consider the experimental frequency response of the smart beam at point $r = 0.99L_b$. Because experimental frequency analysis is based upon the exact dynamics of the smart beam, the values of the resonance frequencies determined from experimental identification were treated as being more accurate than the ones obtained analytically, where the analytical values are presented in Table 2. The first two resonance frequencies were extracted as 6.728 Hz and 41.433 Hz from experimental system model. Since the analytical and experimental models should share the same resonance frequencies in order to coincide in the frequency domain, the analytical model for the location $r = 0.99L_b$ was coerced to have the same resonance frequencies given above. Notice that, the corresponding measurement point can be selected from any of the measurement locations shown in Fig.8. Also note that, the analytical system model is the corrected model of the form given in equation (22). The resultant frequency responses are shown in Fig.9.

The analytical frequency response was obtained by considering the system as undamped. The point $r = 0.99L_b$ was selected as measurement point because of the fact that the free end displacement is significant enough for the laser displacement sensor measurements to be more reliable. After obtaining both experimental and analytical system models, the modal damping ratios were tuned until the magnitude of both frequency responses coincide at resonance frequencies, i.e.:

$$\left| G_E(s, r) - G_C(s, r) \right|_{\omega=\omega_i} < \lambda \quad (26)$$

where $G_E(s, r)$ is the experimental transfer function and λ is a very small constant term. Similar approach can be employed by minimizing the 2-norm of the differences of the displacements by using least square estimates (Reinelt, 2002).

Fig.10 shows the effect of tuning modal damping ratios on matching both system models in frequency domain where λ is taken as 10^{-6} . Note that each modal damping ratio can be tuned independently.

Consequently, the first two modal damping ratios were obtained as 0.0284 and 0.008, respectively. As the resonance frequencies and damping ratios are independent of the location of the measurement point, they were used to obtain the analytical system models of the smart beam for all measurement points. Afterwards, experimental system identification was again performed for each point and both system models were again compared in

frequency domain. The experimentally identified flexural resonance frequencies and modal damping ratios were determined by tuning for each point and finally a set of resonance frequencies and modal damping ratios were obtained. The amount of uncertainty on resonance frequencies and modal damping ratios can also be determined by spatial system identification. There are different methods which can be applied to determine the uncertainty and improve the values of the parameters ω and ξ such as boot-strapping (Reinelt, 2002). However, in this study the uncertainty is considered as the standard deviation of the parameters and the mean values are accepted as the final values, which are presented at Table 3.

	ω_1 (Hz)	ω_2 (Hz)	ξ_1	ξ_2
Mean	6.742	41.308	0.027	0.008
Standard Deviation	0.010	0.166	0.002	0.001

Table 3. Mean and standard deviation of the first two resonance frequencies and modal damping ratios

For more details about spatial system identification one may refer to (Kırçalı, 2006a). The estimated and analytical first two mode shapes of the smart beam are given in Fig.11 and Fig.12, respectively (Kırçalı, 2006a).

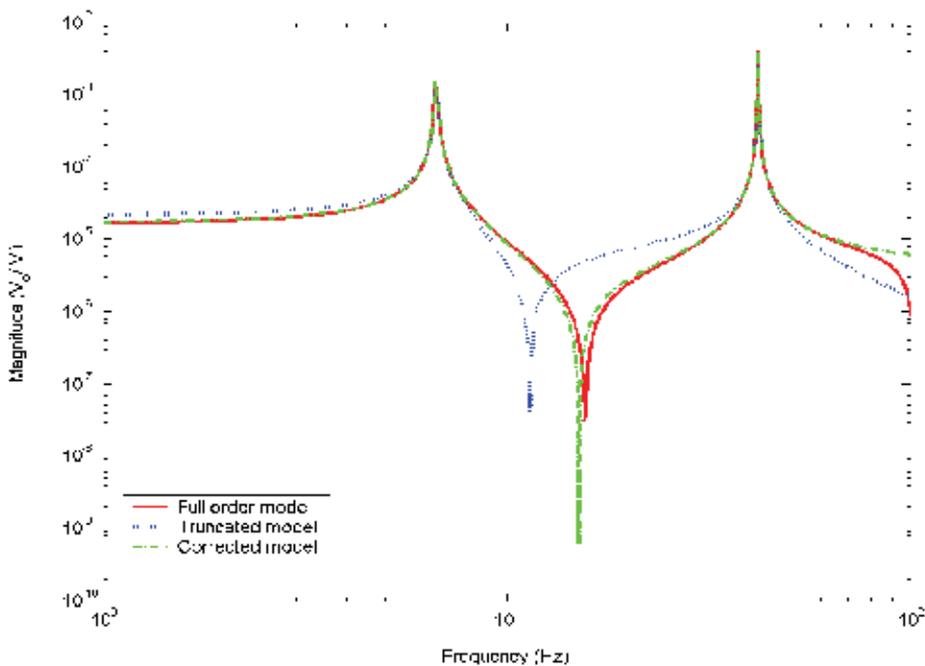


Fig. 3. Frequency response of the smart beam at $r = 0.14L_b$

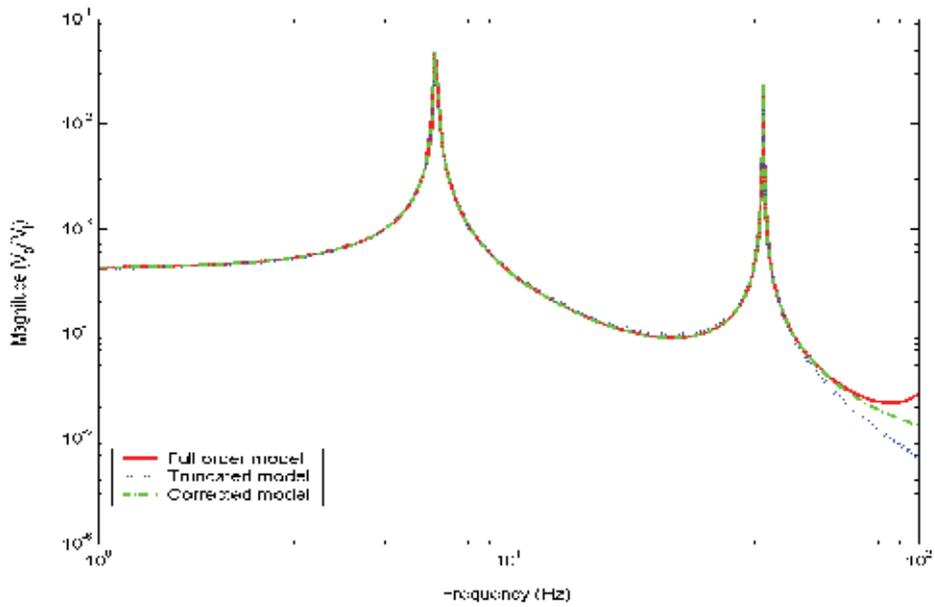


Fig. 4. Frequency response of the smart beam at $r = 0.99L_b$

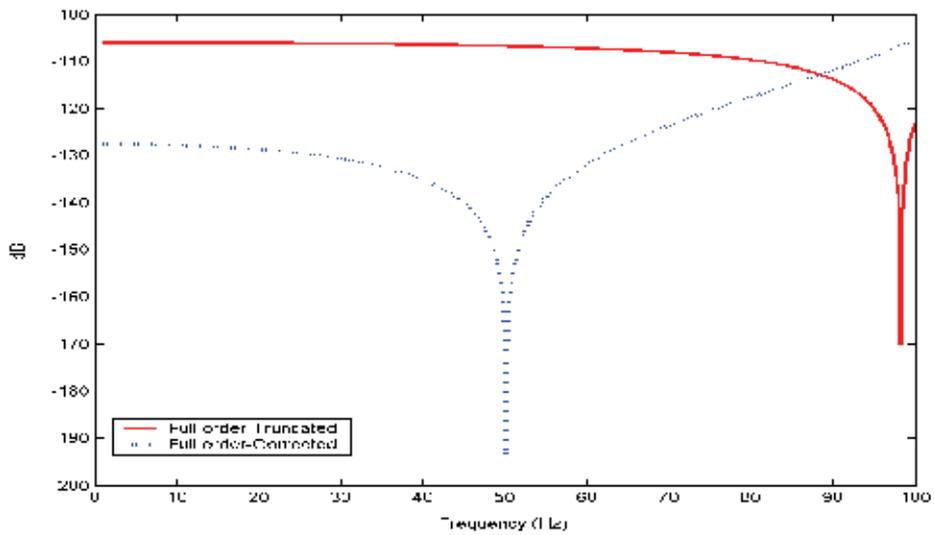


Fig. 5. Frequency responses of the error system models at $r = 0.14L_b$

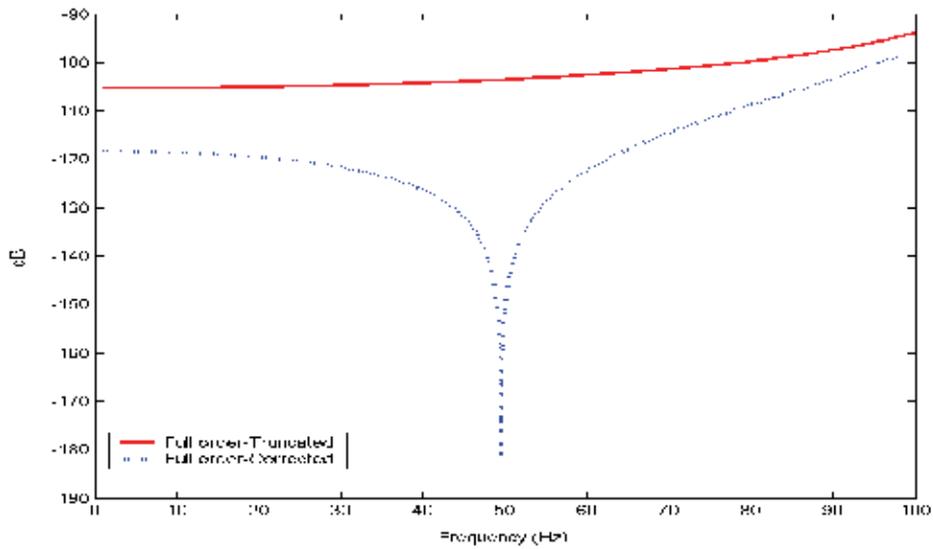


Fig. 6. Frequency responses of the error system models at $r = 0.99L_b$

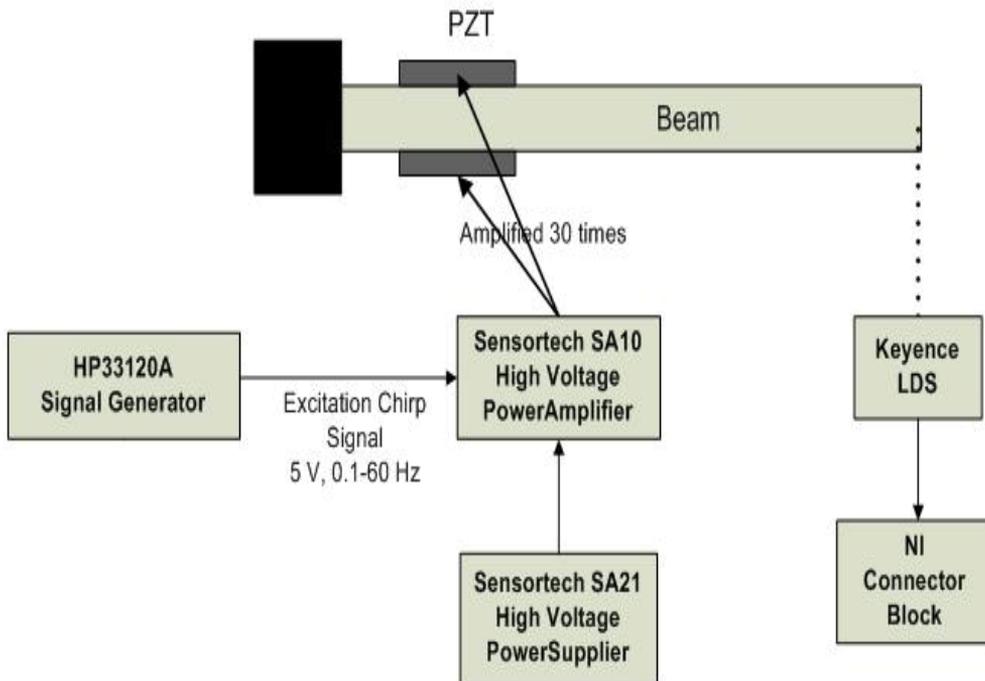


Fig. 7. Experimental setup for the spatial system identification of the smart beam

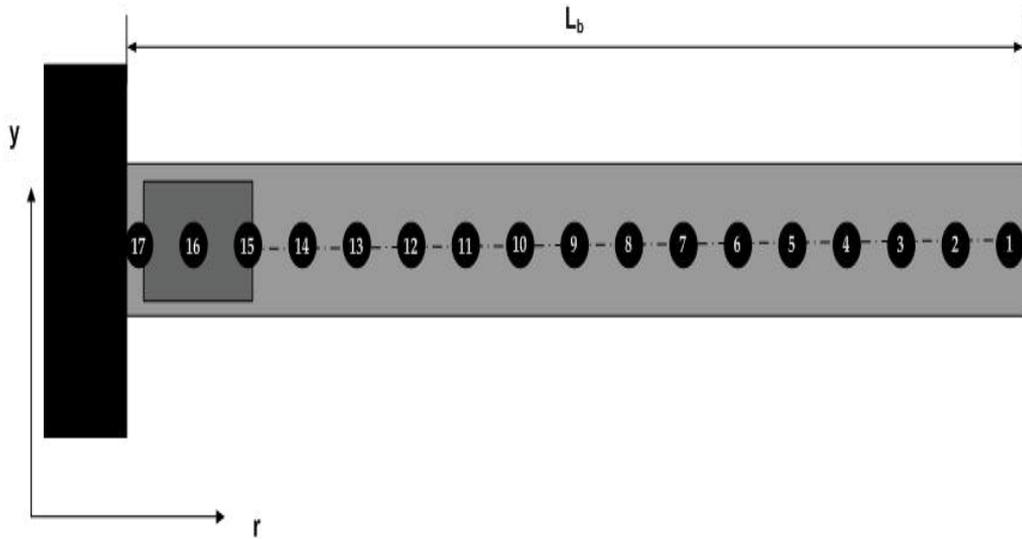


Fig. 8. The locations of the measurement points

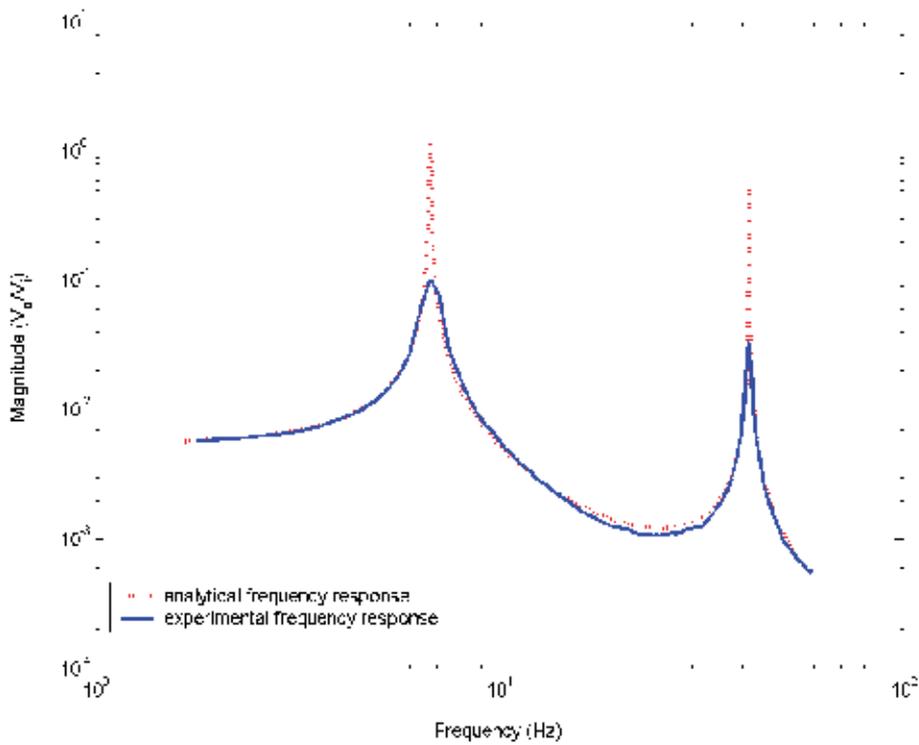


Fig. 9. Analytical and experimental frequency responses of the smart beam at $r=0.99 L_b$

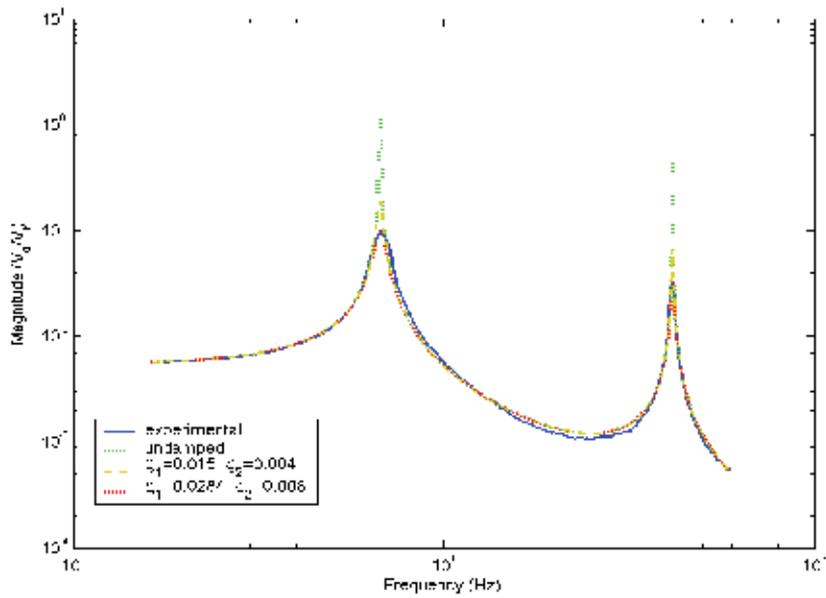


Fig. 10. Experimental and tuned analytical frequency responses at $r=0.99 L_b$

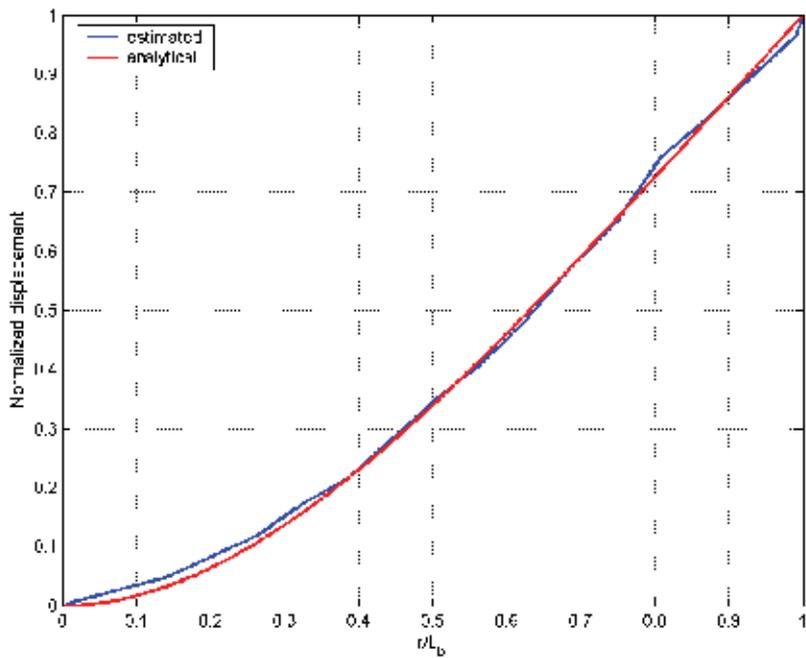


Fig. 11. First mode shape of the smart beam

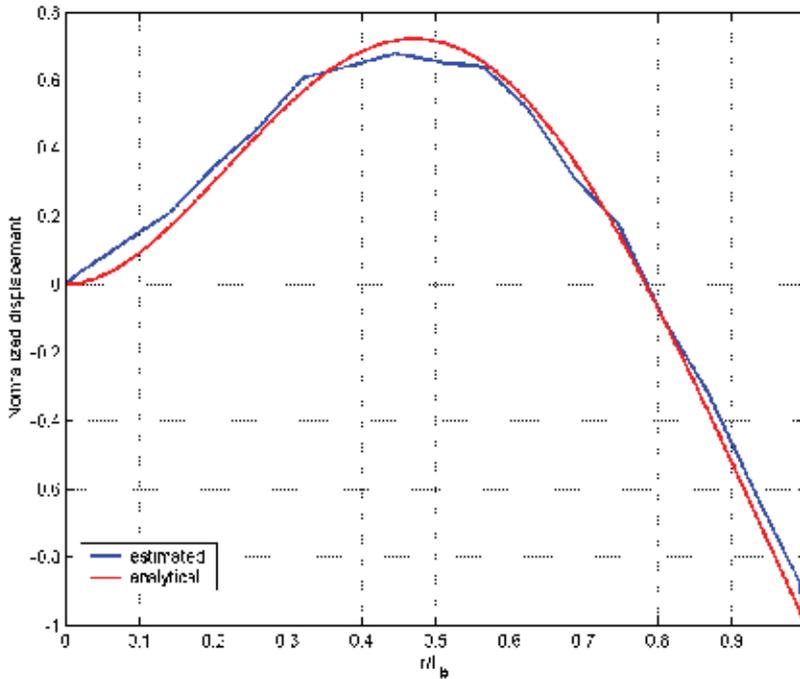


Fig. 12. Second mode shape of the smart beam

4. Spatial H_∞ Control Technique

Obtaining an accurate system model lets one to understand the system dynamics more clearly and gives him the opportunity to design a consistent controller. Various control design techniques have been developed for active vibration control like H_∞ or H_2 methods (Francis, 1984 and Doyle, 1989).

The effectiveness of H_∞ controller on suppressing the vibrations of a smart beam due to its first two flexural modes was studied by Yaman et al. (2001) and the experimental implementation of the controller was presented (2003). By means of H_∞ theory, an additive uncertainty weight was included to account for the effects of truncated high frequency modes as the model correction. Similar work has been done for suppressing the in-vacuo vibrations due to the first two modes of a smart fin (Yaman, 2002a, 2002b) and the effectiveness of the H_∞ control technique in the modeling of uncertainties was also shown.

However, H_∞ theory does not take into account the multiple sources of uncertainties, which yield unstructured uncertainty and increase controller conservativeness, at different locations of the plant. That problem can be handled by using the μ -synthesis control design method (Nalbantoğlu, 1998; Ülker, 2003 and Yaman, 2003).

Whichever the controller design technique is employed, the major objective of vibration control of a flexible structure is to suppress the vibrations of the first few modes on well-defined specific locations over the structure. As the flexible structures are distributed parameter systems, the vibration at a specific point is actually related to the vibration over the rest of the structure. As a remedy, minimizing the vibration over entire structure rather than at specific points should be the controller design criterion. The cost functions minimized as design criteria in standard H_2 or H_∞ control methodologies do not contain any information about the spatial nature of the system. In order to handle this absence, Moheimani and Fu (1998c), and Moheimani et al. (1997, 1998a) redefined H_2 and H_∞ norm concepts. They introduced spatial H_2 and spatial H_∞ norms of both signals and systems to be used as performance measures.

The concept of spatial control has been developed since the last decade. Moheimani et al. (1998a) studied the application of spatial LQG and H_∞ control technique for active vibration control of a cantilevered piezoelectric laminate beam. They presented simulation based results in their various works (1998a, 1998b, 1999). Experimental implementation of the spatial H_2 and H_∞ controllers were first achieved by Halim (2002a, 2002b, 2002c). These studies proved that the implementation of the spatial controllers on real systems is possible and that kind of controllers show considerable superiority compared to pointwise controllers on suppressing the vibration over entire structure. However, these works examined only simply-supported piezoelectric laminate beam. The contribution to the need of implementing spatial control technique on different systems was done by Lee (2005). Beside vibration suppression, he studied attenuation of acoustic noise due to structural vibration on a simply-supported piezoelectric laminate plate.

This section gives a brief explanation of the spatial H_∞ control technique based on the complete theory presented in reference (Moheimani, 2003). For more detailed explanation the reader is advised to refer to the references (Moheimani, 2003 and Halim, 2002b).

Consider the state space representation of a spatially distributed linear time-invariant (LTI) system:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + B_1w(t) + B_2u(t) \\ z(t, r) &= C_1(r)x(t) + D_1(r)w(t) + D_2(r)u(t) \\ y(t) &= C_2x(t) + D_3w(t) + D_4u(t)\end{aligned}\tag{27}$$

where r is the spatial coordinate, x is the state vector, w is the disturbance input, u is the control input, z is the performance output and y is the measured output. The state space representation variables are as follows: A is the state matrix, B_1 and B_2 are the input matrices from disturbance and control actuators, respectively, C_1 is the output matrix of

error signals, C_2 is the output matrix of sensor signals, D_1 , D_2 , D_3 and D_4 are the correction terms from disturbance actuator to error signal, control actuator to error signal, disturbance actuator to feedback sensor and control actuator to feedback sensor, respectively.

The spatial H_∞ control problem is to design a controller which is:

$$\begin{aligned}\dot{x}_k(t) &= A_k x_k(t) + B_k y(t) \\ u(t) &= C_k x_k(t) + D_k y(t)\end{aligned}\quad (28)$$

such that the closed loop system satisfies:

$$\inf_{K \in U} \sup_{w \in L_2[0, \infty)} J_\infty < \gamma^2 \quad (29)$$

where U is the set of all stabilizing controllers and γ is a constant. The spatial cost function to be minimized as the design criterion of spatial H_∞ control design technique is:

$$J_\infty = \frac{\int_0^\infty \int_R z(t, r)^T Q(r) z(t, r) dr dt}{\int_0^\infty w(t)^T w(t) dt} \quad (30)$$

where $Q(r)$ is a spatial weighting function that designates the region over which the effect of the disturbance is to be reduced. Since the numerator is the weighted spatial H_2 norm of the performance signal $z(t, r)$, J_∞ can be considered as the ratio of the spatial energy of the system output to that of the disturbance signal (Moheimani, 2003). The control problem is depicted in Fig.13:

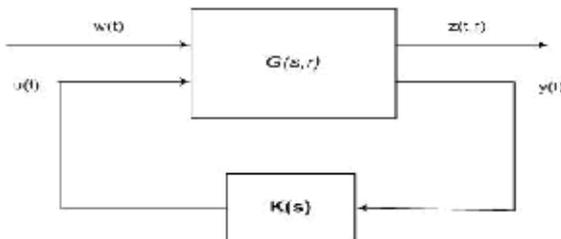


Fig. 13. Spatial H_∞ control problem

Spatial H_∞ control problem can be solved by the equivalent ordinary H_∞ problem (Moheimani, 2003) by taking:

$$\int_0^\infty \int_R z(t,r)^T Q(r) z(t,r) dr dt = \int_0^\infty \tilde{z}(t)^T \tilde{z}(t) dt \quad (31)$$

so, the spatial cost function becomes:

$$J_\infty = \frac{\int_0^\infty \tilde{z}(t)^T \tilde{z}(t) dt}{\int_0^\infty w(t)^T w(t) dt} \quad (32)$$

So the spatial H_∞ control problem is reduced to a standard H_∞ control problem for the following system:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B_1 w(t) + B_2 u(t) \\ \tilde{z}(t) &= \Pi x(t) + \Theta_1 w(t) + \Theta_2 u(t) \\ y(t) &= C_2 x(t) + D_3 w(t) + D_4 u(t) \end{aligned} \quad (33)$$

However, in order to limit the controller gain and avoid actuator saturation problem, a control weight should be added to the system.

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B_1 w(t) + B_2 u(t) \\ \tilde{z}(t) &= \begin{bmatrix} \Pi \\ 0 \end{bmatrix} x(t) + \begin{bmatrix} \Theta_1 \\ 0 \end{bmatrix} w(t) + \begin{bmatrix} \Theta_2 \\ \mathcal{K} \end{bmatrix} u(t) \\ y(t) &= C_2 x(t) + D_3 w(t) + D_4 u(t) \end{aligned} \quad (34)$$

where \mathcal{K} is the control weight and it designates the level of vibration suppression. Control weight prevents the controller having excessive gain and smaller \mathcal{K} results in higher level of vibration suppression. However, optimal value of \mathcal{K} should be determined in order not to destabilize or neutrally stabilize the system.

Application of the above theory to our problem is as follows: Consider the closed loop system of the smart beam shown in Fig.14. The aim of the controller, K , is to reduce the effect of disturbance signal over the entire beam by the help of the PZT actuators.

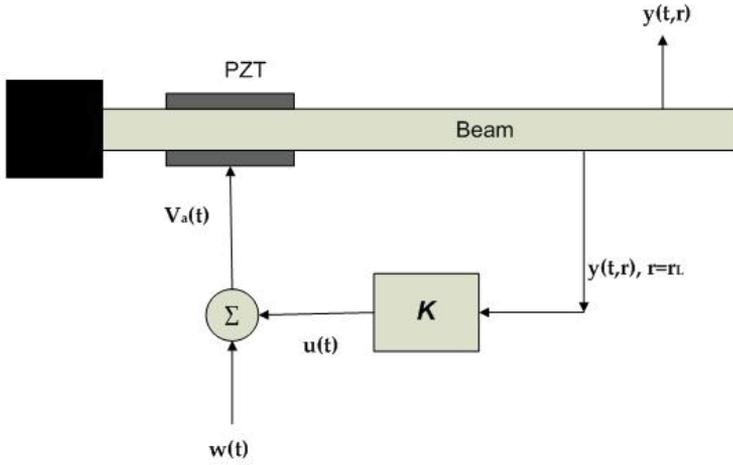


Fig. 14. The closed loop system of the smart beam

The state space representation of the system above can be shown to be (Kircali, 2008 and 2006a):

$$\begin{aligned}
 \dot{x}(t) &= Ax(t) + B_1w(t) + B_2u(t) \\
 y(t,r) &= C_1(r)x(t) + D_1(r)w(t) + D_2(r)u(t) \\
 y(t,r_L) &= C_2x(t) + D_3w(t) + D_4u(t)
 \end{aligned} \tag{35}$$

where all the state space parameters were defined at Section 2.4, except the performance output and the measured output which are now denoted as $y(t,r)$ and $y(t,r_L)$, respectively. The performance output represents the displacement of the smart beam along its entire body, and the measured output represents the displacement of the smart beam at a specific location, i.e. $r = r_L$. The disturbance $w(t)$ is accepted to enter to the system through the actuator channels, hence, $B_1 = B_2$, $D_1(r) = D_2(r)$ and $D_3 = D_4$.

The state space form of the controller design, given in equation (28), can now be represented as:

$$\begin{aligned}
 \dot{x}_k(t) &= A_kx_k(t) + B_ky(t,r_L) \\
 u(t) &= C_kx_k(t) + D_ky(t,r_L)
 \end{aligned} \tag{36}$$

Hence, the spatial H_∞ control problem can be represented as a block diagram which is given in Fig.15:

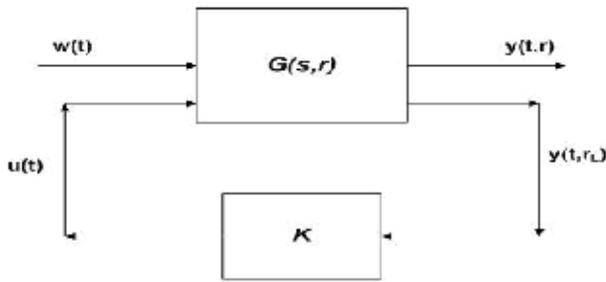


Fig. 15. The Spatial H_∞ control problem of the smart beam

As stated above, the spatial H_∞ control problem can be reduced to a standard H_∞ control problem. The state space representation given in equation (35) can be adapted for the smart beam model for a standard H_∞ control design as:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + B_1 w(t) + B_2 u(t) \\ \tilde{y}(t) &= \begin{bmatrix} \Pi \\ 0 \end{bmatrix} x(t) + \begin{bmatrix} \Theta_1 \\ 0 \end{bmatrix} w(t) + \begin{bmatrix} \Theta_2 \\ \kappa \end{bmatrix} u(t) \\ y(t, r_L) &= C_2 x(t) + D_3 w(t) + D_4 u(t)\end{aligned}\quad (37)$$

The state space variables given in equations (35) and (37) can be obtained from the transfer function of equation (22) as:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\omega_1^2 & 0 & -2\xi_1\omega_1 & 0 \\ 0 & -\omega_2^2 & 0 & -2\xi_2\omega_2 \end{bmatrix}, B_1 = B_2 = \begin{bmatrix} 0 \\ 0 \\ \bar{P}_1 \\ \bar{P}_2 \end{bmatrix}\quad (38)$$

$$\Pi = \begin{bmatrix} L_b^{3/2} & 0 & 0 & 0 \\ 0 & L_b^{3/2} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \Theta_1 = \Theta_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \left(\sum_{i=3}^{50} L_b^3 (k_i^{opt})^2 \right)^{1/2} \end{bmatrix}\quad (39)$$

$$C_1 = [\phi_1(r) \quad \phi_2(r) \quad 0 \quad 0], C_2 = [\phi_1(r_L) \quad \phi_2(r_L) \quad 0 \quad 0] \quad (40)$$

$$D_1 = D_2 = \sum_{i=3}^{50} \phi_i(r) k_i^{opt}, D_3 = D_4 = \sum_{i=3}^{50} \phi_i(r_L) k_i^{opt} \quad (41)$$

The detailed derivation of the above parameters can be found in (Kircali, 2006a).

One should note that, in the absence of the control weight, \mathcal{K} , the major problem of designing an H_∞ controller for the system is that, such a design will result in a controller with an infinitely large gain (Moheimani, 1999). As previously described, in order to overcome this problem, an appropriate control weight, which is determined by the designer, is added to the system. Since the smaller \mathcal{K} will result in higher vibration suppression but larger controller gain, it should be determined optimally such that not only the gain of the controller does not cause implementation difficulties but also the suppression of the vibration levels are satisfactory. In this study, \mathcal{K} was taken as 7.87×10^{-7} . The simulation of the effect of the controller is shown in Fig.16 as a bode plot. The frequency domain simulation was done by Matlab v6.5.

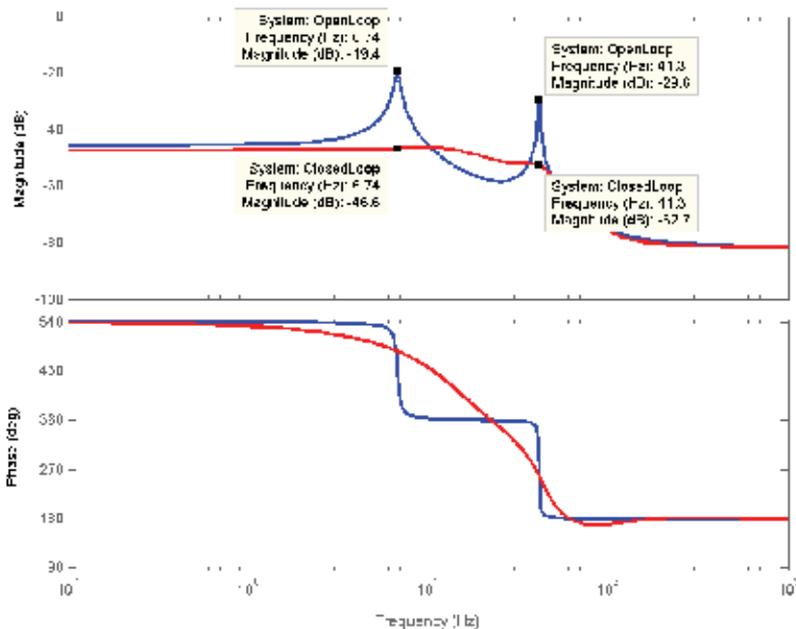


Fig. 16. Bode plots of the open loop and closed loop systems under the effect of spatial H_∞ controller

The vibration attenuation levels at the first two flexural resonance frequencies were found to be 27.2 dB and 23.1 dB, respectively. The simulated results show that the designed controller is effective on the suppression of undesired vibration levels.

4.1 Implementation of the Spatial Controller

This section presents the implementation of the spatial H_∞ controller for suppressing the free and forced vibrations of the smart beam. The closed loop experimental setup is shown in Fig.17. The displacement of the smart beam at a specific location was measured by using a Keyence Laser Displacement Sensor (LDS) and converted to a voltage output that was sent to the SensorTech SS10 controller unit via the connector block. The controller output was converted to the analog signal and amplified 30 times by SensorTech SA10 high voltage power amplifier before being applied to the piezoelectric patches. The controller unit is hosted by a Linux machine on which a shared disk drive is present to store the input/output data and the C programming language based executable code that is used for real-time signal processing.

For the free vibration control, the smart beam was given an initial 5 cm tip deflection and the open loop and closed loop time responses of the smart beam were measured. The results are presented in Fig.18 which shows that the controlled time response of the smart beam settles nearly in 1.7 seconds. Hence, the designed controller proves to be very effective on suppressing the free vibration of the smart beam.

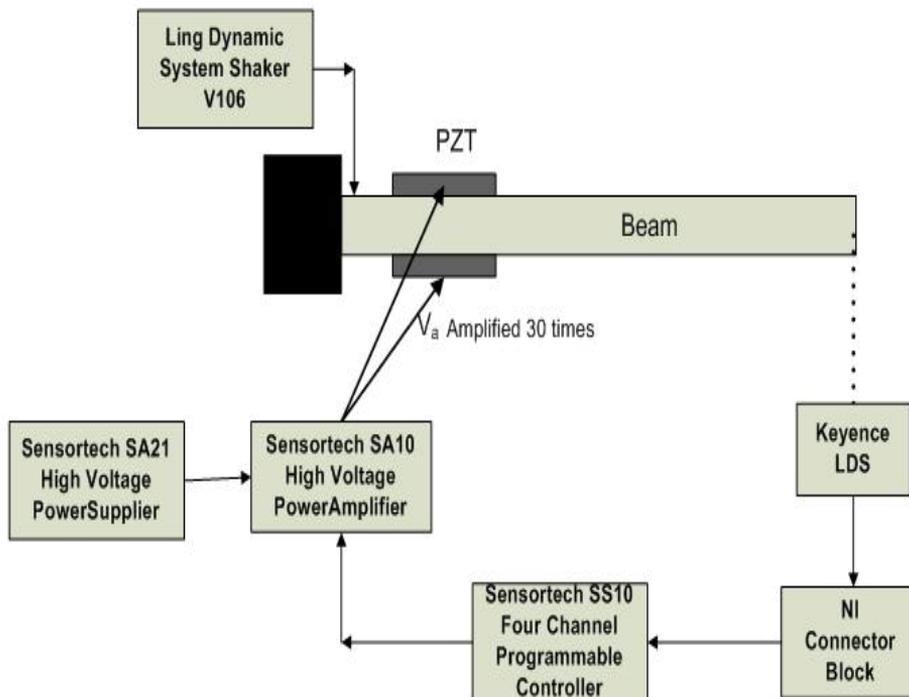


Fig. 17. The closed loop experimental setup

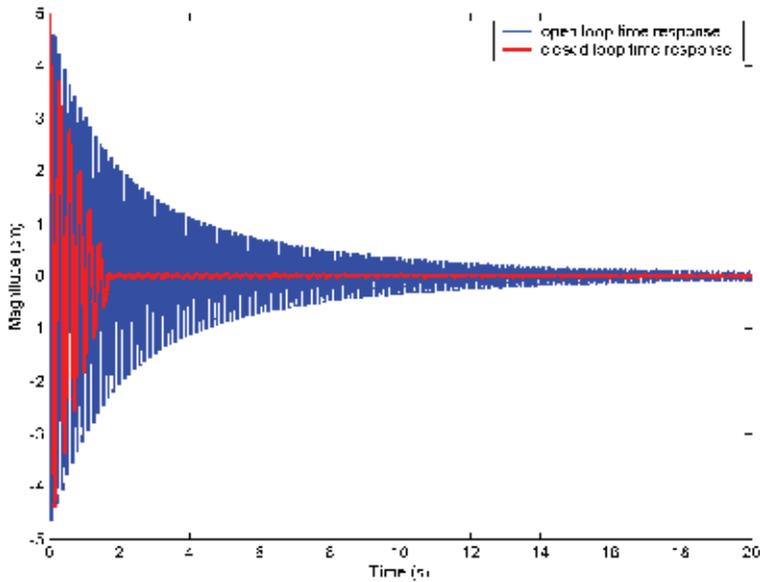


Fig. 18. Open and closed loop time responses of the smart beam under the effect of spatial H_{∞} controller

The forced vibration control of the smart beam was analyzed in two different configurations. In the first one, the smart beam was excited for 180 seconds with a shaker located very close to the root of the smart beam, on which a sinusoidal chirp signal of amplitude 4.5V was applied. The excitation bandwidth was taken first 5 to 8 Hz and later 40 to 44 Hz to include the first two flexural resonance frequencies separately. The open loop and closed loop time and frequency responses of the smart beam under respective excitations are shown in Fig.19-a, Fig.19-b, Fig.20. Note that the Nyquist plot of the nominal system loop gain under the effect of spatial H_{∞} controller given in Fig. 21 shows that the nominal system is stable.

The experimental attenuation of vibration levels at first two resonance frequencies were determined from the Bode magnitude plots of the frequency responses of the smart beam and shown in Fig.20-a and Fig.20-b. The resultant attenuation levels were found as 19.8 dB and 14.2 dB, respectively. Hence, the experimental results show that the controller is effective on suppression of the vibration levels. The reason why experimental attenuation levels are less than the simulated ones is that, the excitation power of the shaker was not enough to make the smart beam to reach the larger deflections which in turn causes a smaller magnitude of the open loop time response. The hardware constraints prevent one to apply higher voltages to the shaker. On the other hand, the magnitude of the experimental and simulated closed loop frequency responses at resonance frequencies being close to each other makes one to realize that, the controller works exactly according to the design criteria. Additionally, one should note that the attenuation levels were obtained from the decibel

magnitudes of the frequency responses. Hence, a simple mathematical manipulation can give the absolute attenuation levels as a ratio of the maximum time responses of the open and closed loop systems at the specified resonance frequencies.

In the second configuration, instead of using a sinusoidal chirp signal, constant excitation was applied for 20 seconds at the resonance frequencies with a mechanical shaker. The open loop and closed loop time responses of the smart beam were measured and shown in Fig.21 and Fig.22. Although, it is hard to control such a resonant excitation, the time responses show that the designed controller is still very effective on suppressing the vibration levels. Recall that the ratio of the maximum time responses of the open and closed loop systems can be considered as absolute attenuation levels; hence, for this case, the attenuation levels at each resonance frequency were calculated approximately as 10.4 and 4.17, respectively.

The robustness analysis of the designed controller was performed by Matlab v6.5 μ -synthesis toolbox. The results are presented in Fig. 23. The theoretical background of μ -synthesis is detailed in the References (Zhou, 1998 and Ülker, 2003). One should know that the μ values should be less than unity to accept the controllers to be robust. The Fig.23 shows that the spatial H_∞ controller is robust to the perturbations.

The efficiency of spatial controller in minimizing the overall vibration over the smart beam was compared by a pointwise controller that is designed to minimize the vibrations only at point $r = 0.99L_b$. For a more detailed description of the pointwise controller design, the interested reader may refer to the reference (Kırcalı, 2006a and 2006b). However, in order to give the idea of the previous studies, the comparative effects of the spatial and pointwise H_∞ controllers on suppressing the first two flexural vibrations of the smart beam are briefly presented in Table 4:

Modes	Spatial H_∞ controller		Pointwise H_∞ controller	
	1 st mode	2 nd mode	1 st mode	2 nd mode
Simulated attenuation levels (dB)	27.2	23.1	23.5	24.4
Experimentally obtained attenuation levels (dB)	19.8	14.2	21.02	21.66
Absolute attenuation levels under constant resonant excitation (max. OL time response/ max. CL time response)	10.4	4.17	5.75	4.37

Table 4. The comparison of attenuation levels under the effect of spatial and pointwise H_∞ controllers in forced vibrations

The simulations show that both controllers work efficiently on suppressing the vibration levels. The forced vibration control experiments of first configuration show that the attenuation levels of pointwise controller are slightly higher than those of the spatial one. Although the difference is not significant especially for the first flexural mode, better attenuation of pointwise controller would not be a surprise since the respective design criterion of a pointwise controller is to suppress the undesired vibration level at the specific measurement point. Additionally, absolute attenuation levels show that under constant resonant excitation at the first flexural mode, the spatial H_∞ controller has better performance than the pointwise one. This is because the design criterion of spatial controller is to suppress the vibration over entire beam; hence, the negative effect of the vibration at any point over the beam on the rest of the other points is prevented by spatial means. So, the spatial H_∞ controller resists more robustly to the constant resonant excitation than the pointwise one.

The implementations of the controllers showed that both controllers reduced the vibration levels of the smart beam due to its first two flexural modes in comparable efficiency (Kırcalı, 2006a and 2006b). The effect of both controllers on suppressing the first two flexural vibrations of the smart beam over entire structure can be analyzed by considering the H_∞ norm of the entire beam. Fig.24 shows the H_∞ norm plots of the smart beam as a function of r under the effect of both controllers.

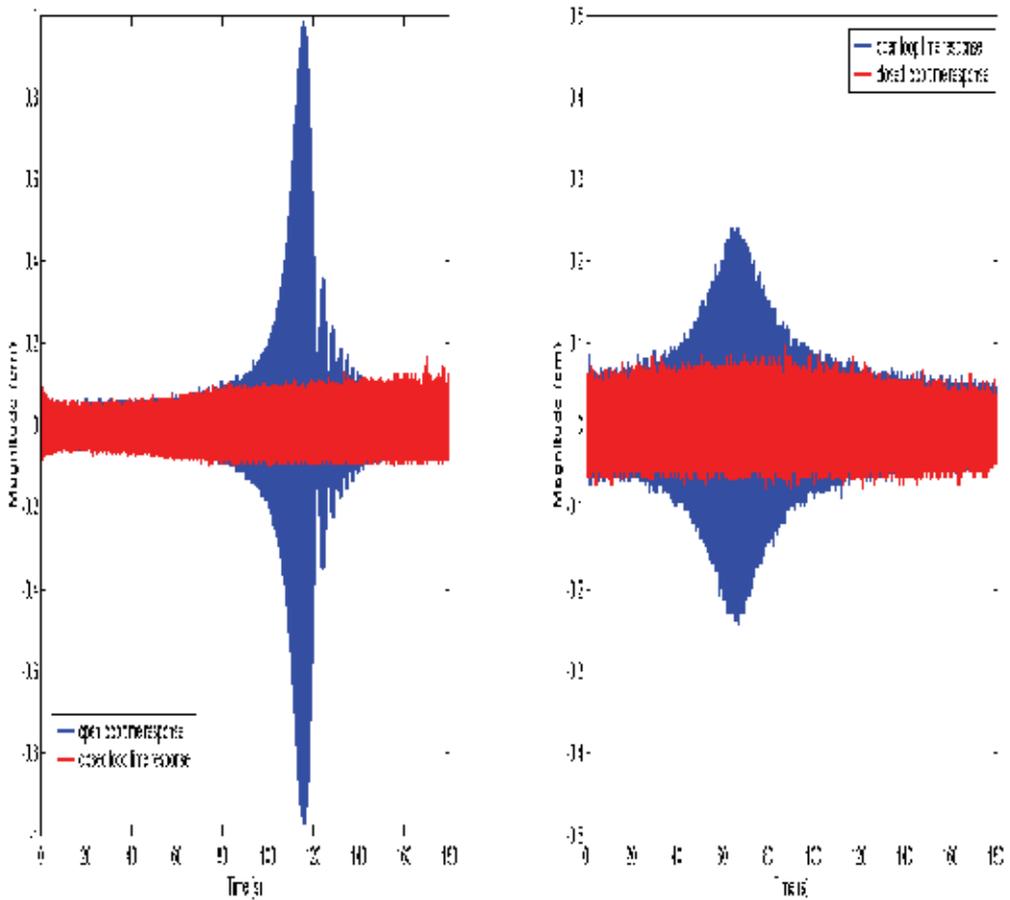
5. General Conclusions

This study presented a different approach in active vibration control of a cantilevered smart beam.

The required mathematical modeling of the smart beam was conducted by using the assumed-modes method. This inevitably resulted in a higher order model including a large number of resonant modes of the beam. This higher order model was truncated to a lower model by including only the first two flexural vibrational modes of the smart beam. The possible error due to that model truncation was compensated by employing a model correction technique which considered the addition of a correction term that consequently minimized the weighted spatial H_2 norm of the truncation error. Hence, the effect of out-of-range modes on the dynamics of the system was included by the correction term. During the modeling phase the effect of piezoelectric patches was also conveniently included in the model to increase the accuracy of the system model. However, the assumed-modes modeling alone does not provide any information about the damping of the system. It was shown that experimental system identification, when used in collaboration with the analytical model, helps one to obtain more accurate spatial characteristics of the structure. Since the smart beam is a spatially distributed structure, experimental system identification based on several measurement locations along the beam results in a number of system models providing the spatial nature of the beam. Comparison of each experimental and analytical system models in the frequency domain yields a significant improvement on the determination of the natural frequencies and helps one to identify the uncertainty on them.

Also, tuning the modal damping ratios until the magnitude of both frequency responses coincide at resonance frequencies gives valid damping values and the corresponding uncertainty for each modal damping ratio.

This study also presented the active vibration control of the smart beam. A spatial H_∞ controller was designed for suppressing the first two flexural vibrations of the smart beam. The efficiency of the controller was demonstrated both by simulations and experimental implementation. The effectiveness of spatial controller on suppressing the vibrations of the smart beam over its entire body was also compared with a pointwise one.



a) Within excitation of 5-8 Hz

b) Within excitation of 40-44 Hz

Fig. 19. Time responses of the smart beam

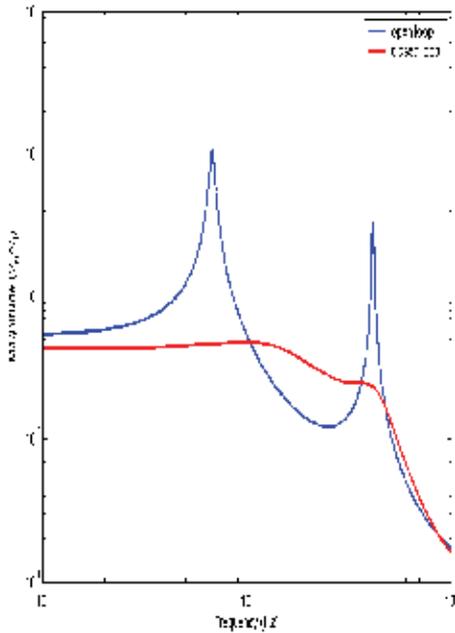


Fig. 20. Open and closed loop frequency responses of the smart beam

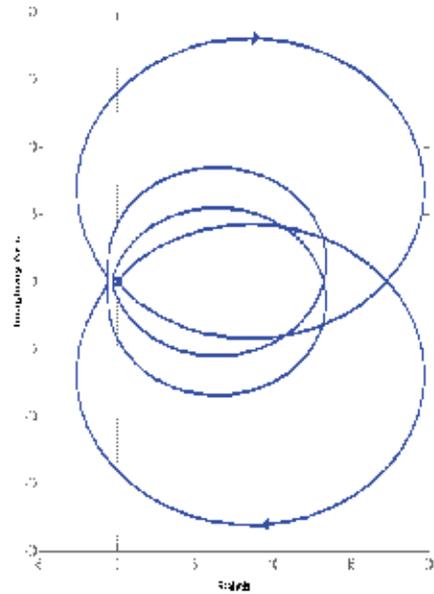


Fig. 21. Nyquist plot

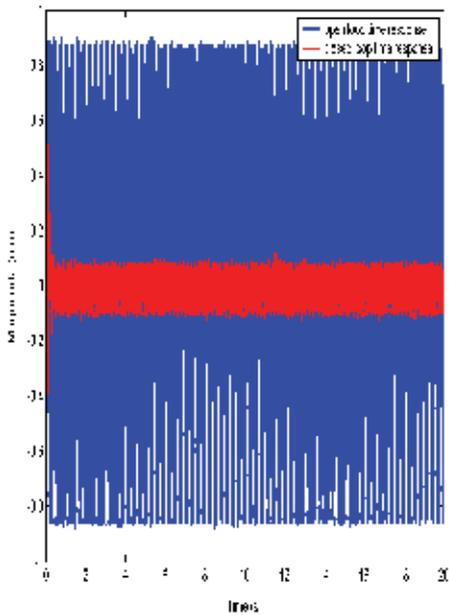


Fig. 21. Open and closed loop time responses at first resonance frequency

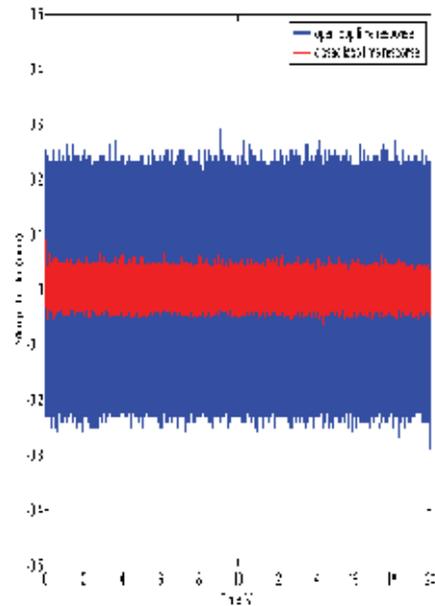


Fig. 22. Open and closed loop time responses at second resonance frequency

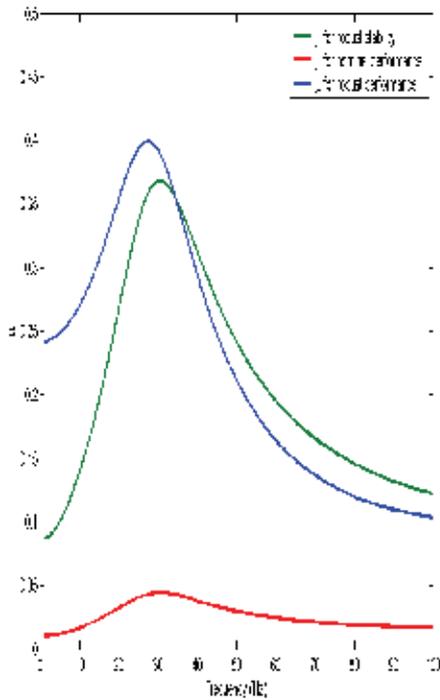


Fig. 23. μ -analysis for spatial H_∞ controller

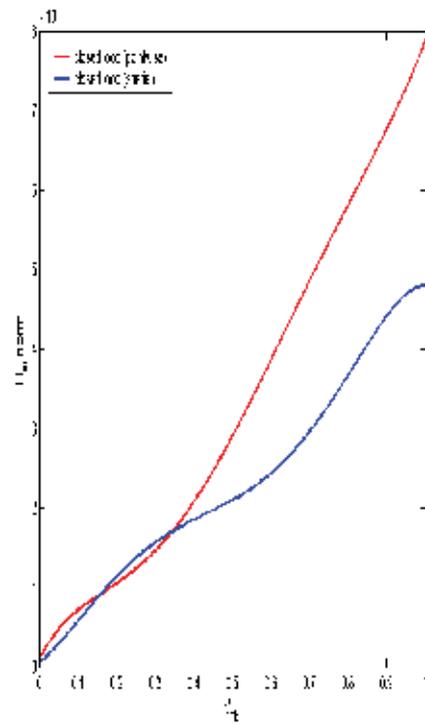


Fig. 24. Simulated H_∞ norm plots of closed loop systems under the effect of pointwise and spatial H_∞ controllers

6. References

- Bai M., Lin G. M., (1996). The Development of DSP-Based Active Small Amplitude Vibration Control System for Flexible Beams by Using the LQG Algorithms and Intelligent Materials, *Journal of Sound and Vibration*, 198, Vol. 4, pp. 411-427.
- Balas G., Young P. M., (1995). Control Design For Variations in Structural Natural Frequencies, *Journal of Guidance, Control and Dynamics*, Vol. 18, No. 2.
- Baz A., Poh S., (1988). Performance of an Active Control System with Piezoelectric Actuators, *Journal of Sound and Vibration*, 126(2), pp. 327-343.
- Clark R.L., (1997). Accounting for Out-of-Bandwidth Modes in the Assumed Modes Approach: Implications on Colocated Output Feedback Control, *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, Vol. 119, pp. 390-395.
- Crawley E.F., Louis J., (1989). Use of Piezoelectric Actuators as Elements of Intelligent Structures, *AIAA Journal*, Vol. 125, No. 10, pp. 1373-1385
- Çalışkan T., (2002). *Smart Materials and Their Applications in Aerospace Structures*, PhD Thesis, Middle East Technical University.

- Dimitridis E.K.C., Fuller R., Rogers C. A., (1991). Piezoelectric Actuators for Distributed Vibration Excitation of Thin Plates, *Journal of Vibration and Acoustics*, Vol. 113, pp. 100-107.
- Doyle J. C., Glover K., Khargonekar P.P., Francis B. A., (1989). State-Space Solutions to Standard and Control Problems, *IEEE Transactions on Automatic Control*, 34(8), pp. 831-847.
- Francis B. A., Zames G., (1984). On Optimal Sensitivity Theory for SISO Feedback Systems, *IEEE Transactions on Automatic Control*, Vol. AC-29, No.1.
- Gosavi S.V., Kelkar A.G., (2004). Passivity-Based Robust Control of Piezo-Actuated Flexible Beam, *Journal of Vibration and Acoustics*, Vol. 126, pp. 260-271.
- Halim D., Moheimani S.O.R., (2002a). Experimental Implementation of Spatial Control on a Piezoelectric Laminate Beam, *IEEE/ASME Transactions on Mechatronics*, Vol. 7, No:3.
- Halim D., (2002b). *Vibration Analysis and Control of Smart Structures*, PhD. Thesis, School of Electrical Engineering and Computer Science, University of Newcastle, Australia.
- Halim D., Moheimani S.O.R., (2002c). Spatial Control of a Piezoelectric Laminate Beam: Experimental Implementation, *IEEE Transactions on Control System Technology*, Vol. 10, No:4
- Hanagoud S., Obal M. W., Calise A. J., (1992). Optimal Vibration Control by the Use of Piezoceramic Sensors and Actuators, *Journal of Guidance, Control and Dynamics*, Vol. 15, No. 5, pp. 1199-1206.
- Hughes P.C., (1987). Space Structure Vibration Modes: How Many Exist? Which Ones Are Important?, *IEEE Control Systems Magazine*, pp. 22-28.
- Hughes P.C., Skelton R.E., (1981). Modal Truncation for Flexible Spacecraft, *Journal of Guidance and Control*, Vol. 4, No.3.
- Kırçalı Ö.F., Yaman Y., Nalbantoğlu V., Şahin M., Karadal F.M., (2008). Spatial Control of a Smart Beam, *Journal of Electroceramics*, 20(3-4): 175-185
- Kırçalı Ö.F., Yaman Y., Nalbantoğlu V., Şahin M., Karadal F.M., (2007). Application of Spatial H_∞ Control Technique for Active Vibration Control of a Smart Beam, *ICINCO 2007, International Conference on Informatics in Control, Automation and Robotics*, Paper C3-425, Angers, France
- Kırçalı Ö.F., (2006a). *Active Vibration Control of a Smart Beam: A Spatial Approach*, M.S. Thesis, Middle East Technical University.
- Kırçalı Ö.F., Yaman Y., Nalbantoğlu V., Şahin M., Karadal F.M., (2006b). Comparison of Spatial and Pointwise Controllers in Vibration Control of a Smart Beam, *I. National Aerospace Symposium*, METU, Ankara, Turkey (in Turkish)
- Lee Y.K., (2005). *Active Vibration Control of a Piezoelectric Laminate Plate Using Spatial Control Approach*, M.S. Thesis, Department of Mechanical Engineering, University of Adelaide, Australia
- Lenz K., Özbay H., (1993). Analysis and Robust Control Techniques for an Ideal Flexible Beam, *Control and Dynamic Systems*, 57: 369-421
- Ljung L., (1987). *System Identification: Theory for the User*, Prentice-Hall.
- Mason W.P., (1981). Piezoelectricity, its History and Applications, *Journal of Acoustical Society of America*, Vol. 70, No. 6
- Meirovitch L., (1986). *Elements of Vibration Analysis*, The McGraw-Hill Company
- Moheimani S.O.R., Halim D., Fleming A.J., (2003). *Spatial Control of Vibration. Theory and Experiments*, World Scientific Publishing Co. Pte. Ltd.

- Moheimani S.O.R., (2000a). Minimizing the Effect of Out-of-Bandwidth Dynamics in the Models of Reverberant Systems That Arise in Modal Analysis: Implications on Spatial Control, *Automatica*, Vol. 36, pp. 1023-1031
- Moheimani S.O.R., (2006b). Experimental Verification of the Corrected Transfer Function of a Piezoelectric Laminate Beam, *IEEE Transactions on Control Systems Technology*, Vol. 8, No. 4, pp. 660-666
- Moheimani S.O.R., Clark R.L., (2000c). Minimizing the Truncation Error in Assumed Modes Models of Structures, *Transactions of the ASME, Journal of Vibration and Acoustics*, Vol. 122, pp.332-335
- Moheimani S.O.R., (2000d). Minimizing the Effect of out of Bandwidth Modes in Truncated Structure Models, *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, Vol. 122, pp.237-239
- Moheimani S.O.R., Petersen I.R., Pota H.R., (1999). Broadband Disturbance Attenuation over an Entire Beam, *Journal of Sound and Vibration*, 227(4): 807-832
- Moheimani S.O.R., Pota H.R., Petersen I.R., (1998a). Spatial Control for Active Vibration Control of Piezoelectric Laminates, *International Proceedings of 37th IEEE CDC*, Tampa, Florida, USA, December 1998
- Moheimani S.O.R., Pota H.R., Petersen I.R., (1998b). Active Control of Noise and Vibration in Acoustic Ducts and Flexible Structures – a Spatial Control Approach, *Proceedings of ACC*, Philadelphia, Pennsylvania, USA, June 1998
- Moheimani S.O.R, Fu M., (1998c). Spatial Norm of Flexible Structures and its Application in Model Order Selection, *International Proceedings of 37th IEEE Conference on Decision and Control*, Tampa Florida, USA, 1998
- Moheimani S.O.R., Pota H.R., Petersen I.R., (1997). Spatial Balanced Model Reduction for Flexible Structures, *Proceedings of the American Control Conference*, pp. 3098-3102, Albuquerque, New Mexico, June 1997
- Nalbantoğlu V., Bokor J., Balas G., Gaspar P., (2003). System Identification with Generalized Orthonormal Basis Functions: an Application to Flexible Structures, *Control Engineering Practice*, Vol. 11, pp.245-259
- Nalbantoğlu V., (1998). *Robust Control and System Identification for Flexible Structures*, Ph. D. Thesis, University of Minnesota
- Pota H.R., Alberts T.E., (1992). Multivariable Transfer Functions for a Slewing Piezoelectric Laminate Beam, *Proceedings IEEE International Conference on Systems Engineering*, Japan, September 1992
- Prasad S.E., Ahmad A., Wheat T.A., (1998). The Role of Smart Structures in Robotics, *Canada-US CanSmart Workshop on Smart Materials and Structures*, Quebec, Canada Proceedings pp:133-147, 1998
- Reinelt W., Moheimani S.O.R., (2002). Identification of a Flexible Beam, *In Proceedings of the 8th International Mechatronics Conference*, Enschede, Netherlands, 2002
- Ülker F.D., (2003). *Active Vibration Control of Smart Structures*, M.S. Thesis, Middle East Technical University
- Yaman Y., Ülker F.D., Nalbantoğlu V., Çalışkan T., Prasad E., Waechter D., Yan B., (2003a). Application of μ -Synthesis Active Vibration Control Technique to a Smart Fin, *6th CanSmart, International Workshop on Smart Materials and Structures*, Montreal, Canada Proceedings pp:109-118, 2003

- Yaman Y., Ülker F. D., Nalbantoğlu V., Çalışkan T., Prasad E., Waechter D., Yan B., (2003b). Application of Active Vibration Control Strategy in Smart Structures, *AED2003, 3rd International Conference on Advanced Engineering Design*, Paper A5.3, Prague, Czech Republic, 01-04 June, 2003
- Yaman Y., Çalışkan T., Nalbantoğlu V., Prasad E., Waechter D., (2002a). Active Vibration Control of a Smart Plate, *ICAS2002, International Council of the Aeronautical Sciences*, Paper 424, Toronto, Canada, September 8-13, 2002
- Yaman Y., Çalışkan T., Nalbantoğlu V., Ülker F. D., Prasad E., Waechter D., Yan B., (2002b). Active Vibration Control of Smart Plates by Using Piezoelectric Actuators, *ESDA2002, 6th Biennial Conference on Engineering Systems Design and Analysis*, Paper APM-018, Istanbul, Turkey, July 8-11, 2002
- Yaman Y., Çalışkan T., Nalbantoğlu V., Prasad E., Waechter D., Yan B., (2001). Active Vibration Control of a Smart Beam, *Canada-US CanSmart Workshop on Smart Materials and Structures*, Montreal, Canada Proceedings pp:137-147, 2001
- Zames G., (1981). Feedback and Optimal Sensitivity: Model Reference Transformations, Multiplicative Seminorms, and Approximate Inverses, *IEEE Transactions on Automatic Control*, Vol. AC-26, No.2, April 1981
- Zhou G., Doyle J.C., (1998). *Essentials of Robust Control*, Prentice-Hall

Time-scaling in the control of mechatronic systems

Bálint Kiss and Emese Szádeczky-Kardoss
Budapest University of Technology and Economics
Hungary

1. Introduction

Time-scaling is not a new concept in the theory of dynamical systems. It has been used to modify the time distribution along the reference paths and also to transform a system by changing the clock with which it evolves. The motivation to introduce time-scaling is often to gain useful properties for the system which evolves according to the modified time. It is shown in (Sampei & Furuta, 1986) that such a property to gain with time-scaling may be feedback linearizability.

The notion of orbital flatness introduced by Fliess et al. (Fliess et al., 1995, 1999) involves also time-scaling to define an equivalence between a class of nonlinear systems and finite chains of integrators. The problem to check the orbital flatness of single input systems is addressed in (Respondek, 1998) and (Guay, 1999) together with the well known example of the kinematic car with constant longitudinal velocity which is shown to be orbitally flat.

The time-scaling introduced by these concepts involves the state variables to express the relation between the different time scales, hence the time-scaling does not involve any new input or variable external to the system.

Another concept for time-scaling is to use the tracking error in closed loop to modify the time-scaling of the reference path (Lévine, 2004). Such methods change the traveling time of the reference path according to the actual tracking error by decelerating if the motion is not accurate enough and by accelerating if the errors are small or vanish.

This chapter presents a new time-scaling scheme which is not driven only by the state variables of the system but also by a new input, referred to as the time-scaling input. In the setup suggested in this chapter, the new input variable, which is not an input of the original physical system, is also used to drive the time-scaling of the reference in closed loop.

The usefulness of our approach is demonstrated for the nonholonomic model of the kinematic car with one input. Notice that solutions to the motion planning and tracking algorithms are reported to the kinematic car with two inputs (Cuesta and Ollero, 2005) exploiting its differentially flatness property or using other methods (Dixon et al., 2001). We show that the kinematic car with one input, such that the longitudinal velocity does not vanish, can track any smooth trajectory with non-vanishing longitudinal velocity such that the tracking error is reduced exponentially along the path. This is achieved using time-scaling and a dynamical feedback similar to the differentially flat case.

The remaining part of the chapter is organized as follows. The next section introduces our new time-scaling concept in details and its application in some general cases. Section 3 addresses the particular problem of the control of a car using its kinematic model such that the only control input is the angle of the steered wheels. Section 4 presents the simulation and real measurement results obtained by the application of the controllers described in Section 3. The conclusion is given in Section 5.

2. The time-scaling concept

This section introduces in a general context our novel time-scaling scheme and shows how dynamical systems are transformed by its application. For this reason, consider a finite dimensional and time invariant dynamical system given by its state equation

$$\frac{d\xi}{dt} = \Phi(\xi, u) \quad (1)$$

where $\xi \in R^n$, $u \in R^m$ are the state vector and the input vector, respectively. This system evolves according to the time t which we refer to as the real time. Let τ denote the scaled time. A time-scaling law is an invertible mapping

$$t \mapsto \tau(t) \quad \tau \mapsto t(\tau) . \quad (2)$$

The time-scaling scheme proposed by Sampei and Furuta (Sampei & Furuta, 1986) depends on the state of the system

$$\frac{dt}{d\tau} = s(\xi) \quad \tau(t_0) = \tau_0 \quad (3)$$

and the authors show that the system rewritten according to the time τ , namely

$$\frac{d\xi}{d\tau} = \frac{d\xi}{dt} \frac{dt}{d\tau} = s(\xi)\Phi(\xi, u) \quad (4)$$

may exhibit properties which were not satisfied by the system evolving according to the time t . Such a property studied in the paper is feedback linearizability. Let us point out that the time-scaling defined by (3) does not change the number of inputs of the system. Instead of (3) we introduce a time-scaling concept which increases the number of inputs of the system with a new input referred to as a time-scaling (or simply scaling) input. Hence the time-scaling is driven by

$$\frac{dt}{d\tau} = \sigma(\xi, u_s) \quad \tau(t_0) = \tau_0 . \quad (5)$$

This time-scaling results a scaled dynamics

$$\frac{d\xi}{d\tau} = \sigma(\xi, u_s) \Phi(\xi, u) \quad (6)$$

with the inputs u and u_s . The same time-scaling can be applied for systems with more specific form of state equation. Considering a driftless system with the state equation

$$\frac{d\xi}{dt} = \sum_{i=1}^m g_i(\xi) u_i \quad (7)$$

the time-scaling (5) results

$$\frac{d\xi}{d\tau} = \sum_{i=1}^m \sigma(\xi, u_s) g_i(\xi) u_i \quad (8)$$

The time-scaling defined by (5) can be generalized by the introduction of a chain of integrators evolving according to the time τ

$$\frac{dt}{d\tau} = s_1 \quad \frac{ds_1}{d\tau} = s_2 \quad \dots \quad \frac{ds_k}{d\tau} = \sigma(s, \xi, u_s) \quad \tau(t_0) = \tau_0 \quad (9)$$

with $s = (s_1, s_2, \dots, s_k)$. Let us now consider the dynamical model of a mechatronic system with d degrees of freedom such that the d generalized coordinates are the elements of the vector q . Such a dynamics reads

$$F_i = \sum_{j=1}^d D_{ij} \ddot{q}_j + \sum_{j=1}^d \sum_{k=1}^d D_{ijk} \dot{q}_j \dot{q}_k + D_i \quad i = 1 \dots d \quad (10)$$

where D_{ij} are the elements of the inertia tensor, D_{ijk} are the coefficients of the centripetal and Coriolis effects, D_i contain the gravitational forces, and F_i give the generalized

external forces. We use the standard notations $\dot{q} = \frac{dq}{dt}$; $\ddot{q} = \frac{d^2q}{dt^2}$ and introduce $q' = \frac{dq}{d\tau}$;

$q'' = \frac{d^2q}{d\tau^2}$. Using simple derivation rules and (9) for $k=1$, the relation between the time derivatives with respect to t and τ read

$$\dot{q} = q' \frac{d\tau}{dt} \quad \ddot{q} = q'' \left(\frac{d\tau}{dt} \right)^2 + q' \frac{d^2\tau}{dt^2} \quad \frac{d^2\tau}{dt^2} = \frac{1}{dt} \frac{1}{s_1} = \frac{d}{d\tau} \left(\frac{1}{s_1} \right) \frac{d\tau}{dt} = -\frac{\sigma(s_1, \xi, u_s)}{s_1^3} \quad (11)$$

which result the following time-scaled system

$$F_i = \sum_{j=1}^d D_{ij} \left(q'' s_1^{-2} - q' \frac{\sigma(s_1, \xi, u_s)}{s_1^3} \right) + \sum_{j=1}^d \sum_{k=1}^d D_{ijk} q'_j q'_k s_1^{-2} + D_i \quad i=1 \dots d \quad (12)$$

subject to (9) and with the state vector $\xi = (q, \dot{q})$.

Another possibility for the generalization is to take into account in the model of the dynamical system the effect of some external, but measureable signals. Such signals can be measureable (or estimable) disturbances or one may think of them as input signals which are not generated by the controller but by some other means, e.g. by a human operator. We will denote the vector of these signals by w which is included in the state equation and can be also incorporated in the time-scaling

$$\frac{d\xi}{dt} = \Phi(\xi, w, u) \quad \frac{dt}{d\tau} = \sigma(\xi, w, u_s) \quad \tau(t_0) = \tau_0 \quad (13)$$

to result a scaled dynamics similar to (6)

$$\frac{d\xi}{d\tau} = \sigma(\xi, w, u_s) \Phi(\xi, w, u) \quad (14)$$

Notice that one reason for the use of the time-scaling can be the elimination of the effect of the external signals in the vector w .

It is important to note that the time-scaling must not rewind the time, hence (5), (9), and (13) have to be chosen such that the resulting time-scaling law (2) is monotonous.

3. Case study: semi-autonomous maneuvering with a passenger car

The time-scaling concept described in the previous section will be applied to the design of a semi-autonomous maneuvering feature for a passenger car. Semi-autonomy means in this case that the controller does not generate all variables that influence the motion of the car, but only a subset of such variables. To be more specific, the driver generates the velocity of the car by the proper actuation of the gas, break, and clutch pedals while the control system determines the angle of the steering wheel. This implies that the geometry of the trajectory can be influenced, but not the traveling time which is needed to complete a given section of the trajectory. Hence the application of the time-scaling introduced in the previous section seems to be natural such that the control objective is to follow a path with the car as the controller can eliminate the geometric error between the desired and the real trajectory.

3.1 Car model

We use the simple kinematic model of the car, also referred to as the one-track or bicycle model in the literature. The model is illustrated in Figure 1 such that the bicycle is fitted on the longitudinal symmetry axis of the car. We suppose that the Ackermann steering

geometry assumption holds true such that all wheels turn around the same point P which is on the rear axle of the car.

The motion of the car is described by the time functions of its (rear axle midpoint) position (x, y) and its orientation θ . The axle-space of the car is denoted by l . Two variables influence the motion of the car in the horizontal plane: the longitudinal velocity, denoted by v_{car} , is generated by the human driver ($w = v_{car}$) and the steering angle φ is the only control input. The kinematic model reads

$$\begin{aligned}\dot{x} &= v_{car} \cos \theta \\ \dot{y} &= v_{car} \sin \theta \\ \dot{\theta} &= \frac{v_{car}}{l} \tan \varphi\end{aligned}\quad (15)$$

It is known from the literature that this model (and its generalized version with trailers) is differentially flat if both v_{car} and φ are control inputs (Fliess et al., 1995, 1999).

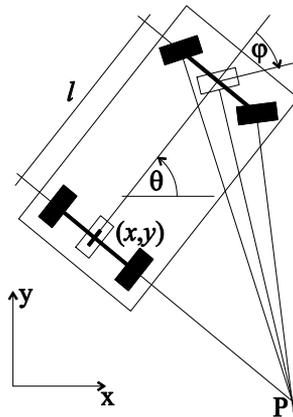


Fig. 1. Bicycle model of a car moving in the horizontal plane

It is also known that for $v_{car} = 1$ the same model with one input is orbitally flat (Guay, 1999 and Respondek, 1998). Orbital flatness implies the equivalence of the system (15) to a chain of integrators using a transformation which involves dynamic state feedback, coordinate transformation, and time-scaling such that this transformation leaves the number of inputs unchanged.

Using the time-scaling scheme introduced in the previous section we explicitly show how to stabilize a given trajectory for an arbitrary non-vanishing velocity profile v_{car} .

3.2 Model transformation using time-scaling

Since we use a purely kinematic model of the car, the introduction of a time-scaling similar to (13) is sufficient. Moreover, the state variables are not necessary to be included in

the expression which reads in our case (Kiss and Szádezcky-Kardoss, 2007)

$$\frac{dt}{d\tau} = \frac{u_s}{v_{car}} \quad (16)$$

assuming that v_{car} does not vanish during the trajectory. If v_{car} and u_s are both positive (respectively negative) than the resulting time-scaling law does not rewind the time. The dynamics after the application of the time-scaling is given by

$$\begin{aligned} x' &= u_s \cos \theta \\ y' &= u_s \sin \theta \\ \theta' &= \frac{u_s}{l} \tan \varphi \end{aligned} \quad (17)$$

This dynamics, which evolves according to the time τ , has two inputs (u_s and φ) and it is known to be differentially flat which implies that it is feedback linearizable by dynamic state feedback such that the flat output contains two variables, namely the (x, y) position of the car.

3.3 Asymptotic stabilization of the reference trajectory (according to the time τ)

Suppose that a reference trajectory (x_{ref}, y_{ref}) is given for the position of the car. A feedback law should be found such that the reference trajectory is asymptotically (or eventually exponentially) tracked. Since the human driver generates the longitudinal velocity of the car, the reference trajectory cannot be designed according to the real time t but only according to some "virtual" time τ .

To understand this fact, consider a reference path with a length of 100 meters and suppose that the time functions $(x_{ref}(\cdot), y_{ref}(\cdot))$ are determined such that the desired travelling time along the trajectory should be 10 seconds which is a 10 m/s average speed along the path. If the driver generates a constant longitudinal velocity which equals to 20 m/s than the real traveling time (supposing that the tracking is perfect) will be 5 seconds. Similarly, if the driver generates a constant longitudinal velocity which equals to 5 m/s than the real traveling time (supposing again perfect tracking of the geometry of the path) will be 20 seconds. Moreover, the velocity profile for a given maneuver is not know in advance so the time-scaling must use the current velocity value generated by the driver and eventually its time derivatives.

It follows that one cannot design a controller which ensures a desired travelling time, according to the real time t , but it is possible with respect to the "virtual" time τ . So we will suppose that the reference $(x_{ref}(\tau), y_{ref}(\tau))$ is given according to τ and the designed controller is able to ensure the asymptotic stabilization according to the same virtual time τ . The reference is given by the mapping

$$\begin{aligned}\tau &\mapsto \{x_{ref}, x'_{ref}, x''_{ref}, x'''_{ref}\} \\ \tau &\mapsto \{y_{ref}, y'_{ref}, y''_{ref}, y'''_{ref}\}\end{aligned}\quad (18)$$

for $\tau \in [0, T]$ where T is the desired traveling time along the trajectory according to the time τ . The control loop is depicted in Figure 2.

Let us define the tracking error as $e_x = x - x_{ref}$ and $e_y = y - y_{ref}$. The closed loop system must guaranty the exponential decay of the tracking errors according to the time τ so the following equations

$$\begin{aligned}k_{x,0}e_x + k_{x,1}e'_x + k_{x,2}e''_x + e'''_x &= 0 \\ k_{y,0}e_y + k_{y,1}e'_y + k_{y,2}e''_y + e'''_y &= 0\end{aligned}\quad (19)$$

must be satisfied such that the coefficients $k_{a,i}$ ($a \in \{x, y\}, i = 0, 1, 2$) are design parameters and have to be chosen such that the characteristic polynomials of (19) are Hurwitz.

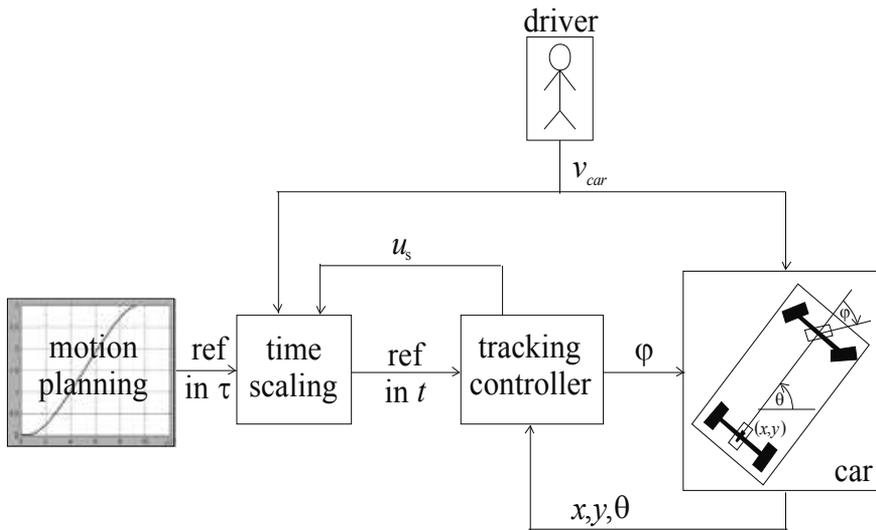


Fig. 2. Scheme of the control loop including the tracking controller and the time-scaling

The following calculations are based on the differential flatness property of (17) which implies that the model can be linearized by a dynamic state feedback.

In order to satisfy (19) we first add some integrators in front of both inputs of (17). These integrators are realized in the controller and the new inputs of the integrator chains are v_1 and v_2 .

$$\begin{aligned}\zeta'_1 &= \zeta_2 = u'_s & \zeta'_3 &= v_2 \\ \zeta'_2 &= v_1 = u''_s & \varphi &= \zeta_3 \\ u_s &= \zeta_1\end{aligned}\quad (20)$$

Note that one could add more integrators and the minimal number of integrators such that the calculations that follow can be carried out is one preceding the u_s input. The chains of integrators are illustrated in Figure 3.

Using the states of (17) and the states of the dynamic extension (20), the derivatives x' , x'' , x''' , y' , y'' , y''' can be determined.

$$x' = \zeta_1 \cos \theta \quad (21)$$

$$x'' = \zeta_2 \cos \theta - \frac{\zeta_1^2 \sin \theta \tan \zeta_3}{l} \quad (22)$$

$$x''' = v_1 \cos \theta - \frac{3\zeta_1 \zeta_2 \sin \theta \sin \zeta_3}{l \cos \zeta_3} - \frac{\zeta_1^3 \cos \theta}{l^2 (\cos \zeta_3)^2} + \frac{\zeta_1^3 \cos \theta}{l^2} - \frac{v_2 \zeta_1^2 \sin \theta}{l (\cos \zeta_3)^2} \quad (23)$$

$$y' = \zeta_1 \sin \theta \quad (24)$$

$$y'' = \zeta_2 \sin \theta + \frac{\zeta_1^2 \cos \theta \tan \zeta_3}{l} \quad (25)$$

$$y''' = v_1 \sin \theta + \frac{3\zeta_1 \zeta_2 \cos \theta \sin \zeta_3}{l \cos \zeta_3} - \frac{\zeta_1^3 \sin \theta}{l^2 (\cos \zeta_3)^2} + \frac{\zeta_1^3 \sin \theta}{l^2} + \frac{v_2 \zeta_1^2 \cos \theta}{l (\cos \zeta_3)^2} \quad (26)$$

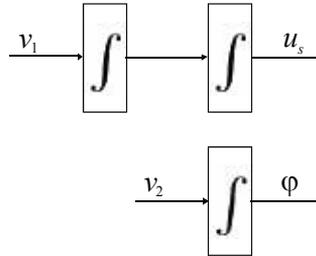


Fig. 3. Integrators (dynamic extension) in the controller

The inputs of the integrator chains of Figure 3 (v_1 and v_2) appear in the expressions of x''' (23) and y''' (26). This implies that one can calculate them such that the linear differential equations for the tracking errors (19) are satisfied. To achieve this, isolate first x''' and y''' from (19) to obtain

$$\begin{aligned} x''' &= x'''_{ref} - k_{x,0}e_x - k_{x,1}e'_x - k_{x,2}e''_x = \omega_x \\ y''' &= y'''_{ref} - k_{y,0}e_y - k_{y,1}e'_y - k_{y,2}e''_y = \omega_y \end{aligned} \quad (27)$$

Combining (23) and (26) with (27) on gets

$$\begin{bmatrix} \cos \theta & -\frac{\zeta_1^2 \sin \theta}{l(\cos \zeta_3)^2} \\ \sin \theta & \frac{\zeta_1^2 \cos \theta}{l(\cos \zeta_3)^2} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \omega_x - A \\ \omega_y - B \end{bmatrix} \quad (28)$$

with

$$\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} -\frac{3\zeta_1\zeta_2 \sin \theta \sin \zeta_3}{l \cos \zeta_3} - \frac{\zeta_1^3 \cos \theta}{l^2(\cos \zeta_3)^2} + \frac{\zeta_1^3 \cos \theta}{l^2} \\ \frac{3\zeta_1\zeta_2 \cos \theta \sin \zeta_3}{l \cos \zeta_3} - \frac{\zeta_1^3 \sin \theta}{l^2(\cos \zeta_3)^2} + \frac{\zeta_1^3 \sin \theta}{l^2} \end{bmatrix} \quad (29)$$

Finally, the inputs v_1 and v_2 are obtained by the inversion of the coefficient matrix in (28)

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\frac{l \sin \theta (\cos \zeta_3)^2}{\zeta_1^2} & \frac{l \cos \theta (\cos \zeta_3)^2}{\zeta_1^2} \end{bmatrix} \begin{bmatrix} \omega_x - A \\ \omega_y - B \end{bmatrix} \quad (30)$$

The coefficient matrix is singular if $\zeta_1 = u_s = 0$ which occurs at zero longitudinal velocity. Another singularity occurs if $\zeta_3 = \varphi = \pm 90^\circ$. Singularities coincide with the loss of controllability of the model.

Based on the previous calculations, the reference trajectory (18), the states of the kinematic car (17) and the states of the dynamic extension (20) allow determining the inputs v_1 and v_2 of the system extended by the integrators and thus to obtain the time-scaling input u_s and the steering angle φ which results the exponential stabilization of the reference trajectory.

3.4 Time-scaling of time functions

Expressions (21)-(26) hold true if the variables on the left hand sides are functions of τ . Notice however that the measured states of the car evolve with the real time t so the time derivatives with respect to τ cannot be directly determined. Therefore one needs the transformation which allows mapping a time function of t and its derivatives to a time function of τ and to its derivatives using the time-scaling law specified by (16). Consider a variable α . We wish to obtain the mapping

$$\{\alpha(t), \dot{\alpha}, \ddot{\alpha}, \dots\} \leftrightarrow \{\alpha(\tau), \alpha', \alpha'', \dots\} \quad (31)$$

The time-scaling law (2) can be obtained by the integration of (16) and reads in our case

$$\tau(t) = \int_0^t \frac{v_{car}}{u_s} d\vartheta \quad \tau(0) = t(0) = 0 \quad (32)$$

which allows expressing the transformation (31) as

$$\alpha(t) = \alpha(\tau(t)) \quad (33)$$

$$\dot{\alpha}(t) = \alpha'(\tau(t))\dot{\tau} \quad (34)$$

$$\ddot{\alpha}(t) = \alpha''(\tau(t))\dot{\tau}^2 + \alpha'(\tau(t))\ddot{\tau} \quad (35)$$

$$\dddot{\alpha}(t) = \alpha'''(\tau(t))\dot{\tau}^3 + 3\alpha''(\tau(t))\ddot{\tau}\dot{\tau} + \alpha'(\tau(t))\ddot{\tau} \quad (36)$$

The successive time derivatives ($\dot{\tau}, \ddot{\tau}, \ddot{\tau}$) of the time-scaling law (32) can be also obtained using the time derivatives of v_{car} and the time derivatives of u_s .

$$v_{car} = \dot{u}_s \quad (37)$$

$$\dot{v}_{car} = \ddot{u}_s + \dot{\tau}^2 u_s' \quad (38)$$

$$\ddot{v}_{car} = \ddot{\tau} u_s + 3\dot{\tau} \ddot{u}_s' + \dot{\tau}^3 u_s'' \quad (39)$$

We suppose that the time derivatives of v_{car} can be measured or estimated. The time derivatives of u_s are the states of the dynamic extension (20).

3.5 Stabilization using linearized error dynamics

The method presented in the previous subsection required the time derivatives of v_{car} but their measurement or estimation may present difficulties. We propose therefore an alternative stabilization technique which overcomes this difficulty. The price to pay is that the stability will be only locally achieved since the feedback is based on the linearized error dynamics. The method is derived from the results presented in (Dixon et al., 2001).

We suppose that the reference time functions of all variables ($x_{ref}, y_{ref}, \theta_{ref}, u_{s,ref}, \varphi_{ref}$) are given such that they satisfy the model (17).

Let us define the error variables $e_x = x - x_{ref}$, $e_y = y - y_{ref}$, $e_\theta = \theta - \theta_{ref}$ and the error transformation

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} \quad (40)$$

which can be used to express the error dynamics as

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} 0 & \dot{\theta} & 0 \\ -\dot{\theta} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} 0 \\ \sin e_3 \\ 0 \end{bmatrix} u_{s,ref} \frac{v_{car}}{u_s} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \quad (41)$$

with the inputs w_1 and w_2 obtained as

$$w_1 = v_{car} - \frac{v_{car}}{u_s} u_{s,ref} \cos e_3 \quad (42)$$

$$w_2 = \frac{1}{l} \left(v_{car} \tan \varphi - \frac{v_{car}}{u_s} u_{s,ref} \tan \varphi_{ref} \right) \quad (43)$$

The expressions (42)-(43) also show how to calculate u_s , $\dot{\tau}$, and φ from w_1 and w_2 . The error dynamics (41) can be linearized around zero error and zero inputs (which implies that $\dot{\theta} = 0$ in (41)) and a state-feedback (pole-placement)

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = -K \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \quad (44)$$

can be used to stabilize the system around the reference trajectory.

4. Simulation and measurement results

Both stabilizing feedback laws presented in Section 3 are implemented in Matlab-Simulink¹ environment. The controller described in Subsection 3.3 is also implemented in a Ford Focus type passenger car using the rapid control prototyping environment comprising Matlab-Simulink, the Real-Time Workshop together with DSpace² Autobox hardware connected to the CAN bus of the car. The vehicle is equipped with a special prototype of an Electronic Power Assist Steering (EPAS) System developed by our industrial partner. This EPAS can receive and precisely track a reference for the steering angle (denoted by φ in our model) sent via the CAN bus of the vehicle.

4.1 Simulation results

For the simulation we use the simple kinematic model of the car as given by Equation (15) with $l = 1$ m. A lane-changing-like maneuver is the reference such that the lane-change

¹ <http://www.mathworks.com>

² <http://www.dspace.de>

represents a 3.5 meters shift to the lateral direction while advancing 10 meters. The motion planning was carried out with a default velocity profile which is referred to as the reference driver with a travelling time of 9 seconds. To demonstrate the time-scaling in simulation, we generated two additional velocity profiles such that the first is quicker and the second is slower than the reference driver. These three velocity profiles are depicted in Figure 4-a. The figure also shows the different traveling times necessary to complete the lane-changing maneuver for each velocity profile.

To check the asymptotic tracking property of the controller presented in Subsection 3.3 we set an initial position and orientation which is different from the ones used for motion planning such that the initial tracking errors are $e_x = -1.5$ m, $e_y = 2$ m, and $e_\theta = \pi/4$ rad. Due to time-scaling, the geometries of the trajectories in both cases (quick and slow drivers) are the same in the horizontal plane as shown in Figure 4-b. All trajectories converge to the reference despite the relatively large initial errors.

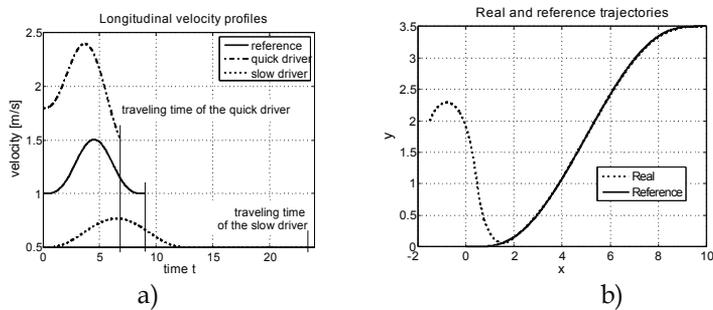


Fig. 4. a) Velocity profiles; b) Reference and real trajectories (same for all velocity profiles)

The time-scaling is shown in Figure 5. It can be seen that the same amount of “virtual” time τ (9 seconds) is needed to complete the maneuver but in terms of real time, the driver with quick velocity profile required around 7 seconds and the driver with slow velocity profile required more than 20 seconds completing the same trajectory.

Figure 5-b shows when the reference in τ was decelerated or accelerated with respect to the real time t . It is interesting to see that the reference was decelerated even for the quick driver while the tracking error was large enough.

The steering angles produced by the controller are shown in Figure 6.

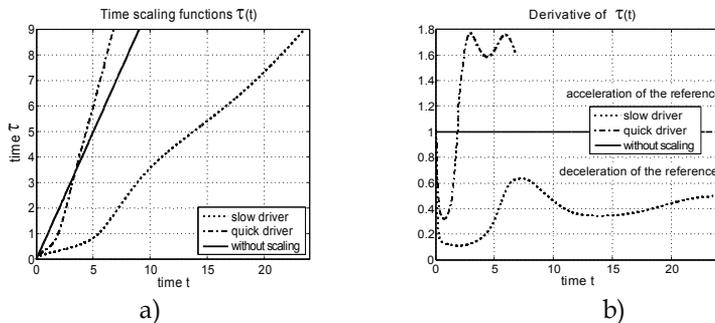


Fig. 5. Time-scaling for the different velocity profiles; a) function $\tau(t)$; b) function $\dot{\tau}(t)$

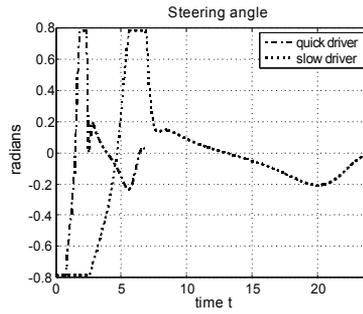


Fig 6. Steering angles during the maneuvers

We carried out similar simulation studies for the controller based on the linearized error dynamics described in Subsection 3.5. The controller locally stabilizes the reference trajectory so the initial errors were chosen to be smaller as in the previous case: $e_x = -0.5$ m, $e_y = 0.75$ m, and $e_\theta = \pi/4$ rad.

The velocity profiles and the geometry of the path are depicted in Figure 7. The time-scaling function $\tau(t)$ and its derivative are shown in Figure 8. Notice again that the time is decelerated first due to the relatively large tracking error.

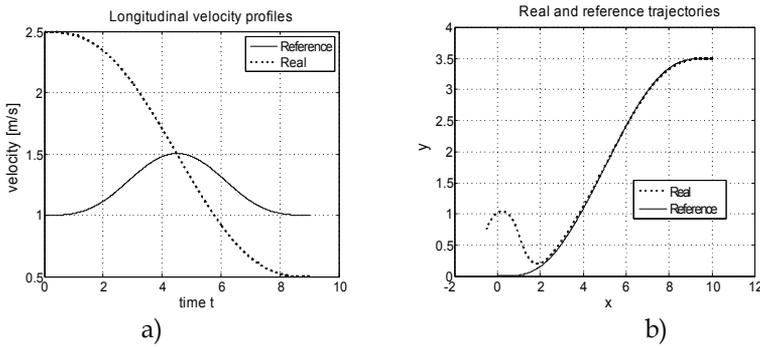


Fig. 7. Performance of the controller designed for the linearized error dynamics; a) velocity profiles; b) Real and reference trajectories in the horizontal plane

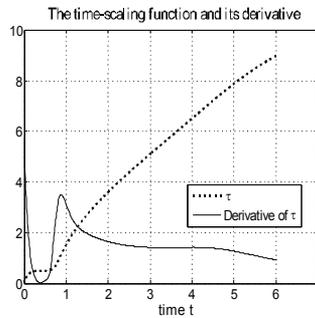


Fig. 8. Time-scaling function and its derivative.

4.2 Measurement results

The controller is implemented in a real passenger car with 10ms sampling time. Figure 9 shows the Ford Focus type test vehicle with the Autobox. The position and orientation of the car is estimated using the ABS wheel sensors the signals of which are available on the CAN bus of the vehicle. The description of the estimation algorithm is beyond the scope of this chapter, for similar algorithms the reader may refer to (Kochen at al., 2002).

The controller is applied here to track trajectories in parking scenarios. Two examples are studied. The first is the case of a parallel parking maneuver in backward direction where an initial position error was introduced as shown in Figure 10-a. The velocity profile generated by the driver (as measured on the CAN bus of the vehicle) and the steering wheel angle generated by the controller (in degrees) are depicted in Figure 10-b. Notice that the special EPAS which is built in the vehicle receives the steering wheel angle and not directly the angle denoted by φ .

The second example is a backward direction perpendicular parking maneuver (see Figure 11-a) with an initial error in position and orientation which is so large that the steering input is saturated during the motion as shown in Figure 11-b. Despite the saturation of the input, the reference trajectory is joined by the vehicle and the initial error is eliminated.



Fig. 9. The Ford Focus test vehicle; a) the DSpace Autobox in the carrier, connected to the CAN bus; b) a possible reference trajectory for a parking maneuver

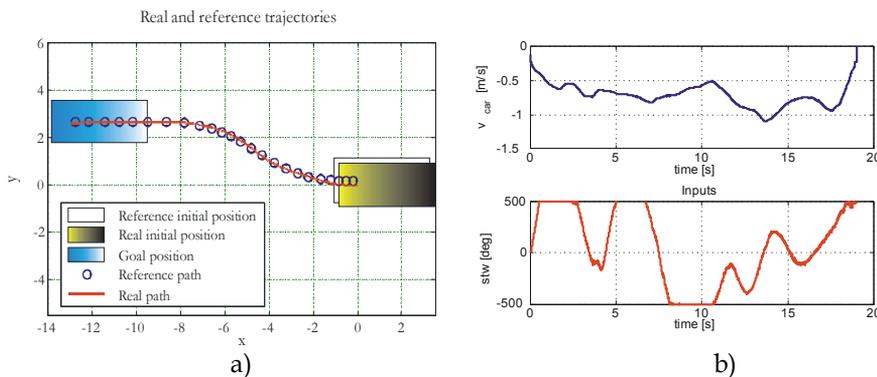


Fig. 10. Backward direction parallel parking maneuver with the Ford Focus; a) Trajectory in the horizontal plane; b) Driver's velocity profile and the steering wheel angle generated by the controller

5. Conclusion

This chapter presented a new time-scaling scheme applicable to the control of dynamical systems. The novelty resides in the fact that, in addition to the state variables, the time-scaling is influenced by a new variable which becomes an additional input of the time-scaled system. No general result is formulated at the time being concerning the properties of this time-scaling scheme but it turns out to be useful for a particular application, namely in the semi-autonomous control of a passenger car such that the velocity is generated by the driver but the steering action is determined by the controller.

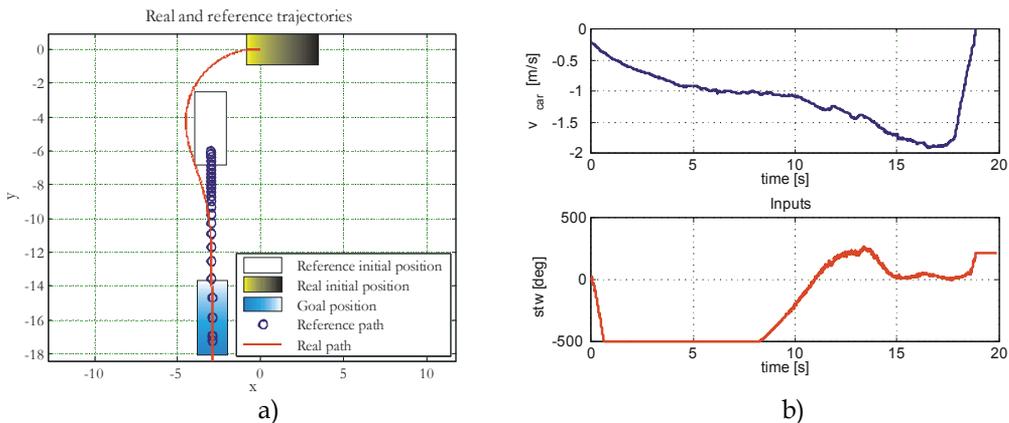


Fig. 11. Backward direction perpendicular parking maneuver with the Ford Focus; a) Trajectory in the horizontal plane; b) Driver's velocity profile and the steering wheel angle generated by the controller

6. Acknowledgements

The research was partially funded by the Hungarian Science Research Fund under Grant OTKA K71762 and by the Advanced Vehicles and Vehicle Control Knowledge Center under grant RET 04/2004.

7. References

- Cuesta, F. & Ollero., A. (2005). Intelligent Mobile Robot Navigation, ser. *Springer Tracts in Advanced Robotics*. Vol. 16, ISBN 3540239561, Springer, 2005.
- Dixon, W.E.; Dawson, D.M.; Zergeroglu, E. & Behal, A. (2001). Nonlinear control of wheeled mobile robots, *Lecture Notes in Control and Information Sciences*. Vol. 262, ISBN: 978-1-85233-414-7, Springer, 2001.
- Fliess, M.; Lévine, J. ; Martin, Ph. & Rouchon, P. (1995). Flatness and Defect of Nonlinear Systems: Introductory Theory and Examples. *International Journal of Control*, Vol. 61, No. 6, June, 1995, pp. 1327-1361, ISSN: 0020-7179

- Fliess, M.; Lévine, J. ; Martin, Ph. & Rouchon, P. (1999). A Lie-Bäcklund Approach to Equivalence and Flatness of Nonlinear Systems. *IEEE Transactions on Automatic Control*, Vol. 44, No. 5, May, 1999, pp. 922-937, ISSN 0018-9286
- Guay, M. (1999). An algorithm for orbital feedback linearization of single-input control affine systems. *Systems & Control Letters*, Vol. 38, No. 4, December, 1999, pp. 271-281, ISSN 0167-6911
- Kiss, B. & Szádeczky-Kardoss, E. (2007). Tracking control of the orbitally flat kinematic car with a new time-scaling input. *Proceedings of the 46th Conference on Decision and Control*, pp. 1969-1974, ISBN 1-4244-1498-9, New Orleans LA, December 2008, IEEE
- Kochen, M.; Neddenriep, R.; Isermann, R.; Wagner, N. & Hamann C-D. (2002). Accurate local vehicle dead-reckoning for a parking assistance system. *Proceedings of the American Control Conference*, pp. 4297-4302, ISBN 0-7803-7298-0, Anchorage AK, May 2002, IEEE
- Lévine, J. (2004). On the Synchronization of a Pair of Independent Windshield Wipers. *IEEE Transactions on Control Systems Technology*, Vol. 12, No. 5, 2004, pp. 787-795, ISSN 1558-0865
- Respondek, W. (1998). Orbital feedback linearization of single-input nonlinear control systems, *Proceedings of the 4th IFAC Nonlinear Control Systems Design Symposium*, pp. 499-504, ISBN: 0-08-043049-X, Enschede, The Netherlands, July 1998, IFAC
- Sampei, M. & Furuta, K. (1986). On Time Scaling for Nonlinear Systems: Application to Linearization. *IEEE Transactions on Automatic Control*, Vol. AC-31, No. 5, May, 1986, pp. 459-462, ISSN 0018-9286

Heap Models, Composition and Control

Jan Komenda, Sébastien Lahaye* & Jean-Louis Boimond*
*Institute of Mathematics, Czech Academy of Sciences, Žitkova 22, Brno
 Czech Republic*

**LISA, Université d' Angers, 62, Av. Notre Dame du Lac, Angers
 France*

1. Introduction

This chapter deals with modelling and control of discrete event systems. Discrete event systems (DES) are event driven man made systems whose evolution is guided by occurrences of asynchronous events as opposed to classical time driven discrete or continuous systems. DES are modelled by tools from computer science like automata, Petri nets and process algebras. There are two major streams in control theory of DES. The first stream, known as supervisory control theory, has been introduced by Wonham and Ramadge for logical automata, e.g. (Ramadge & Wonham,1989).

The second stream, more specialized, which uses the class of timed Petri nets, called timed event graphs is based on linear representation in the $(\max,+)$ algebra. It is of large interest to make a bridge between both approaches while generalizing both of them at the same time. Being inspired by papers on $(\max,+)$ automata (e.g. (Gaubert & Mairesse, 1999); (Gaubert, 1995), which generalize both logical automata and $(\max,+)$ -linear systems, it is interesting to develop a control method for $(\max,+)$ automata by considering supervisory control approach.

The time semantics of the parallel composition operation (called supervised product) we have proposed for control of $(\max,+)$ automata in (Komenda et al, 2007) are different from the standard time semantics for timed automata or timed Petri nets. One has to increase the number of clocks in order to define a synchronous product of $(\max,+)$ automata viewed as 1-clock timed automata. This goes in general beyond the class of $(\max,+)$ automata and makes powerful algebraic results for $(\max,+)$ automata difficult to use.

The results of (Gaubert & Mairesse, 1999) suggest however an alternative for the subclass of $(\max,+)$ automata corresponding to safe timed Petri nets, where synchronous product is standard composition of subnets through shared (synchronization) transitions. The intermediate formalism of heap models enables a letter driven $(\max,+)$ -linear representation of 1-safe timed Petri nets. Therefore it is interesting to work with heaps of pieces instead of $(\max,+)$ -automata and introduce a synchronous composition of heap models that yields essentially reduced nondeterministic $(\max,+)$ - automata representation of synchronous

composition of corresponding $(\max,+)$ -automata. This way we obtain representations allowing for use of powerful dioid algebras techniques and the reduced dimension of concurrent systems at the same time: the dimension of synchronous product of two heap models is the sum of each models dimensions, while the dimension of supervised product of $(\max,+)$ -automata is the product of the individual dimensions, which causes an exponential blow up of the number of states in the number of components.

The extension of supervisory control to timed DES represented by timed automata is mostly based on abstraction methods (for instance region construction turning a timed automaton into a logical one). On the other hand abstraction methods are not suitable for $(\max,+)$ or heap automata, because their timed semantics (when weights of transitions are interpreted as their minimal durations) are based on the earliest possible behavior similarly as for timed Petri nets. The method we propose avoids any abstraction and works with timed DES (TDES) represented by heap models.

Similarly as for logical DES our approach to supervisory control is based on the parallel composition (synchronous product) of the system with the supervisor (another heap model). Our research is motivated by applying supervisory control on heap models, which are appropriate to model 1-safe timed Petri nets. This is realized by using the synchronous product: the controlled system is the synchronous product of the system with its controller (another heap model). The algebraization of the synchronous product that is translated into idempotent sum of suitable block matrices together with a linear representation of composed heap models using its decomposed morphism matrix is then applied to the control problem for heap models.

This research work is organized as follows. Algebraic preliminaries needed throughout this chapter are recalled in Section 2. In Section 3 are introduced heap models together with their synchronous product and their modelling by fixed-point equations in the dioid of formal power series. Section 4 is devoted to the study of properties of synchronous product of heap models that will be applied in Section 5 to supervisory control of heap models using residuation theory. Section 6 is devoted to an example illustrating the proposed approach. The conclusion is given and future extensions of our approach are discussed in Section 7.

2. Preliminaries on dioid algebras.

In this section we recall some fundamental algebraic notions needed in the next sections. An idempotent semigroup is a set M equipped with a commutative, associative operation \oplus that has a unit element ε that satisfies the idempotency condition $a \oplus a = a$ for each $a \in M$. There is a naturally defined partial order \preceq on any idempotent semigroup, namely, $a \preceq b$ if and only if $a \oplus b = b$.

An idempotent semigroup is called idempotent semiring or dioid if it is equipped with another associative operation \otimes that has a unit element e , distributes over \oplus on the left and on the right.

An idempotent semiring is said to be commutative if the multiplication is commutative. A dioid D is called to be complete if any nonempty subset has an infimum and a supremum with respect to the order generated by \oplus and the distributivity axioms extend to infinite sums. In the context of complete dioids, the star operation has been introduced by

$$a^* = \bigoplus_{i=0}^{\infty} a^i, \text{ where by convention } a^0 = e. \text{ Let us recall the notation } a^+ = \bigoplus_{i=1}^{\infty} a^i, \text{ i.e. } e \oplus a^+ = a^*$$

and the well know properties $aa^* = a^+$ and $a^* \oplus a^+ = a^*$ Moreover, the infimum of each non empty subset A of D always exists and it is denoted by $a = \bigwedge_{x \in A} x$.

Elementary examples of dioids are number dioids such as the semiring $R_{\max} = (R \cup \{-\infty\}, \max, +)$ endowed with maximum (denoted additively) and the usual addition (denoted multiplicatively). The zero element of R_{\max} is $\varepsilon = -\infty$, the unity is $e = 0$.

The complete version of R_{\max} is denoted by $\bar{R}_{\max} = (R \cup \{\pm\infty\}, \max, +)$.

2.1 Fixed point equations and residuation

Two powerful tools have been developed for complete dioids: fixed-point theorems and the residuation theory. Residuation theory generalizes the concept of inversion for mappings that do not necessarily admit an inversion, in particular those among ordered sets. If $f : C \rightarrow D$ is a mapping between two dioids, in most cases there does not exist a solution to the equation $f(x) = b$. Instead of solutions to this equation, the greatest solution to the inequality $f(x) \leq b$ or the least solution to the inequality $f(x) \geq b$ are considered. In the case these exist for all $b \in D$, the mapping f is called residuated, and dually residuated, respectively. The notation proposed in (Baccelli et al., 1992) is used in this paper.

Concretely, the residual mapping of a residuated mapping f will be denoted by $f^\#$. It is defined by the formula: $f^\#(y) = \max\{x : f(x) \leq y\}$. The residual of the right multiplication by an element $a \in D$ in a commutative dioid will be denoted through x/a , i.e. $x/a = \max_{y \in D} (y \otimes a \leq x)$. Similarly, the residual of the left multiplication is denoted by $a \setminus x = \max_{y \in D} (a \otimes y \leq x)$.

Finally, residuation of matrix multiplication will be needed. In this paper only residuated mappings of matrix multiplication are used. The residuated mapping of the left matrix multiplication, i.e. the greatest solution to the inequality $AX \leq B$ is denoted by $A \setminus B$. Similarly, the residuated mapping of the right matrix multiplication, i.e. the greatest solution to the inequality $XA \leq B$ is denoted by B/A . Recall from (Gaubert 1992) that for matrices $A \in D^{m \times n}$, $B \in D^{m \times p}$, and $C \in D^{n \times p}$ over a complete dioid D $A \setminus B \in D^{n \times p}$ is given

$$\text{by } (A \setminus B)_{ij} = \bigwedge_{l=1}^m A_{li} \setminus B_{lj} \text{ and } B/C \in D^{m \times n} \text{ is given by } (B/C)_{ij} = \bigwedge_{k=1}^p B_{ik} / C_{jk}.$$

Now fixed point equations in complete dioids are recalled (see Baccelli et al. 1992).

Theorem 1. Let D be a complete dioid, and $x = a \otimes x \oplus b$ be a linear equation of the fixed-point type. The least solution to this equation exists and is given by $x = a^* \otimes b$.

The following Lemma will be needed.

Lemma 1. Let D be a complete dioid, $a, b \in D$. Then

$$(a \oplus b)^* = (a^*b)^* a^* = (b^*a)^* b^* = a^*(ba^*)^* = b^*(ab^*)^* . \tag{1}$$

2.2 Formal power series and their properties

Now we recall formal power series, which include both formal languages and dater functions from R to R_{max} and their properties. Formal power series in the noncommutative variables from and coefficients from R_{max} form a dioid called dioid of formal power series.

The standard notation A^* is used for the free monoid of finite sequences (words) from A . The empty word is denoted by 1 .

Formal power series form a dioid denoted $R_{max}(A)$, where addition and (Cauchy or convolution) multiplication are defined as follows.

For any $s, s' \in R_{max}(A)$

$$\begin{aligned} (s \oplus s')(w) &= s(w) \oplus s'(w) \\ (s \otimes s')(w) &= \bigoplus_{uv=w} s(u) \oplus s'(v) \end{aligned} \tag{2}$$

This dioid is isomorphic to the dioid of generalized dater functions from A^* to R_{max} similarly as the dioid $Z_{max}(\gamma)$ used to study Timed Event Graphs (TEG), is isomorphic to the dioid of daters Z to Z_{max} . The isomorphism associates to any

$$y : A^* \rightarrow R_{max} \text{ the formal power series } \bigoplus_{w \in A^*} y(w)w \in R_{max}(A) .$$

The zero and identity series are respectively denoted \mathcal{E} and e . It will always be clear from the context whether these elements are meant for a number dioid or a dioid of formal power series.

$$\text{Let us recall } \forall w \in A^*, \mathcal{E}(w) = -\infty \text{ and } e(w) = \begin{cases} 0 & w = 1 \\ -\infty & w \neq 1 \end{cases} .$$

We consider in the sequel the complete version of $R_{max}(A)$ with coefficients in $\overline{R_{max}}$.

An order relation on $R_{max}(A)$ will be needed for introduction and study of control problems in Section 5. Let us recall the natural order relation on formal power series from $R_{max}(A)$. For $s, s' \in R_{max}(A)$ we put $s \leq s'$ iff $\forall w \in A^* : s(w) \leq s'(w)$, where in the latter inequality usual order on R_{max} that coincides with the natural order is used.

3. Heap models and their synchronous products

In this section heap models together with their basic properties are first recalled. Then synchronous product of heap models is proposed as a mechanism to build large heap models out of smaller ones.

3.1 Basic properties of heap models

Let us first recall the definition of time extension of heap models, also called task-resource systems (Gaubert & Mairesse, 1999), which model an important class of TDES exhibiting both synchronization and resource sharing phenomena.

Definition 1. (Heap model)

A heap model is the structure $\mathfrak{R} = (A, P, r, l, u)$, where

- A is a finite set of pieces (also called tasks)
- P is a finite set of slots (also called resources)
- $r : A \rightarrow \text{Pwr}(P)$ determines the subset of resources required by a task. It is always assumed that $\forall a \in A : r(a) \neq \emptyset$.
- $l : A \times P \rightarrow \mathbb{R}_{\max}$ is a function such that $l(a, p)$ determines the height of the lower contour of piece a at the slot p .
- $u : A \times P \rightarrow \mathbb{R}_{\max}$ is function such that $u(a, p)$ determines the height of the upper contour of piece a at the slot p .

By convention, $u \leq l$, $l(a, p) = u(a, p) = -\infty$ if $p \notin r(a)$, and $\min_{p \in r(a)} l(a, p) = 0$.

Any sequence of pieces (tasks) is called a heap, i.e. heaps are just words of the free monoid A^* . There is a nice geometrical interpretation of heaps.

The dynamics of heap models is described by row vectors of generalized dater function $x(w)_p; p \in P$ corresponding to the height of the heap $w = a_1 \dots a_n$ on individual slots $p \in P$.

It has been shown in (Gaubert & Mairesse, 1999) that the upper contour of a heap w , denoted by $x(w)$, and the overall height of the heap w , denoted $y(w)$, are given by the following letter driven (max,+)-linear equations in terms of generalized dater functions x and y .

$$\begin{aligned} x(1) &= (0 \dots 0) \\ x(wa) &= x(w)\mu(a) \\ y(w) &= x(w)(0 \dots 0)^T \end{aligned}$$

with the so called morphism matrix associated to the heap model and given by:

$$[\mu(a)]_{pq} = \begin{cases} 0 & \text{if } p = q \notin r(a) \\ u(a, q) - l(a, s) & \text{if } p \in r(a) \text{ and } q \in r(a) \\ -\infty & \text{otherwise} \end{cases} \quad (3)$$

It is called the morphism matrix associated to the heap model $\mathfrak{R} = (A, P, r, l, u)$.

In fact; it is shown in (Gaubert & Mairesse, 1999) that heap models are special (max,+)-automata with input and output functions as row, resp. column vectors of zeros, i.e. (max,+) identity elements, and the morphism matrix defined above. The (max,+)-automaton $A(\mathfrak{R}) = (\alpha, \mu, \beta)$ given by the triple input function $\alpha = (0 \dots 0)$, output function $\beta = (0 \dots 0)^T$, and the morphism matrix μ recalled above is then called heap automaton. Therefore one may view heap models as special (max,+)-automata called heap automata. On one hand the upper contour (height) of heap models is recognized by (max,+) automata and are considered as a subclass of (max,+) automata, but on the other hand it is known from (Gaubert & Mairesse, 1999) that heap models have a strong expressive power in terms of Timed Petri Nets (TPN), a very important tool in the study of Timed DES. Let us recall at this point few basic facts about TPNs.

Definition 2. (Timed Petri Net)

A Timed Petri Net (TPN) is a valued bipartite graph represented by a 5-tuple $N = (P, T, W, M, t_p)$. The finite sets P and T are called the set of places and the set of transitions, respectively. $W \in N^{|P| \times |T|}$ is the Boolean incidence matrix, $W_{ij} = W_{ij}^+ - W_{ij}^-$, where W_{ij}^+ equals 1 iff there exists an edge going from the transition T_j to the place P_i and similarly W_{ij}^- equals 1 iff there exists an edge going from P_i to T_j . Otherwise the corresponding values of W_{ij}^+ and W_{ij}^- are 0. The vector $M \in N^P$ denotes the distribution of the initial marking in the places. The vector $t_p \in N^P$ is formed by the sojourn times associated to the places of the net.

A place may contain tokens (called marking), which move from place to place according to the following (earliest functioning) firing rule. A transition T_j fires as soon as all the places P_i upstream T_j contain at least one available token. A token entering a place P_i is considered available after having sojourned t_{P_i} units of time. A Timed Event Graph (TEG) is a TPN such that each place has exactly one upstream and one downstream transition.

Although heap models similarly as Petri nets already support the concurrency phenomenon, it is useful for control purposes to introduce explicit concurrency by defining synchronous product of heap models. DES are typically composed of large number of components and the compositionality, i.e. building of large system out of smaller ones and inversely analysis or control synthesis based on decomposition into elementary components are very efficient techniques.

So we will consider two levels of concurrency: "local" (or implicit) concurrency that is inside each heap model given by "local" tasks that do not share any resource and the "global" (or explicit) concurrency between tasks of two component heap models that are not common to both heaps, but belong only to one component. This is given by the distribution of tasks in the definition of synchronous composition below. We then speak about private and shared tasks of two heap models.

The situation is similar to logical automata, where there can be two levels of concurrency (an explicit one using synchronous composition and an implicit one inside individual automata that may have "hidden concurrency").

We assume in the definition of synchronous product below that there are no shared resources between two heap models. Otherwise stated: resources are shared only by tasks within individual heap models. This requirement is best understood if one considers safe timed Petri nets (which can be viewed (Gaubert & Mairesse, 1999) as particular heap models), where synchronous compositions of subnets is realized by synchronizing shared transitions (in heap models tasks), while the set of places (in heap models resources) of the individual subnets are disjoint.

Definition 3. (Synchronous product of heap models)

Let $\mathfrak{R}_i = (A_i, P_i, r_i, l_i, u_i), i = 1, 2$ be two heap models with $P_1 \cap P_2 = \emptyset$. Their *synchronous product* is the heap model $\mathfrak{R}_1 \parallel \mathfrak{R}_2 = (A_1 \cup A_2, P_1 \cup P_2, r, l, u)$, where

$$r(a) = \begin{cases} r_1(a) \cup r_2(a) & \text{if } a \in A_1 \cap A_2 \\ r_1(a) & \text{if } a \in A_1 \setminus A_2 \\ r_2(a) & \text{if } a \in A_2 \setminus A_1 \end{cases},$$

$$l(a, p) = \begin{cases} l_1(a, p) & \text{if } p \in P_1 \\ l_2(a, p) & \text{if } p \in P_2 \end{cases}, \text{ and } u(a, p) = \begin{cases} u_1(a, p) & \text{if } p \in P_1 \\ u_2(a, p) & \text{if } p \in P_2 \end{cases}.$$

Since the slots (resources) of component heaps are disjoint, l and u are well defined: even though in $A_1 \cap A_2 = \emptyset$ for any $p \in P$ there is only one $i \in \{1, 2\}$, namely i such that $p \in P_i$, with $l_i(a, p)$ and $u_i(a, p)$ being defined.

Similarly as in the supervisory control of (logical) automata the purpose of synchronous product is twofold. Firstly, explicitly concurrent heap models (*cf.* concurrent or modular automata) are heap models built by the synchronous product of "local" heap models, whence the interest in studying the properties of synchronous composition of heap models. Secondly, synchronous product is used to describe the action of the supervisor, *i.e.* interaction of the supervisor with the system *cf.* (Kumar & Heymann, 2000).

Let us remark that if the above definition is used for control purposes, it is symmetric with respect to both the plant heap (say \mathfrak{R}_1) and the controller (say \mathfrak{R}_2). We then implicitly assume in the above definition that the supervisor is complete, *i.e.* that it never tries to disable an uncontrollable task. This is always true in the special case, where all tasks are controllable.

Now (max,+)-linear representation of heap models will be used for the study of the morphism matrix of the synchronous product of two heap models. An approach for just in time control of flexible manufacturing systems based on Petri net and heap models, that builds upon the approach of (Menguy, 1997), has been developed in (Al Saba et al., 2006).

Our aim is to develop the control theory directly for heap models using synchronous composition of a heap model with its controller (another heap model).

Let us consider the following flexible manufacturing system modelled by Petri net displayed below in Figure 1.

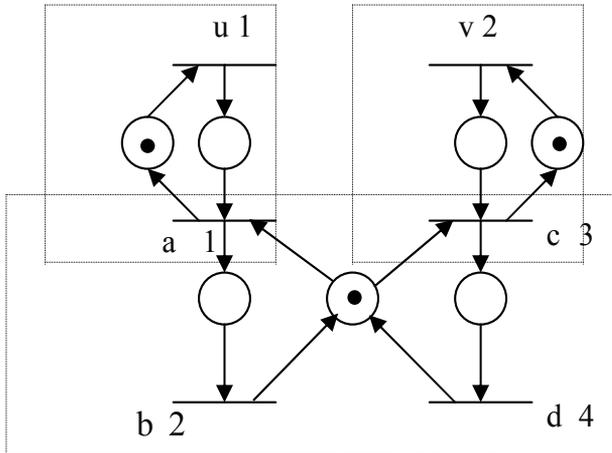


Fig. 1. Example of a Timed Petri Net composed of three modules

Note that this 1-safe Petri net is T-timed (timing is associated to transitions) and it can be decomposed into three parts, which are synchronized using shared (synchronization) transitions a and c. Equivalently, each component of this Petri net can be viewed as a separate heap model. The "global" heap model corresponding to the whole timed Petri net is the synchronous product of "local" heap models.

Similarly as a TEG admits a linear representation in the dioid of formal power series $Z_{max}(\gamma)$, see (Baceli et al., 1992), a heap model admits linear representation in the dioid of formal power series with noncommutative variables from A that we denote by $R_{max}(A)$. As an example let us consider the following heap automaton \mathfrak{R} corresponding to the TPN depicted on figure 1.

The set of tasks is $A = \{a, b, c, d\}$. The set of resources is $P = \{p_1, p_2, p_3\}$. Let $r(a) = r(b) = \{p_1, p_2\}$ and $r(c) = r(d) = \{p_1, p_3\}$. The lower and upper contours of tasks are given by

$$\begin{aligned}
 l(a, \cdot) &= (0 \ 0 \ -\infty); l(b, \cdot) = (0 \ 0 \ -\infty) \\
 l(c, \cdot) &= (0 \ -\infty \ 0); l(d, \cdot) = (0 \ -\infty \ 0) \\
 u(a, \cdot) &= (0 \ 1 \ -\infty); l(b, \cdot) = (2 \ 0 \ -\infty) \\
 u(c, \cdot) &= (0 \ -\infty \ 3); u(d, \cdot) = (4 \ -\infty \ 0)
 \end{aligned}$$

It has been shown in (Gaubert & Mairesse, 1999) that any heap model is a special (max,+)-automaton with the morphism matrix defined in equation (3). The graphical interpretation of the morphism matrix is given in terms of transition weights:

$[\mu(a)]_{ij} = k$ means that there is a transition labelled by $a \in A$ from state i to state j with weight k provided $k \neq -\infty$, while in the case $k = -\infty$ there is no transition from state i to state j . Note that the morphism matrix μ of a heap model can be also considered as element of $R_{max}(A)^{|R| \times |R|}$, and by extending the definition of μ from letters $a \in A$ to sequences (words) $w \in A^*$ using the morphism property $\mu(a_1 \dots a_n) = \mu(a_1) \dots \mu(a_n)$ and hence we can even write $\mu = \bigoplus_{w \in A^*} \mu(w)w$.

However, μ has an important property of being finitely generated, because it is completely determined by its values on A : $\mu(a), a \in A$. For this reason we have in fact

$\mu^* = \left(\bigoplus_{a \in A} \mu(a)a \right)^*$. Since we are interested in behaviors of heap models that are given in terms of μ^* (more precisely by $\alpha\mu^*\beta$) we abuse the notation and write simply $\mu = \bigoplus_{a \in A} \mu(a)a$.

The corresponding heap automaton is in Figure 2 below.

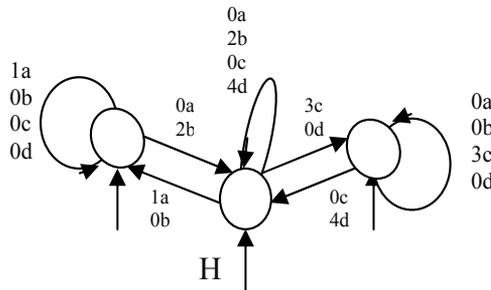


Fig. 2. Heap automaton corresponding to the Timed Petri Net

The state vector is associated to resources of \mathfrak{R} . We denote the component variables (formal power series) $x_2, x_1, x_3 \in R_{max}(A)$ from left to right. We obtain the following equations in dioid $R_{max}(A)$:

$$\begin{aligned}
 x_1 &= x_1(0a \oplus 2b \oplus 0c \oplus 4d) \oplus x_2(0a \oplus 2b) \oplus x_3(0c \oplus 4d) \oplus e \\
 x_2 &= x_1(1a \oplus 0b) \oplus x_2(1a \oplus 0b \oplus 0c \oplus 0d) \oplus e \\
 x_3 &= x_1(3c \oplus 0d) \oplus x_3(0a \oplus 0b \oplus 3c \oplus 0d) \oplus e \\
 y &= x_1 \oplus x_2 \oplus x_3
 \end{aligned}
 \tag{4}$$

The corresponding matrix form of system of equations (4) is

$$\begin{aligned} x &= x\mu \oplus \alpha \\ y &= x\beta \end{aligned} \quad \text{with } x = (x_1 \ x_2 \ x_3), \quad \alpha = (0 \ 0 \ 0), \quad \beta = (0 \ 0 \ 0)^T, \quad \text{and finally}$$

$$\mu = \begin{pmatrix} 0a \oplus 2b \oplus 0c \oplus 4d & 1a \oplus 0b & 3c \oplus 0d \\ 0a \oplus 2b & 1a \oplus 0b \oplus 0c \oplus 0d & \varepsilon \\ 0c \oplus 4d & \varepsilon & 0a \oplus 0b \oplus 3c \oplus 4d \end{pmatrix}. \quad (5)$$

It is now easy to see that in general we always have the following linear description of $(\max, +)$ automata in the dioid $R_{\max}(A)$ of formal power series:

$$\begin{aligned} x &= x\mu \oplus \alpha \\ y &= x\beta \end{aligned} \quad \text{with } \mu = \bigoplus_{a \in A} \mu(a)a \quad \text{the so called morphism matrix.}$$

Let us recall from Theorem 1 that the least solution to this equation is $y = \alpha \mu^* \beta$.

4. Linear representation of synchronous product

Since we introduce supervisory control of heap models using the synchronous product of the plant heap model with the controller heap model, it is important to study properties of the synchronous product. In this section the behavior of a synchronous product of two heap models will be represented in terms of morphism matrix of the synchronous product. According to the definition of synchronous product, the dimension (here number of resources) of the synchronous product of subsystems is the sum of dimensions of each subsystem. It should be intuitively clear that morphism matrices of synchronous products are block matrices, where the blocks are formed according to dimensions (number of resources) of the component heap models.

In order to simplify the approach we require that $l(a, p) = 0$ whenever $p \in r(a)$. This is a reasonable assumption that simply means that pieces are on the ground in all slots. Thus, the upper contour gives information about the duration of tasks for different resources used. We recall at this point that $l(a, p) = -\infty$ whenever $p \notin r(a)$.

Let \mathfrak{R}_1 and \mathfrak{R}_2 be two heap models with morphism matrices denoted by μ_1 and μ_2 , respectively. Their dimensions, i.e. the number of resources of \mathfrak{R}_1 and \mathfrak{R}_2 are m_1 and m_2 , respectively. The Booleans morphism matrices of underlying Boolean automata are denoted by B_1 and B_2 , respectively. Let us recall from (Komenda et al, 2007) that

$$[B_1(a)]_{ij} = \begin{cases} e & \text{if } \mu_1(a)_{ij} \neq \varepsilon \\ \varepsilon & \text{if } \mu_1(a)_{ij} = \varepsilon \end{cases}$$

Let $\varepsilon_{ij}, i, j = 1, 2$ be the rectangular matrices of $(\max, +)$ zeros of dimensions $m_i \times m_j$ and $E_i, i = 1, 2$ the square $(\max, +)$ -identity matrices of dimensions $m_i \times m_i$.

It will be shown that the morphism matrix of the parallel composition admits an interesting decomposition into blocks.

Indeed, the following block form of $\mu_{\mathfrak{R}}(a)$ depending on $a \in A = A_1 \cup A_2$ is now claimed.

Theorem 2. The morphism matrix of $\mathfrak{R} = \mathfrak{R}_1 \mid \mathfrak{R}_2$ admits the following decomposition:

- if $a \in A_1 \cap A_2$ then $\mu_{\mathfrak{R}}(a) = \begin{pmatrix} \mu_1(a) & \bar{\mu}_2(a) \\ \bar{\mu}_1(a) & \mu_2(a) \end{pmatrix}$, where

$$\bar{\mu}_1(a)_{ij} = \begin{cases} \mu_1(a)_{kj}, k \in r_1(a) \text{ arbitrary} & \text{if } i \in r_2(a) \\ -\infty & \text{if } i \notin r_2(a) \end{cases} \quad (6)$$

$$\text{and similarly, } \bar{\mu}_2(a)_{ij} = \begin{cases} \mu_2(a)_{lj}, l \in r_2(a) \text{ arbitrary} & \text{if } i \in r_1(a) \\ -\infty & \text{if } i \notin r_1(a) \end{cases}$$

- if $a \in A_1 \setminus A_2$ then $\mu_{\mathfrak{R}}(a) = \begin{pmatrix} \mu_1(a) & \varepsilon_{12} \\ \varepsilon_{21} & E_2 \end{pmatrix} \quad (7)$

- if $a \in A_2 \setminus A_1$ then $\mu_{\mathfrak{R}}(a) = \begin{pmatrix} E_1 & \varepsilon_{12}(a) \\ \varepsilon_{21}(a) & \mu_2(a) \end{pmatrix} \quad (8)$

Proof. This result can be obtained from the form of the morphism matrix in equation (3). In accordance with the definition of morphism matrix of heap models for $a \in A = A_1 \cup A_2$ three cases are to be distinguished. Let us start with the most complicated case, where $a \in A_1 \cap A_2$ is a shared task. It should be clear that for resources p, q from $P = P_1 \cup P_2$ there are four possibilities depending on whether p and q belong to P_1 or P_2 , whence the block form of $\mu_{\mathfrak{R}}(a)$. At this point we recall our assumption that $P_1 \cap P_2 = \emptyset$. It is then straightforward to see that in the diagonal blocks the individual morphism matrices appear. Also, the remaining non diagonal terms according to the definition of the morphism matrix of heap automata are equal to:

$$\bar{\mu}_1(a)_{ij} = \begin{cases} u(a, j) - l(a, i) = u_1(a, j) - l_2(a, i) = u_1(a, j) & \text{if } i \in r_2(a) \text{ and } j \in r_1(a) \\ -\infty & \text{if } i \notin r_2(a) \text{ or } j \notin r_1(a) \end{cases} \quad (9)$$

Note that any such element above in the lower left block must satisfy $i \in P_2$ and $j \in P_1$, hence $u(a, j) = u_1(a, j)$, $l(a, i) = l_2(a, i)$, and $l_2(a, i) = 0$ for $i \in r_2(a)$ according to our assumption and similarly for elements in the right upper block of $\mu_{\mathfrak{R}}(a)$ we get:

$$\bar{\mu}_2(a)_{ij} = \begin{cases} u(a,j) - l(a,i) = u_2(a,j) - l_1(a,i) = u_2(a,j) & \text{if } i \in r_1(a) \text{ and } j \in r_2(a) \\ -\infty & \text{if } i \notin r_1(a) \text{ or } j \notin r_2(a) \end{cases} \quad (10)$$

Since $\bar{\mu}_1(a)_{ij}$ equals either $u_1(a,j)$ (for $i \in r_2(a)$ and $j \in r_1(a)$) or to $(\max,+)$ zero otherwise, we can see that the rows corresponding $i \in r_2(a)$ of $\bar{\mu}_1(a)$, i.e. $\bar{\mu}_1(a)_{(i, \cdot)}$ are the same as the rows $\bar{\mu}_1(a)_{(k, \cdot)}$ for $k \in r_2(a)$, which are all the same.

Thus, the corresponding entry does not depend on j anymore. Similar arguments can show that $\bar{\mu}_2(a)$ is of the form above. The morphism matrix $\mu_{\mathfrak{R}}(a)$ of the composed heap has then the claimed form for $a \in A_1 \cap A_2$. In much easier situation when $a \in A_1 \setminus A_2$ it is sufficient to notice that no resource from P_2 is used by a . It is easily seen that $\mu_{\mathfrak{R}}(a)$ has again the claimed form. Finally, the case $a \in A_2 \setminus A_1$ is symmetric to the previous one.

Theorem on decomposition can be generalized to the case of n component heaps, where morphism matrix of synchronous product are matrices with $n \times n$ blocks. This is useful for decentralized control, but in this paper we only need synchronous product of the system with its controller, i.e. the case $n = 2$.

The algebraization of synchronous product presented in this section will be useful for control purposes in the next section.

5. Application to supervisory control

In this section supervisory control of heap models is studied. The aim is to satisfy a behavioral specification given by a formal power series. The closed-loop system is represented by parallel composition (synchronous product) of the plant with a supervisor to be found, which is itself represented by a heap model.

In general a supervisor acts on both timing and logical properties of the plant's behavior under supervision. In this section we focus mainly on timing aspects, when the supervisor does not disable, but only delay the execution of tasks corresponding to transitions in underlying timed Petri nets. This is similar to control of TEG in the maxplus algebra, where input transitions are added in order to delay the timed behavior of a TEG (Cottenceau et al, 2000).

Note that adding input transitions to a TEG corresponds to synchronization of the TEG with a TEG of the controller that has predefined structure given by the graph topology of input transitions. This way control of TEG can be seen in this sense as a special case of supervisory control, where the supervisor has a predefined fixed structure (logical behavior) and only output values (timing) of transitions are to be determined such that a specified behavior is met. Note however that such a methodology enables only to consider periodic inputs without a transient regime, which would require to either consider a controller,

where weights are time depending or equivalently to use a unfolding (max,+) automaton representation of the controller, which would have much more states.

Now we apply Theorem 2 to the control of heap models. The synchronous product of heap models $G = (A_g, P_g, r_g, l_g, u_g)$ and $C = (A_c, P_c, r_c, l_c, u_c)$ of dimensions m and n (respectively) corresponds to the controlled (closed-loop) system. The event alphabets of G and C are denoted by A_g and A_c , respectively. The event alphabet of the controlled system is denoted by A . According to definition of synchronous product, we then have $A = A_g \cup A_c$. Let us denote the morphism matrices of G and C by μ_g and μ_c , respectively. Now let us return to the description of behaviors of heap models in the dioid of formal power series $R_{max}(A)$. The vector of formal power series from $R_{max}(A)$ associated to generalized dater functions $x_{G|C} : A^* \rightarrow R_{max}^{m+n}$ satisfies the following equations:

$$\begin{aligned} x_{G|C} &= x_{G|C} \otimes \mu_{G|C} \oplus \alpha \\ y_{G|C} &= x_{G|C} \otimes \beta \end{aligned} \quad (11)$$

where $\mu_{G|C}$ the morphism matrix of $G|C$, α , and β are row, resp. column, vectors of zeros of dimension $m+n$, i.e. $\alpha = (e\dots e)$, and $\beta = (e\dots e)^T$. First of all, according to Theorem 1 the greatest solutions to equations above are

$$\begin{aligned} x_{G|C} &= \alpha \mu_{G|C}^* \\ y_{G|C} &= \alpha \mu_{G|C}^* \beta \end{aligned} \quad (12)$$

whence an interest in studying properties of $\mu_{G|C}^*$.

Given a specification behavior (e.g. language or formal power series), the goal in supervisory control of DES is to find a supervisor that achieves this specification as the behavior of the controlled system. In a first approach we assume, similarly as in control of TEG, that the structure of the controller is given, which means here that the controller heap model only delays executions of different tasks in the plant. This is done by the choice of upper contour functions (i.e. duration of controller's tasks) from $\mu_c(u), u \in A_c$. The delaying effect of the controller is naturally realized via its tasks (transitions of the corresponding heap automaton) shared with the plant heap.

The morphism matrix of the composed system is given by

$$\mu_{G|C} = \bigoplus_{a \in A} \mu_{G|C}(a) a = \bigoplus_{u \in A_c \setminus A_g} \mu_{G|C}(u) u \oplus \bigoplus_{a \in A_c \cap A_g} \mu_{G|C}(a) a \oplus \bigoplus_{u \in A_g \setminus A_c} \mu_{G|C}(a) a. \quad (13)$$

In order to simplify the approach our attention is from now on limited to the case $A_c = A_g$. This is a standard assumption in the supervisory control with complete observations. Since state vector in equation (12) is associated to resources, it can be written as $x_{G||C} = (x \ u)$, where the first component corresponds to the (uncontrolled) plant (heap G) and the second to the controller heap C .

Owing to Theorem 2 we have:

$$(x \ u) = (x \ u) \left[\bigoplus_{a \in A} \begin{pmatrix} H(a)a & \bar{F}(a)a \\ \bar{H}(a)a & F(a)a \end{pmatrix} \right] \oplus (\alpha_1 \ \alpha_2), \quad (14)$$

where α_1 and α_2 are vectors of zeros of corresponding dimensions, for any $a \in A$ $\mu_g(a)$ is for convenience denoted by $H(a)$, i.e. $\bar{H}(a)$ denotes $\bar{\mu}_g(a)$. Similarly, $F(a)$ is the notation for $\mu_c(a)$, i.e. $\bar{F}(a)$ denotes $\bar{\mu}_c(a)$.

This way we obtain

$$(x \ u) = (x \ u) \begin{pmatrix} H & \bar{F} \\ \bar{H} & F \end{pmatrix} \oplus (\alpha_1 \ \alpha_2),$$

where $H = \bigoplus_{a \in A} H(a)a$ and similarly $F = \bigoplus_{a \in A} F(a)a$, $\bar{H} = \bigoplus_{a \in A} \bar{H}(a)a$, and $\bar{F} = \bigoplus_{a \in A} \bar{F}(a)a$.

Hence,

$$\begin{aligned} x &= xH \oplus u\bar{H} \oplus \alpha_1 \\ u &= x\bar{F} \oplus uF \oplus \alpha_2 \end{aligned}$$

According to Theorem 1 the least solution of the second equation is $u = (x\bar{F} \oplus \alpha_2)F^*$. The above equation gives us the expression of the closed-loop system transfer, where the matrix F plays the role of a feedback. It is worth to notice that there is a significant difference from standard feedback control: unlike classical control theory the control variables have their inner dynamics. This is caused by adopting a supervisory control approach, where a controller is itself a dynamical system of the same kind as the uncontrolled system: a heap automaton. Therefore, our \bar{F} , which plays the role of feedback mapping, is determined by inner dynamics of the controller given by its morphism matrix F . The substitution into the first equation then yields

$$x = xH \oplus \left[(x\bar{F} \oplus \alpha_2)F^* \right] \bar{H} \oplus \alpha_1$$

Another application of Theorem 1 leads to the least solution given by

$$x = (\alpha_2 F^* \bar{H} \oplus \alpha_1) (H \oplus \bar{F} F^* \bar{H})^*.$$

From a different viewpoint, there is a strong analogy with the feedback approach for control of TEG, see (Cottenceau et al, 2000), which should not be surprising, because supervisory control (realized here by synchronous product) is based on a feedback control architecture. In supervisory control the control specification (as counterpart of reference output from control of TEG using dioid algebras) are given in terms of behaviors of $(\max,+)$ automata (*i.e.* formal power series). In fact, for a reference output y_{ref} we are interested in the greatest feedback F such that the output of the closed-loop system y is less or equal to the specification $y_{ref} : y \leq y_{ref}$. We obtain in our situation:

$$\left(\alpha_2 F^* \bar{H} \oplus \alpha_1\right) \left(H \oplus \bar{F} F^* \bar{H}\right)^* \beta \leq y_{ref} . \quad (15)$$

And $\left(H \oplus \bar{F} F^* \bar{H}\right)^* \leq \left(\alpha_2 F^* \bar{H} \oplus \alpha_1\right) \setminus y_{ref} / \beta$, where $\left(\alpha_2 F^* \bar{H} \oplus \alpha_1\right) \setminus y_{ref} / \beta$ is a matrix playing the role of reference model in (Cottenceau et al, 2000). An application of Lemma 1 leads to:

$$\left(H \oplus \bar{F} F^* \bar{H}\right)^* = H^* \left(\bar{F} F^* \bar{H} H^*\right)^* \leq \left(\alpha_2 F^* \bar{H} \oplus \alpha_1\right) \setminus y_{ref} / \beta . \quad (16)$$

Note that all operations involved in the description above, *i.e.* \oplus, \otimes , and the Kleene star, are lower semicontinuous and residuated when the image is suitably constrained. Hence, using residuation theory (see Section 2) it should be possible to obtain the greatest series F corresponding to the "controller part" of the morphism matrix.

Moreover, as follows from Theorem 2 that \bar{F} can simply be expressed using F . Hence, there is a hope that at least in some special cases residuation theory (see Section 2) can be applied to obtain the greatest series F corresponding to the "controller part" of the morphism matrix $\mu_{G|C}$ such $y_{G|C}$ satisfies a given specification (*e.g.* it is less than or equal to a given reference output series y_{ref}).

We aim at obtaining the greatest $\bar{F} F^*$ such that inequality (16) is satisfied. This is far from being an easy problem. The situation is much simpler in case $\bar{F} = F$ and $\bar{H} = H$. This is satisfied if we assume that the controller heap model has the same number of resources as the uncontrolled heap model and the logical structure of the controller (given by $r_c : A \rightarrow Pwr(P_c)$) mimics the logical structure of the plant (given by $r_g : A \rightarrow Pwr(P_g)$). Formally stated it is required that there exists an isomorphism between P_c and P_g such that r_c and r_g are equal up to this isomorphism.

In terms of Petri nets this can be interpreted as having a controller net with the same net topology (*i.e.* logical structure as the uncontrolled net). This means that in the closed-loop system there are always parallel places of the controller corresponding to places of the uncontrolled net.

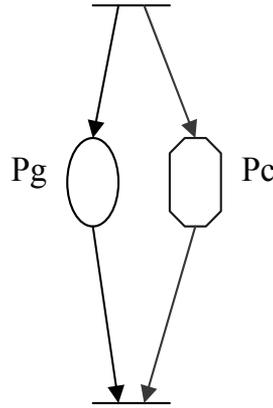


Fig. 3. The interpretation of controller as a Timed Petri Net.

The role of the controller is only to act on the system through holding times of the controller's places that correct the holding times of the places in the original net. Because of the fixed parallel structure of the controller it is clear that the controller can in this case only delay the firing of the transitions, which are all shared by the system and the controller.

It is easy to check that Theorem 2 in such a case gives $\bar{F} = F$ as well as $\bar{H} = H$ and $\alpha_1 = \alpha_2 = \alpha$, row vector of zeros of dimension $m=n$.

Hence, inequality (15) leads to

$$\alpha(F^*H \oplus E)(H \oplus FF^*H)^* \beta \leq y_{ref}, \text{ where } E \text{ is the identity matrix.}$$

An easy calculation yields $(H \oplus FF^*H)^* = ((E \oplus F^+)H)^* = (F^*H)^*$ and then $(F^*H \oplus E)(F^*H)^* = (F^*H)^+ \oplus (F^*H)^* = (F^*H)^*$. Thus, the control problem becomes :

$$y = \alpha(F^*H)^* \beta \leq y_{ref}. \tag{17}$$

i.e. the problem is to find the greatest F such that $(F^*H)^* \leq \alpha \setminus y_{ref} / \beta$.

Since the Kleene star is not a residuated mapping in general, such a problem has only a solution if $\alpha \setminus y_{ref} / \beta$, playing the role of reference model G_{ref} from see (Cottenceau et al, 2000), is of a special form to be found.

We notice that in our particular situation $E \leq H$, which follows from the form of morphism matrix of heap models (see Section 3).

The following Lemma is now useful.

Lemma 2. If $E \leq H$ then for any $B \in R_{max}(A)^{n \times n}$ any solution of $(X^*H)^* \leq B$ is a solution of $(X^*H)^* X^* \leq B$ and vice versa.

Proof. If X a solution of $(X^*H)^* \leq B$, then $(X^*H)^* = E \oplus (X^*H)^+ \leq B$, hence also $(X^*H)^+ \leq B$. Therefore,

$$(X^*H)^* X^* = (X^*H)^* X^* E \leq (X^*H)^* X^* H = (X^*H)^+ \leq B,$$

where the first inequality follows from isotony of multiplication and the assumption $E \leq H$. Conversely, if X is a solution of $(X^*H)^* X^* \leq B$, then

$$(X^*H)^* = (X^*H)^* E \leq (X^*H)^* X^* \leq B$$

as follows from isotony of multiplication and the assumption that $E \leq X^*$, a general and very well known property of the Kleene star.

Using Lemma 2 (with F playing the role of X) our problem is equivalently formulated as follows: find the greatest solution in F of $(F^*H)^* F^* \leq \alpha \setminus y_{ref} / \beta$. It follows from Lemma 1 that $(F^*H)^* F^* = (H \oplus F)^* = H^*(FH^*)^*$, thus we get formally the same problem as the one solved in (Cottenceau et al, 2000) with H^* playing the role of transfer function H in the TEG setting. The following result adapted from Proposition 3 in (Cottenceau et al, 2000) is useful: If there exists a matrix $D \in R_{max}(A)^{n \times n}$ such that $\alpha \setminus y_{ref} / \beta = H^* D^*$ or there exists $\tilde{D} \in R_{max}(A)^{n \times n}$ such that $\alpha \setminus y_{ref} / \beta = \tilde{D}^* H^*$ then there exists the greatest F such that $H^*(FH^*)^* \leq \alpha \setminus y_{ref} / \beta$, namely

$$F^{opt} = (\alpha H^*) \setminus y_{ref} / (H^* \beta). \quad (18)$$

Let us note that it is very difficult to compute the optimal feedback according to formula (18), although the formula is similar to the corresponding one used in feedback control of timed event graphs. In fact, dealing with formal power series in several non commutative variables from A is much more difficult than handling formal power series from $Z_{max}(\gamma)$. Similar simplification rules, the one proposed in (Benveniste et al, 1998a) and (Benveniste et al, 1998b), should be used in the computation according to formula (18). These simplification rules correspond to the fact that nondecreasing series are useful for practical computation.

In the special case we have restricted attention to, our methods yields the greatest feedback such that timing specification given by y_{ref} is satisfied, provided y_{ref} is of one of the above special forms. In our case of a controller with fixed logical structure only timed behavior is under control. At this point it is not clear yet how to leave the restriction on the form of y_{ref} . In timed event graphs this has been done by using the concept of compensator borrowed (extended) from the classical control theory. However there is no similar structure

for heap models, because there is no input function and we use another heap model as a controller.

If we are interested in manufacturing systems, where specifications are given in terms of Petri nets, the reference output is not typically required to be met for all sequences of tasks, but only those having a real interpretation. These are given by the corresponding (logical) Petri net language, say L . Thus, the problem is to find the greatest F , such that

$$\alpha H^*(FH^*)^* \beta \text{char}(L) \leq y_{ref} \text{char}(L),$$

where $\text{char}(L) = \bigoplus_{w \in L} e.w$ is the series with Boolean coefficients, *i.e.* the formal series of language L . Let us recall (Gaubert & Mairesse, 1997) that such a restriction is formally realized by the tensor product (residuable operation) of the heap automaton with the logical (marking) automaton recognizing the Petri net language L , which is compatible with Theorem 4.1 of (Komenda et al, 2007).

Note that specifications based on (multivariable) formal power series are not easy to obtain in many practical problems, in particular those coming from production systems, often represented by Petri nets. In fact, given a reference output series amounts to solve a scheduling problem. A formal power series specification is not given, but it is to be found: *e.g.* using Jackson rule (Jacson, 1955).

6. Example

The following simple example is given in order to illustrate our approach. We consider the following simple timed Petri nets with their underlying heap models described below.

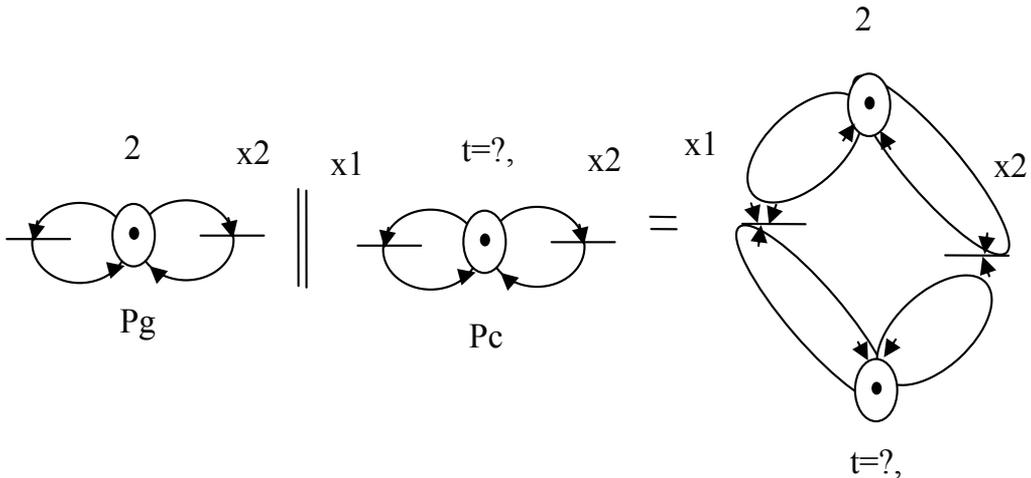


Fig. 4. Control of a simple heap model corresponding to TPNs above.

In the timed Petri nets above the timing (holdig time) of places Pg and Pc are 2 and t , respectively. In the controller net the value t is the control parameter.

The heap model G corresponding to the above simplest possible timed Petri nets with resource sharing is given below togetherwith its controller C .

$G = (A_g, P_g, r_g, l_g, u_g)$ and $C = (A_c, P_c, r_c, l_c, u_c)$ with $A = A_c = A_g = \{x_1, x_2\}$, $P_g = \{Pg\}$, $P_c = \{Pc\}$, $r_g(x_1) = r_g(x_2) = \{Pg\}$, $r_c(x_1) = r_c(x_2) = \{Pc\}$, $l_g(x_i, Pg) = \varepsilon$ for $i \in \{1, 2\}$, $l_c(x_i, Pc) = \varepsilon$ for $i \in \{1, 2\}$, $u_g(x_1, Pg) = 2 = u_g(x_2, Pg)$, and finally $u_c(x_1, Pc) = t = u_c(x_2, Pc)$.

$$x_G = x_G \otimes \mu_G \oplus \alpha$$

$$x_C = x_C \otimes \mu_C \oplus \alpha'$$

where

$\mu_G = 2x_1 \oplus 2x_2$ and $\mu_C = tx_1 \oplus tx_2$ are morphism matrices (here scalars of dimension 1).

In accordance with Theorem 2 we obtain

$$x_{G|C} = x_{G|C} \otimes \begin{bmatrix} \mu_G & \bar{\mu}_C = \mu_C \\ \bar{\mu}_G = \mu_G & \mu_C \end{bmatrix} \oplus \alpha$$

$$y_{G|C} = x_{G|C} \otimes \beta$$

with

$$\mu_{G|C} = \begin{bmatrix} 2(x_1 \oplus x_2) & t(x_1 \oplus x_2) \\ 2(x_1 \oplus x_2) & t(x_1 \oplus x_2) \end{bmatrix}.$$

Let us choose the control reference (specification) as $y_{ref} = (4x_1 \oplus 4x_2)^*$. This specification is of the form

$$\alpha \setminus y_{ref} / \beta = y_{ref} = (2x_1 \oplus 2x_2)^* (4x_1 \oplus 4x_2)^* = H^* D^*.$$

We compute

$$F^{opt} = (\alpha H^*) \setminus y_{ref} / (H^* \beta) = H^* \setminus y_{ref} / H^* = (4x_1 \oplus 4x_2)^*.$$

Indeed, by definition

$$H^* \setminus y_{ref} = \max_x \{ H^* x \leq y_{ref} \} = \max_x \{ (2x_1 \oplus 2x_2)^* x \leq (4x_1 \oplus 4x_2)^* \} = (4x_1 \oplus 4x_2)^*,$$

and similarly $(4x_1 \oplus 4x_2)^* / H^* = (4x_1 \oplus 4x_2)^*$, whence the expected result. In the above computation we use the simple fact that for two series (here polynomial) s and t with $s \leq t$, $s \geq e$, and $t \geq e$ we have $s^* t^* = t^*$.

Let us remark that if one would choose some different specification, e.g. $y_{ref} = (3x_1 \oplus 4x_2)^*$, then this specification is not compatible with the system, because there is only one place, where the timing is a control parameter. In order to achieve such a specification one would need a heap model corresponding to the following net structure.

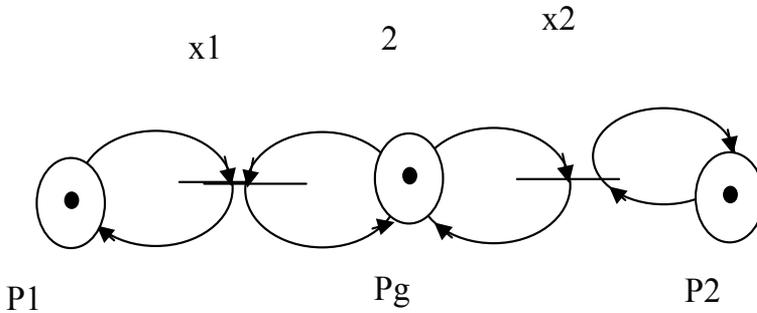


Fig. 5. Another PN corresponding to a heap model allowing for more general specifications.

7. Concluding remarks

It has been shown how methods of dioid algebras can be used in supervisory control of heap models. We have proposed a synchronous product of heap models. The structure of the morphism matrix of synchronous product of two heap models is derived and applied to control of heap models.

The present research is a preliminary step in control of heap automata. We have limited our attention to a particular type of synchronous composition. In this work we have assumed that all events were controllable, i.e. any event may be delayed and even disabled (prevented from happening) by a suitable controller heap automaton. This assumption is however often unrealistic in practice: for instance one can hardly imagine that different kinds of system failures can always be avoided. Another restriction is the one we have imposed on the form of the reference input y_{ref} (in supervisory control also called control specification). One possible way of leaving this restriction is to formulate heap automata in terms of input output automata and use the concept of precompensator from the classical control theory. Let us recall that recently different types of automata (e.g. classical automata, timed automata, stochastic automata) have been formulated in an equivalent way using explicit input and output functions as input output automata (e.g. IO timed automata and IO stochastic automata). It seems therefore interesting to work with IO (max,+) automata in order to extend the techniques based on precompensator from timed event graphs to our setting of heap automata.

Of potential interest is also supervisory control with partial controllability and partial observations or decentralized control of heap automata. Modular control of (explicitly)

concurrent heap automata that are formed as synchronous compositions of heap models is particularly worthy to investigate.

8. Acknowledgement

KJB100190609, of the French-Czech bilateral project Barrande N. 14235XG and of the Academy of Sciences of the Czech Republic, Institutional Research Plan No. AV0Z10190503 are gratefully acknowledged.

9. References

- Al Saba, M.; Boimond J.L. & Lahaye, S (2006). On just in time control of flexible manufacturing systems via dioid algebra. *Proceedings of INCOM'06*, Vol.2, pp. 137-142, Saint-Etienne, France.
- Baccelli, F.; Cohen, G.; Olsder, G.J. & Quadrat, J.P. (1992). *Synchronization and linearity. An algebra for discrete event systems*, John Wiley & Son, New York.
- Benveniste A. ; Jard C. & Gaubert S (1998a). Algebraic techniques for timed systems. *Proceedings of CONCUR'98, International Conference on Concurrency Theory*, 1998.
- Benveniste A. ; Jard C. & Gaubert S (1998b). Monotone rational series and max-plus algebraic models of real-time systems. *Proceedings of the 4th Workshop on Discrete Event Systems, WODES'98*, Cagliari, Italy, august 1998.
- Cottenceau, B.; Hardouin, L.; Boimond J.L. & Ferrier, J.L. (2001). Model Reference Control for Timed Event Graphs in Dioids. *Automatica*, Vol. 37, pp. 1451-1458.
- Gaubert, S. (1992). *Theorie des systèmes linéaires dans les dioïdes*. Thèse de doctorat, Ecole des Mines de Paris, 1992.
- Gaubert, S. (1995). Performance evaluation of (max,+) automata. *IEEE Transactions on Automatic Control*, Vol. 40, N12, pp. 2014-2025.
- Gaubert, S. & Mairesse, J. (1997). Task resource models and (max,+) automata, In J. Gunawardena, Editor: *Idempotency*. Cambridge University Press, 1997.
- Gaubert, S. & Mairesse, J. (1999). Modeling and analysis of timed Petri nets using heaps of pieces. *IEEE Transactions on Automatic Control*, Vol. 44, N4, pp. 683-698.
- Jackson, J.R. (1955). Scheduling a Production Line to Minimize Maximum Tardiness. *Research report 43. University of California Los Angeles*. Management Science Research Project.
- Komenda, J.; Al Saba, M. & Boimond, J.L. (2007). Supervisory Control of Maxplus Automata: Timing Aspects. In *Proceedings of the European Control Conference (ECC) 2007*, Kos (Greece).
- Kumar, R. & Heymann, M. (2000). Masked prioritized synchronization for interaction and control of discrete-event systems. *IEEE Transaction Automatic Control* 45, 1970-1982, 2000.
- Lin, F. & Wonham, W.M. (1998). On Observability of Discrete-Event Systems. *Information Sciences*, Vol. 44, pp. 173-198.
- Menguy, E. (1997). Contribution à la commande des systèmes linéaires dans les dioïdes. *Thèse de doctorat*, Université d'Angers.
- Ramadge, P.J. & Wonham, W.M. (1989). The Control of Discrete-Event Systems. *Proceedings of IEEE*, Vol. 77, pp. 81-98, 1989.

Sifakis, J. & Yovine, S. (1996). Compositional specification of timed systems. *Proceedings of the 13th Symposium on Theoretical Aspects of Computer Science, STACS'96*, pp. 347-359, . LNCS 1046.

Batch Deterministic and Stochastic Petri Nets and Transformation Analysis Methods

Labadi Karim¹, Amodeo Lionel², and Chen Haoxun²

¹*Ecole d'Electricité, de Production et de Méthodes Industrielles (Cergy-Pontoise)*

²*Université de Technologie de Troyes (Troyes)
France*

1. Introduction

Industrial systems such as production systems and distribution systems are often characterized as batch processes where materials are processed in batches and many operations are usually performed in batch modes to take advantages of the economies of scale or because of the batch nature of customer orders. The Batch Deterministic and Stochastic Petri Nets (BDSPN) is a class of Petri nets recently introduced for the modelling, analysis and performance evaluation of such systems which are discrete event dynamic systems with batch behaviours. The BDSPN model enhances the modelling and analysis power of the existing discrete Petri nets. It is able to describe essential characteristics of logistics systems (batch behaviours, batch operational policies, synchronization of various flows, randomness) and more generally discrete event dynamic systems with batch behaviours. The model is particularly adapted for the modelling of flow evolution in discrete quantities (variable batches of different sizes) and is capable of describing activities such as customer order processing, stock replenishment, production and delivery in a batch mode. The capability of the model to meet real needs is demonstrated through applications to modelling and performance optimization of inventory systems (Labadi et al., 2007,2005) and a real-life supply chain (Amodeo et al., 2007; Chen et al., 2005,2003).

Graph transformation is a fundamental concept for analysis of the systems described by graphs. The state of the art reporting for languages, tools and applications for graph transformation is given in the "Handbook of Graph Grammars and Computing by Graph Transformation" (Ehrig et al., 1999). In contrast to most applications of the graph transformation approach, where the states of a system are denoted by a graph, and transformation rules describe the state changes and the dynamic behaviour of systems, in the area of Petri nets (Murata, 1989) we apply transformation rules to modify a net in a stepwise way. This kind of transformation for Petri nets is considered to be a vertically structural technique, known as rule-based net transformation. This approach has been applied to various Petri net models such as basic Petri nets (Lee-Kwang et al., 1985, 1987), timed Petri nets (Juan et al., 2001; Wang et al. 2000), stochastic Petri nets (Ma & Zhou, 1992; Li-Yao et al. 1995), and coloured Petri nets (Haddad, 1988). There are different types of reduction/transformation techniques proposed in the literature. As we know, the reduction

is generally applied to resolve the state explosion problem of Petri nets. It aims at reducing the size of a Petri net model while retaining important properties of the model, such as liveness, safety, and boundeness. We can also find the work which transforms a Petri net model into another model such as UML (unified modelling language), diagrams, max + algebra model or vice versa. The objective of such a transformation is to use analysis methods of the resultant model to analyze the original model. Finally, one class of Petri nets may be transformed into another class in order to use theoretical results and analysis methods of the latter class to analyze the former class. A typical example of such a transformation is the transformation of a coloured Petri into an ordinary Petri net.

This chapter is organized into two parts. The first part is dedicated to a general description of the BDSPN model. A formal description of the model and its dynamic execution rules are presented with some illustrative examples. The capability of the model for modelling discrete event dynamic systems with batch behaviours is demonstrated through these examples. The second part of this chapter presents our recent work on structural and behavioural analysis of the BDSPNs by transforming them into other Petri nets. Two transformation analysis methods for the model are developed. The first method transforms a BDSPN into an equivalent classical discrete Petri net under some conditions. In this case, the corresponding transformation procedures are presented. For other cases, especially for BDSPNs with variable arc weights depending on their marking, the transformation is impossible. The second method analyzes a BDSPN based on its associated discrete Petri net which is obtained by converting the batch components (batch places, batch tokens, batch transitions) of the BDSPN into discrete components of the discrete Petri net. We show that although a BDSPN and its associated discrete Petri net behave differently, they have several common qualitative properties. This study establishes a relationship between BDSPNs and classical discrete Petri nets and demonstrates the necessity of the introduction of the BDSPN model.

2. Fundamentals of the BDSPN model

BDSPN extend Deterministic and Stochastic Petri Net (DSPN) (Lindemann, 1998; Ajmone Marsan et al. 1995) by introducing batch components, new transition enabling and firing rules, and specific policies defining the timing concept of the model. A BDSPN consists of places, transitions, and arcs that connect them. As shown in Fig. 1, a BDSPN has two types of places: discrete places and batch places. *Discrete places* can contain *discrete tokens* as in standard Petri nets. *Batch places* can contain *batch tokens* which are represented by Arabic numbers that indicate the sizes of the tokens. The current state of the modelled system (the marking) is given by the number of tokens in each discrete place and a list of positive integers in each batch place. Transitions are active components. They model activities which can occur (by firing transitions), thus changing the state of the system. Transitions are only allowed to fire if they are enabled, which means that all the preconditions for the activity must be fulfilled. When the transition fires, it removes tokens from its input places and adds some at all of its output places. The *enabling* and the *firing* of a transition depends on the cardinality of each arc, and on the current marking of each input places allowing the synchronization of discrete and batch token flows in the model. Since transitions are often used to model activities (production, delivery, order, etc.), transition enabling duration corresponds to activity execution and transition firing corresponds to activity completion.

Hence, a timing concept is naturally included into the formalism of the model. In a BDSPN, three types of transitions can be distinguished depending on their associated delay: *immediate transitions* (no delay), *exponential transitions* (delay is an exponential distribution), and *deterministic transitions* (delay is fixed). In this section, we recall the basic definition and the dynamical behaviour of the model.

2.1 Definition of the model

A BDSPN is a nine tuple $(P, T, I, O, V, W, \Pi, D, \mu_0)$ where :

- $P = \{p_1, p_2, \dots, p_m\} = P_d \cup P_b$ is a finite set of places consisting of the discrete places in set P_d and the batch places in set P_b . Discrete places and batch places are represented by single circles and squares with an embedded circle, respectively. Each token in a discrete place is represented by a dot, whereas each batch token in a batch place is represented by an Arabic number that indicates its size.
- $T = \{t_1, t_2, \dots, t_n\} = T_i \cup T_d \cup T_e$ is a set of transitions consisting of immediate transitions in set T_i , the deterministic timed transitions in set T_d , and exponentially distributed transitions in set T_e . T can also be partitioned into $T_d \cup T_b$: a set of discrete transitions T_d and a set of batch transitions T_b . For simplicity, here we abuse the notation T_d which is used for both the set of deterministic timed transitions and the set of discrete transitions in case of non confusion. A transition is said to be a batch transition (respectively a discrete transition) if it has at least an input batch place (respectively if it has no input batch place).
- $I \subseteq (P \times T)$, $O \subseteq (T \times P)$, and $V \subseteq (P \times T)$ define the input arcs, the output arcs and the inhibitor arcs of all transitions, respectively. It is assumed that only immediate transitions are associated with inhibitor arcs and that the inhibitor arcs and the input arcs are two disjoint sets.
- W is the arc weight function that defines the weights of the input, output, and inhibitor arcs. For any arc (i, j) , its weight $w(i, j)$ is a linear function of the M-marking with integer coefficients α, β , i.e., $w(i, j) = \alpha_{ij} + \sum_{p \in P} \beta_{(i, j)p} \times M(p)$. The weight $w(i, j)$ is assumed to take a positive value.
- $\Pi: T \rightarrow \mathbb{N}$ is a priority function assigning a priority to each transition. Timed transitions are assumed to have the lowest priority, i.e.; $\Pi(t) = 0$ if $t \in T_d \cup T_e$. For each immediate transition $t \in T_i$, $\Pi(t) \geq 1$.
- $D: T \rightarrow [0, \infty)$ defines the firing times of all transitions. It specifies the mean firing delay for each exponential transition, a constant firing delay for each deterministic transition, and a zero firing delay for each immediate transition
- μ_0 is the initial μ -marking, a row vector that specifies a multiset of batch tokens for each batch place and a number of discrete tokens for each discrete place.

The state of the net is represented by its μ -marking. We use two different ways to represent the μ -marking of a discrete place and the μ -marking of a batch place. The first marking is represented by a nonnegative integer as in standard Petri nets, whereas the second marking is represented by a multiset of nonnegative positive integers. The multiset may contain identical elements and each integer in the multiset represents a batch token with a given size. Moreover, for defining the net, another type of marking, called *M-marking*, is also introduced. For each discrete place, its M-marking is the same as its μ -marking, whereas for each batch place its M-marking is defined as the total size of the batch tokens in the place.

The state or μ -marking of the net is changed with two types of transition firing called "*batch*

firing” and “*discrete firing*”. Whether the firing of a transition is batch firing or discrete firing depends on whether the transition has batch input places. To introduce batch firing, we need some notations. A place connected with a transition by an arc is referred to as input, output, and inhibitor place, depending on the type of the arc. The set of input places, the set of output places and the set of inhibitor places of transition t are denoted by *t , t^\bullet , and \mathcal{A} , respectively, where ${}^*t = \{p \mid (p, t) \in I\}$, $t^\bullet = \{p \mid (t, p) \in O\}$, and $\mathcal{A} = \{p \mid (p, t) \in V\}$. The weights of the input arc from a place p to a transition t , of the output arc from t to p are denoted by $w(p, t)$, $w(t, p)$ respectively.

2.2 Batch enabling and firing rules

A batch transition t is said to be enabled at μ -marking μ if and only if there is a *batch firing index* (positive integer) $q \in \mathbb{N}$ ($q > 0$) such that:

$$\forall p \in {}^*t \cap P_b, \exists b \in \mu(p) : q = b / w(p, t) \quad (1)$$

$$\forall p \in {}^*t \cap P_d, M(p) \geq q \times w(p, t) \quad (2)$$

$$\forall p \in {}^*t, M(p) < w(p, t) \quad (3)$$

The batch firing of t leads to a new μ -marking μ' :

$$\forall p \in {}^*t \cap P_d, \mu'(p) = \mu(p) - q \times w(p, t) \quad (4)$$

$$\forall p \in {}^*t \cap P_b, \mu'(p) = \mu(p) - \{q \times w(p, t)\} \quad (5)$$

$$\forall p \in {}^*t \cap P_a, \mu'(p) = \mu(p) + q \times w(t, p) \quad (6)$$

$$\forall p \in {}^*t \cap P_b, \mu'(p) = \mu(p) + \{q \times w(t, p)\} \quad (7)$$

A batch transition t is said to be *enabled* if : (i) Each batch input place p of the transition has a batch token with size b such that all these batch tokens have the same batch firing index q defined as $b/w(p, t)$ for the transition, (ii) Each discrete input place of the transition has enough tokens to simultaneously fire the transition for a number of times given by the index, (iii) The number of tokens in each inhibitor place of the transition is less than the weight of the inhibitor arc connecting the place to the transition. For any batch output place, the firing of an enabled batch transition generates a batch token with the size given by the multiplication of the batch firing index and the weight of the arc connecting the transition to the batch place. For any discrete output place, the firing of the transition generates a number of discrete tokens with the number given by the multiplication of the discrete firing index and the weight of the arc connecting the transition to the discrete place.

2.3 Batch firing example

To well understand the mathematical and intuitive meaning of the batch transition firing of the BDSPN model, consider the net in Fig.1 describing an assembly-to-order system that requires two components. In the model, discrete places p_1 and p_2 are used to represent the stock of component A and the stock of component B respectively. Batch place p_3 is used to represent batch customer orders with different and variable sizes. To fill a customer order of size b , we need $b \times w(p_1, t_1) = 2b$ units of component A from the stock represented by p_1 and $b \times w(p_2, t_1) = b$ units of component B from the stock represented by p_2 . These components will be assembled to b units of final product to fill the order. For instance, at the current μ -marking $\mu_0 = (4, 3, \{4, 2, 3\}, \emptyset, 0)^T$, it is possible to fill the batch customer order $b = 2$ in batch place p_3 since the batch transition t_1 is enabled with $q = b / w(p_3, t_1) = 2$. After the batch firing of transition t_1 (start assembly), the corresponding batch token $b = 2$ will be removed from batch place p_3 , $q \times w(p_1, t_1) = 4$ discrete tokens will be removed from discrete place p_1 , and $q \times w(p_2, t_1) = 2$ discrete tokens will be removed from discrete place p_2 . A batch token with the size equal to $q \times w(t_1, p_4) = 2$ will be created in batch place p_4 and two discrete tokens will be created in discrete place p_5 . Therefore, the new μ -marking of the net after the batch firing is: $\mu_1 = (0, 1, \{4, 3\}, \{2\}, 2)^T$ and its corresponding M-marking is $M_1 = (0, 1, 7, 2, 2)^T$.

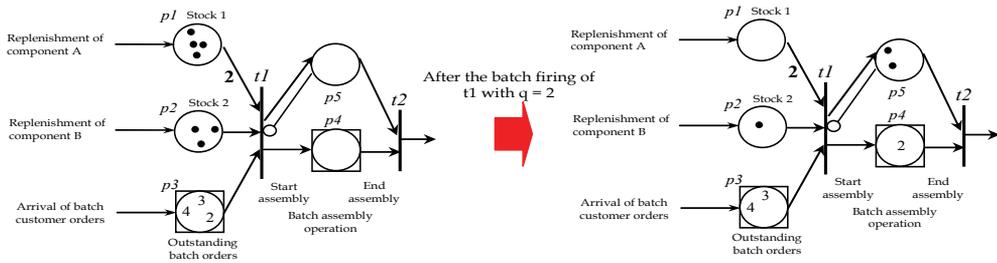


Fig. 1. An assembly-to-order system: synchronization and coordination of material and information flows (Batch firing example)

2.4 Discrete enabling and firing rules

A discrete transition t is said to be enabled at μ -marking μ (its corresponding M-marking M) if and only if:

$$\forall p \in {}^*t, M(p) \geq w(p, t) \quad (8)$$

$$\forall p \in {}^*t, M(p) < w(p, t) \quad (9)$$

The discrete firing of t leads to a new μ -marking μ' :

$$\forall p \in {}^*t, \mu'(p) = \mu(p) - w(p, t) \quad (10)$$

$$\forall p \in t^\bullet \cap P_d, \mu'(p) = \mu(p) + w(t, p) \quad (11)$$

$$\forall p \in t^* \cap P_b, \mu'(p) = \mu(p) + \{w(t, p)\} \tag{12}$$

The firing rules in this case are the same as those for a transition in a classical Petri net. For each output batch place p , after the firing of transition t , a batch token with the size equal to the weight $w(p, t)$ will be created. The discrete enabling and firing rules of a discrete transition t can be regarded as a special case of the batch enabling and firing rules of a batch transition. For $q = 1$ and ${}^*t \cap P_b = \emptyset$ the batch firing rules are reduced to the discrete firing rules. A close look of the enabling conditions (2) and (3) finds that they are actually the q -enabling conditions for a standard Petri net. In other words, in a standard Petri net, a transition t is said to be q -enabled at a marking M if and only if (2) and (3) are satisfied (i.e., there is at least $q \times w(p, t)$ tokens in each input place of t and the number of tokens in each inhibitor place p does not exceed $w(p, t)$). The q -firing of a q -enabled transition t consists of firing the transition q times simultaneously.

2.5 Discrete firing example

As an example, consider an inventory control system represented in Fig. 2. The inventory control policy used in the system is a continuous review (s, S) policy specified by the immediate transition t_3 . The order-up-to-level of the policy are taken as $s = 3$ and $S = 10$ respectively, and the initial μ -marking of the net is $\mu_0 = (2, 2, \emptyset)$. The model is explained in more detail in section 3 (see Fig. 8). At the current μ -marking $\mu_0 = (2, 2, \emptyset)^T$, the discrete transition t_3 is enabled since $M(p_2) = 2 < 3$. After the firing of transition t_3 , a batch token with the size equal to $w(t_3, p_3) = 10 - M(p_2) = 8$ will be created in batch place p_3 and $w(t_3, p_2) = 8$ discrete tokens will be created in discrete place p_2 . Therefore, the new μ -marking of the net after the firing is: $\mu_1 = (2, 10, \{8\})^T$.

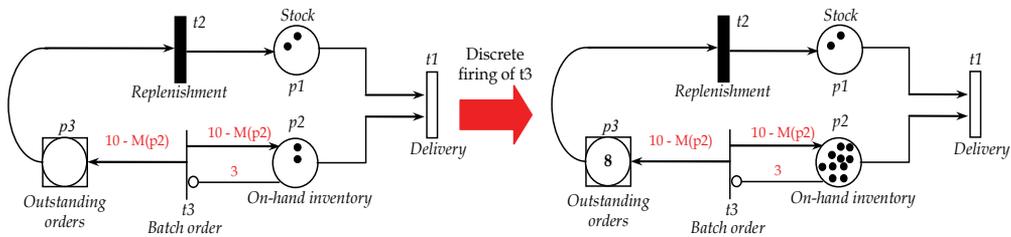


Fig. 2. An inventory control system (discrete firing example)

As shown in the two examples given in Fig. 1 and Fig 2, with their powerful graphical and mathematical formalism, BDSPNs are able to adequately describe the batch behaviour occurring at various stages of discrete event dynamic systems. Batch tokens are used to model batch quantities of material and/or information entities (batch customer orders, batch products, etc.). The sizes of the batch tokens have a quantitative meaning in defining the dynamic behaviour of the model. This is different from the concept of colors used in colored Petri nets (Jensen, 1997). The colors are rather qualitative attribute; programming languages are usually needed to define their data types and values. The BDSPN model keeps the

simplicity and the pertinence of standard discrete Petri nets while being able to model batch behaviours.

2.6 Analysis methods

As a mathematical tool, the BDSPN model has a number of properties. These properties, when interpreted in the context of the system modelled, allow the identification of the presence or absence of functional properties of the system. Besides the graphical representation, a fundamental advantage of the BDSPN is its capacity to systematically investigate many properties and characteristics of the system modelled. Specific analysis methods for BDSPN developed include: (1) the coverability (reachability) tree method, (2) the matrix algebraic approach, (3) reduction techniques, and (4) transformation techniques.

One ultimate goal for the introduction of the BDSPN model is to evaluate the performance of discrete event dynamic systems with batch behaviours, which requires a stochastic BDSPN model. It is clear that when the timing concept is considered in the model, particular policies should be specified to choose a batch token in each batch input place to fire its output transition and to resolve the conflict when multiple transitions are enabled. The temporal and the stochastic behaviour of the model are defined in our previous papers (Chen et al. 2005; Labadi et al. 2007). Similar to the existing stochastic Petri nets, the main performance analysis approach for BDSPN is based on the analysis of the stochastic marking process of the net. The approach is feasible particularly when the underlying stochastic process has a finite number of states (Labadi et al., 2007). For complex systems, simulation methods may be required (Chen et al., 2005, Amodeo et al. 2007).

2.7 Applications of the model

The BDSPN model increases the modelling and analysis power of the existing discrete Petri nets. It is able to describe essential characteristics of logistics systems (batch behaviours, randomness, operational policies, synchronization of various flows) and more generally discrete event dynamic systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. The model is particularly adapted for the modelling of flow evolution in discrete quantities (variable batches of different sizes) and makes it possible to describe more specific activities such as customer order processing, replenishment of stocks, production and delivery in a batch mode. The capability of the model to meet real needs is shown through applications dedicated to modelling and performance optimization of inventory control systems (Labadi et al., 2007) and a real-life supply chain (Chen et al., 2005; Amodeo et al., 2007).

3. Transformation into an equivalent classical Petri net

The objective of this section is to study the transformation of a BDSPN model into an equivalent classical Petri net model. Such a transformation is possible in some cases for which the corresponding transformation methods are developed. We will also show that for the model with variable arc weights depending on its marking, the transformation is impossible. This study establishes a relationship between BDSPNs and classical discrete Petri nets and demonstrates the necessity of the introduction of the BDSPN model.

3.1 Reachability graph

For the introduction of the transformation methods for the BDSPN model, we need to define two types of reachability graphs of the model. An illustrative example of the reachability concept of the model is given in Fig. 3. A μ -marking reachability graph of a given BDSPN is a directed graph (V_μ, E_μ) , where the set of vertices V_μ is given by the reachability set $(\mu_0^*$: all μ -markings reachable from the initial marking μ_0 by firing a sequence of transitions and the initial marking), while the set of directed arcs E_μ is given by the feasible μ -marking changes in the BDSPN due to transition firing in all reachable μ -markings. Similarly, we define M-marking reachability graph (V_M, E_M) which can be obtained from (V_μ, E_μ) by transforming each μ -marking in V_μ into its corresponding M-marking and by merging duplicated M-markings (and duplicated arcs).

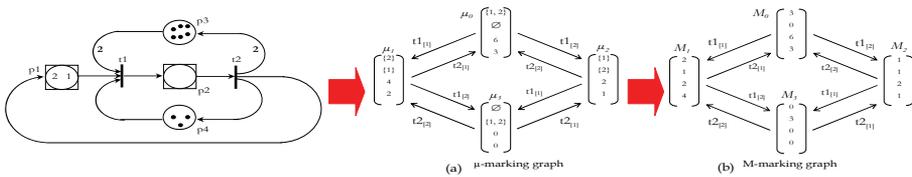


Fig. 3. An illustration of the μ -reachability and the M-reachability graphs

3.2 Special case with simple batch places

Firstly, we consider the case where all batch tokens in each batch place of the BDSPN are always identical (have the same size). A batch place p_i is said to be simple if the sizes of its all batch tokens are the same for any μ -marking reachable from μ_0 .

$$p_i \in P_b \text{ is a simple batch place} \Rightarrow \forall \mu \in \mu_0^*, \forall b \in \mu(p) : b = \text{constant} \tag{13}$$

In order to facilitate the description of the transformation method for this case, we consider an illustrative example given in Fig. 4. As shown in the figure, the net (a) whose all batch places are simple can be easily transformed into an equivalent classical discrete Petri net (b). We observe that the two nets have the same M-marking reachability graph (the same dynamical behaviour). Indeed, the two properties, (i) all batch places of the net are simple and (ii) the net has no variable arc weight, lead to a constant batch firing index q_j for each batch transition $t_j \in T_b$ of the net. As formulated in the following procedure, the transformation method consists of (i) transforming each batch place into a discrete place and (ii) integrating the constant batch firing index of each batch transition in the weights of its input and output arcs in the resulting classical net in order to respect the dynamic behaviour of the original batch net.

Transformation procedure (special case)

Given a batch deterministic and stochastic Petri net BDSPN = $(P, T, I, O, V, W, II, \mu_0)$ whose all batch places are simple and whose all arcs have a constant weight. This net can be transformed into an equivalent classical discrete Petri net, denoted by DPN = $(P^*, T, I, O, V, W^*, II, M_0)$, by the following procedure:

Step 1. The set of discrete places P_d of the BDSPN and their markings remain unchanged for the DPN.

$$\forall p_i \in P_d, M_0(p_i) = \mu_0(p_i). \tag{14}$$

Step 2. Each batch place of the BDSPN is transformed into a discrete place M-marked in the DPN.

$$\forall p_i \in P_b, M_0(p_i) = \sum_{b \in \mu(p_i)} b. \tag{15}$$

Step 3. The set of transitions T of the BDSPN remains unchanged for the DPN.

Step 4. The weight of each output arc of each batch place $p_i \in P_b$ of the BDSPN is set to the size of its batch tokens b_i .

$$\forall p_i \in P_b, \forall t_j \in p_i^*, w^*(p_i, t_j) = w(p_i, t_j) \times \frac{b_i}{w(p_i, t_j)} = b_i. \tag{16}$$

Step 5. The weight of each output arc of each batch transition $t_j \in T_b$ of the BDSPN is set to its original weight multiplied by its batch firing index q_j .

$$\forall p_i \in P_b, \forall t_j \in p_i^*, w^*(t_j, p_i) = w(t_j, p_i) \times q_j = w(t_j, p_i) \times \frac{b_i}{w(p_i, t_j)}. \tag{17}$$

Step 6. The weight of each output arc of each discrete transition $t_j \in T_d$ of the BDSPN remains unchanged for the DPN.

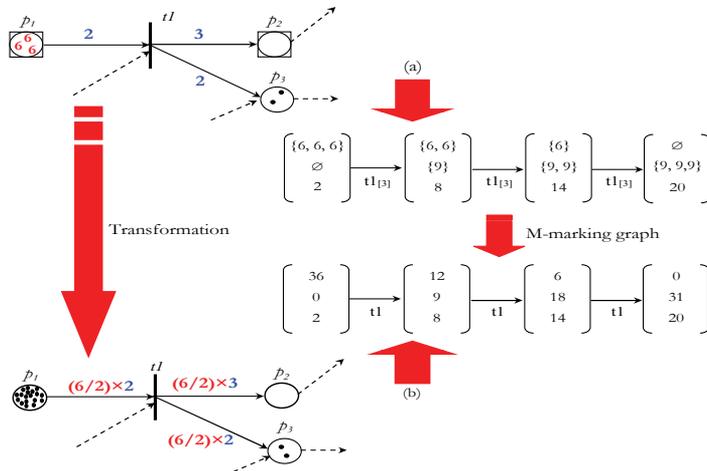


Fig. 4. Transformation of a BDSPN into an equivalent classical Petri net (special case)

3.3 General case

The proposed transformation procedure can be generalized to allow the transformation of a BDSPN containing batch places which are not simple into an equivalent classical Petri net.

The transformation is feasible if we know in advance all possible batch firings of all batch transitions and all possible batch tokens which can appear in each batch place of the net during its evolution. In other words, the transformation can be performed when we well know the dynamic behaviour of the BDSPN for its given initial μ -markings μ_0 .

- Let $D(t_j)$ denote the set of all q -indexed transitions $t_{j[q]}$ generated by the firings of the batch transition t_j with all possible batch firing indexes q during the evolution of the BDSPN starting from μ_0 . Formally:

$$D(t_j) = \{ t_{j[q]} \mid \exists \mu \in \mu_0^*, \mu[t_{j[q]} \rightarrow] \} \tag{18}$$

where μ_0 denotes the set of reachable μ -markings of the net from μ_0 and $\mu[t_{j[q]} \rightarrow]$ denote that the batch transition t_j can be fired from μ with a batch firing index q . For example, the batch transition t_1 of the net in Fig. 5(a) can be fired during the evolution of the net (see the reachability graph) by two possible batch firing indexes $q = 1$ and $q = 2$, then $D(t_1) = \{t_{1[1]}, t_{1[2]}\}$.

- Let $D(p_i)$ denote the set of all possible batch tokens which can appear in the batch place p_i during the evolution the BDSPN starting from its initial μ -marking μ_0 . Formally:

$$D(p_i) = \{ b \mid \exists \mu \in \mu_0^*, b \in \mu(p_i) \} \tag{19}$$

For example, the set of all possible batch tokens which can appear in the batch place p_1 of the BDSPN in Fig. 5(a) during its evolution starting from $\mu_0 = \{(1,2), \emptyset, 6, 3\}^T$ is $B(p_1) = \{1, 2\}$ as shown in the reachability graph.

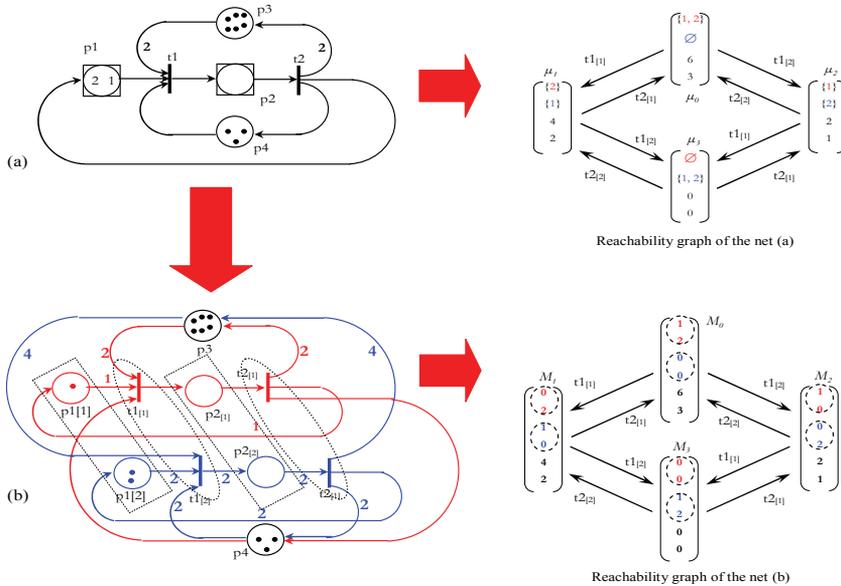


Fig. 5. Transformation of a BDSPN into an equivalent classical Petri net (general case)

By analogy with the transformation procedure for the special case, the transformation for the general case consists of the transformation of its each batch place p_i into a set of discrete places corresponding to $D(p_i)$ and the transformation of its each batch transition t_j into a set of discrete transitions corresponding to $D(t_j)$. For example, the transformation of the BDSPN into a classical Petri net given in Fig. 5 is realized by transforming the batch transition t_1 (resp. t_2) into a set of discrete transitions $\{t_{1[1]}, t_{1[2]}\}$ (resp. $\{t_{2[1]}, t_{2[2]}\}$) and by transforming the batch place p_1 (resp. p_2) into a set of discrete places $\{p_{1[1]}, p_{1[2]}\}$ (resp. $\{p_{2[1]}, p_{2[2]}\}$) as shown in Fig. 5(b). Similar to the special case, to respect the dynamical behaviour of the BDSPN, each possible batch firing index of each batch transition is integrated in the weights of the input and output arcs of the corresponding transition in the resulting classical net. After a close look of the reachability graphs of the two nets, we find that the two nets have the same behaviour. As illustrated in the figure, each μ -marking μ_i of the BDSPN corresponds to the marking M_i of the resulting classical Petri net. The M-marking of each batch place p_i is expressed by its corresponding set of discrete places $D(p_i)$. The transformation procedure for the general case is outlined in the following.

Transformation procedure (general case)

Step 1. The set of discrete places P_d of the BDSPN and their markings remain unchanged for the DPN.

$$p_i \in P_d, M_0(p_i) = \mu_0(p_i). \quad (20)$$

Step 2. Each batch place p_i of the BDSPN is converted into a set of discrete places $D(p_i)$ in the DPN such as:

$$D(p_i) = \{p_{i[b]} \mid b \in D(p_i)\} \text{ and } \forall p_{i[b]} \in D(p_i), M_0(p_{i[b]}) = \sum_{l \in \mu(p_i) \text{ and } l=b} l. \quad (21)$$

Step 3. Each batch transition t_j of the BDSPN is converted into a set of discrete transitions $D(t_j)$ in the DPN such that:

$$D(t_j) = \{t_{j[q]} \mid t_{j[q]} \in D(t_j)\} \quad (22)$$

The set of discrete transitions T_b of the BDSPN remains unchanged for the DPN.

Step 4. Each place $p_{i[b]} \in D(p_i)$ is connected to the output transitions $(p_{i[b]})^\bullet$ such that:

$$\forall p_{i[b]} \in D(p_i), (p_{i[b]})^\bullet = \{t_{j[q]} \mid t_j \in p_i^\bullet \text{ and } q = b / w(p_i, t_j)\} \quad (23)$$

$$\forall p_{i[b]} \in D(p_i), \forall t_{j[q]} \in (p_{i[b]})^\bullet, w(p_{i[b]}, t_{j[q]}) = w(p_i, t_j) \times b. \quad (24)$$

Step 5. Each transition $t_{j[q]} \in D(t_j)$ is connected to the output places $(t_{j[q]})^\bullet$ such that:

$$\forall t_{j[q]} \in D(t_j), (t_{j[q]})^* = \left\{ p_{i[b]} \mid (p_{i[b]} \in D(p_i)), (p_i \in t_j^* \cap P_d) \text{ and } (q = b / w(p_i, t_j)) \right\} \cup \left\{ p_i \mid p_i \in t_j^* \cap P_d \right\} \quad (25)$$

Clearly for each $t_{j[q]}$, its output places will be all discrete output places of transition t_j and all places generated by the batch output places of transition t_j . The weights of the corresponding arcs are given by:

$$\forall t_{j[q]} \in D(t_j), \forall (p_i \vee p_{i[b]}) \in (t_{j[q]})^*, w(t_{j[q]}, p_{i[b]}) = q \times w(t_j, p_i). \quad (26)$$

Step 6. Each place $p_{i[b]} \in D(p_i)$ is connected to the input transitions $(p_{i[b]})^*$ such that:

$$\forall p_{i[b]} \in D(p_i), (p_{i[b]})^* = \left\{ t_{j[q]} \mid t_j \in p_i \text{ and } q = b / w(t_j, p_i) \right\} \cup \left\{ t_j \in (p_i \cap P_d) \right\}. \quad (27)$$

Clearly for each $p_{i[b]}$, its input transitions will be all discrete input transitions of transition t_j and all transitions generated by the output batch transitions of transition t_j . The weights of the corresponding arcs are given by:

$$\forall p_{i[b]} \in D(p_i), \forall (t_j, t_{j[q]}) \in (p_{i[b]})^*, w(t_{j[q]}, p_{i[b]}) = q \times w(t_j, p_i). \quad (28)$$

Step 7. Each transition $t_{j[q]} \in D(t_j)$ will be connected to the set $(t_{j[q]})^*$ of input places such that:

$$\forall t_{j[q]} \in D(t_j), (t_{j[q]})^* = \left\{ p_{i[b]} \mid (p_i \in t_j \cap P_d) \text{ and } (q = b / w(p_i, t_j)) \right\} \cup \left\{ p_i \mid p_i \in t_j \cap P_d \right\} \quad (29)$$

Clearly for transition $t_{j[q]}$, its input places will be all discrete input places of transition t_j and all places generated by the output batch places of transition t_j . The weights of the corresponding arcs are given by:

$$\forall t_{j[q]} \in D(t_j), \forall (p_i \vee p_{i[b]}) \in (t_{j[q]})^*, w(p_{i[b]}, t_{j[q]}) = q \times w(p_i, t_{j[q]}). \quad (30)$$

Step 8. The arcs which connect discrete places with discrete transitions in the BDSPN and their weights remain unchanged in the DPN.

3.3 Case with inhibitor arcs

The transformation is also possible for BDSPNs with inhibitor arcs whose weights are constant. We will illustrate it by using some examples.

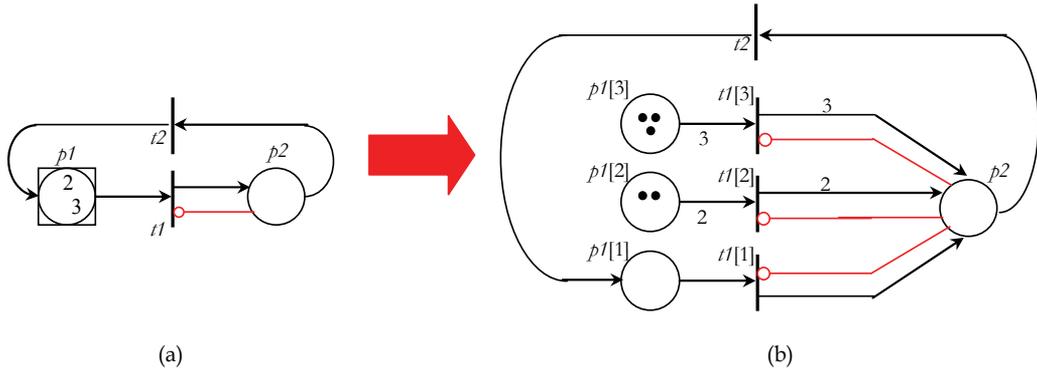


Fig. 6. Transformation of a BDSPN with inhibitor arc connecting a discrete place to a batch transition

Sub-case 1: Inhibitor arc connecting a discrete place to a batch transition.

As shown in the net depicted in Fig. 6(a), in the case where there is an inhibitor arc connecting a discrete place p_i to a batch transition t_j , the corresponding inhibitor condition must be reproduced in the resulting classical Petri net for all q -indexed transitions $t_{j|q}$ generated by the batch transition t_j . Clearly, in this example, the batch transition t_1 can be fired with three possible batch firing indexes during the evolution of the net. In other words, the transition t_1 generates three possible q -indexed transitions $t_{1|1}$, $t_{1|2}$, $t_{1|3}$. Thus, in the corresponding classical Petri net there are three inhibitor arcs which connect the discrete place p_2 to the three q -indexed transitions, respectively. The reason for the duplication of the inhibitor arc is obvious: Firing the transition t_1 with one of the three possible batch firing indexes must respect the enabling condition $M(p_2) < w(t_1, p_2)$ in the net (a). In the classical net, the condition is expressed as $M(p_2) < w(t_{1|q}, p_2)$ for each q -indexed transition generated by t_1 . It is easily to observe that the two nets are identical in terms of their dynamical behaviours.

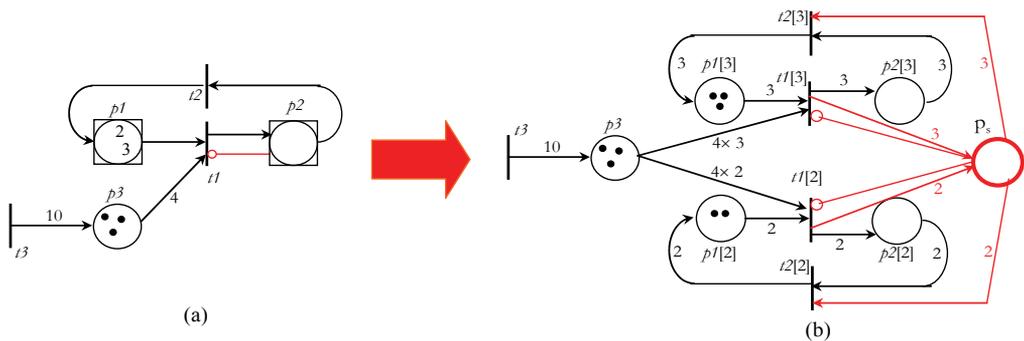


Fig. 7. Transformation of a BDSPN with inhibitor arc connecting a batch place to a transition

Sub-case 2: Inhibitor arc connecting a batch place to a transition

We now consider the case as shown in Fig. 7(a) where there is an inhibitor arc connecting a

batch place to a transition. The enabling of the transition t_1 for a given batch firing index q in the net (a) must satisfy the condition $M(p_2) < w(t_1, p_2)$ imposed by the inhibitor arc. After the transformation of each batch place (resp. batch transition) into a set of discrete places (resp. a set of transitions), we observe that to respect the enabling condition imposed by the inhibitor arc in the net (a), it is necessary to capture the total marking of the discrete places generated by the batch place p_2 by using a supplementary place p_s in the classical Petri net.

3.4 Case of the timed model

The transformation techniques discussed so far do not consider temporal and/or stochastic elements in a BDSPN, but they can be adapted for timed and stochastic BDSPN models. The basic idea is as follows: Each discrete transition in the BDSPN model keeps its nature (immediate, deterministic, stochastic) in the resulting classical Petri net. The q -indexed transition $t_{j|q}$ which may be generated by each batch transition t_j has the same nature as the transition t_j . Other elements of the BDSPN model may also be taken into account in the resulting classical model such as the execution policies; the firing priorities of some transitions; etc.

3.5 Necessity of the BDSPN model

In this section, the necessity of the introduction of the BDSPN model is demonstrated through an analysis of the transformation procedures presented in the previous section. The advantages of the model are discussed in two cases: the case where a BDSPN can be transformed into a classical Petri net and the case where the transformation is impossible.

Case 1 - the BDSPN model is transformable

In the case where the transformation is possible, the advantages of the BDSPN model are outlined in the following:

- As shown in the transformation procedures developed in this section, we note that the resulting classical Petri net depends on the initial μ -marking of the BDSPN. Obviously, if we change the initial μ -marking of the BDSPN given in Fig. 5(a), we will obtain another classical Petri net. For example, if there is another batch token of different size in the batch place p_1 , all the structure of the corresponding classical Petri net must be changed. In fact, the batch places of the BDSPN may not generate the same set of q -indexed transitions $D(t_j)$ for each batch transition t_j and may not generate the same set of discrete places $D(p_i)$ for each batch place p_i during the evolution of the net.
- The transformation of a given BDSPN model into an equivalent classical Petri net may lead to a very large and complex structure. According to the transformation procedure developed in subsection 3.2, the number of places $|P^*|$ and the number of transitions $|T^*|$ in the equivalent classical Petri net are given by:

$$|P^*| = |P_d| + \sum_{i=1}^{|P_b|} |D(p_i)| \quad (31)$$

$$|T^*| = |T_d| + \sum_{j=1}^{|T_b|} |D(t_j)| \quad (32)$$

where $|P_b|$ is the number of the batch places; $|P_d|$ is the number of the discrete places; $|T_b|$ is the number of the batch transitions; $|T_d|$ is the number of the discrete transitions of the given BDSPN. $D(t_j)$ is the set of q-indexed transitions generated by each batch transition $t_j \in T_b$ and $D(p_i)$ is the set of all possible batch tokens which appear in each batch place $p_i \in P_b$ during the evolution of the BDSPN.

As an example, if a BDSPN contains 10 batch places, 10 batch transitions, and its evolution leads to 10 different possible batch tokens in each batch place, and 10 different batch firing indexes for each batch transition, then we need at least 100 places and 100 transitions to construct an equivalent classical Petri net. Note that the number of arcs to be introduced in the classical Petri net is also very large. We see a much higher complexity of the resulting Petri net compared to the original BDSPN.

Case 2 - the BDSPN is not transformable

The modelling of some discrete event systems such as inventory control systems and logistic systems, as shown in (Labadi et al. 2007,2005), require the use of the BDSPN model with variables arc weights depending on its M-marking and possibly on some decision parameters of the systems. It is the case of the BDSPN model of an inventory control system whose inventory replenishment decision is based on the inventory position of the stock considered and the reorder and order-up-to-level parameters (see Fig. 8). The modelling of such a system is possible by using a BDSPN model with variables arc weights depending on its M-marking. This kind of BDSPN model is particularly adapted for the modelling of flow evolution in discrete quantities (variable batches of different sizes).

The BDSPN model shown in Fig. 8 represents an inventory control system where its operations are modelled by using a set of transitions: generation of replenishment orders (t_3); inventory replenishment (t_2); and order delivery (t_1) that are performed in a batch way because of the batch nature of customer orders represented by batch tokens in batch place p_4 and the batch nature of the outstanding orders represented by batch tokens in batch place p_3 . In the model, the weights of the arcs (t_3, p_2), (t_3, p_3) are variable and depend on the parameters s and S of the system and on the M-marking of the model ($S - M(p_2) + M(p_4)$; $s + M(p_4)$). The model may be built for the optimization of the parameters s and S . In this case, the techniques for the transformation of the BDSPN model into an equivalent classical Petri net model proposed in the previous section is not applicable.

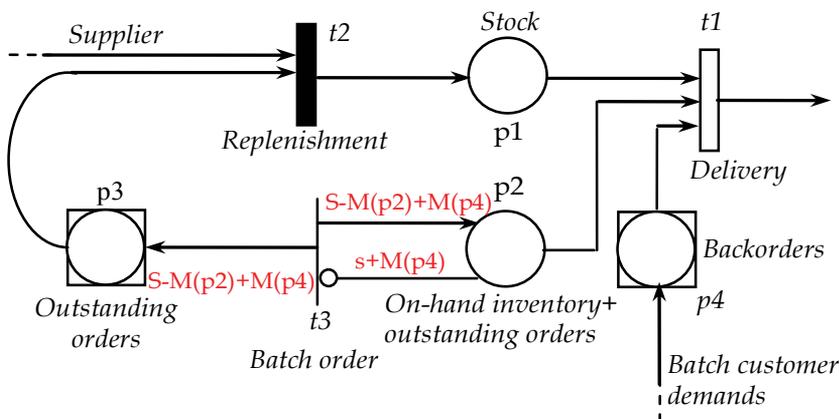


Fig. 8. BDSPN model of an inventory control system: a model that is not transformable

In fact, contrary to the example given in Fig. 5, in this model, the sizes of the batch tokens that may be generated depend on both the initial μ -marking of the model and the parameters s and S . In other words, a change of the decision parameters s and S of the system or the initial μ -marking of the model will lead to another way of the evolution of the discrete quantities (variable batches of different sizes). Moreover, the appearance of stochastic transitions in the model makes more difficult to characterize all possible sizes of the batch tokens that are necessary to be known for the application of the transformation methods.

4. Transformation into an associated discrete Petri net

In this section, the associated discrete Petri net of a BDSPN is defined and the relationships between the two models are then explored. Because the objective is to develop structural (qualitative) analysis methods for the BDSPN model, we confine ourselves hereafter to the untimed version of BDSPN model obtained by removing the firing times of all transitions from the model. Before giving a formal definition of the associated discrete Petri net, we recall the state equation and the q -firing of a transition of the model.

4.1 Incidence matrix and state equation

Suppose that the current μ -marking of a BDSPN is μ_k and transition t_j is enabled at the μ -marking. The firing of the transition will result in a new marking μ_{k+1} written as:

$$\forall p \in \bullet t_j, \mu_{k+1}(p) = \mu_k(p) - q_j^k \times w(p, t_j) \quad (33)$$

$$\forall p \in t_j \bullet, \mu_{k+1}(p) = \mu_k(p) + q_j^k \times w(t_j, p) \quad (34)$$

$$\forall p \notin (\bullet t_j \cup t_j \bullet), \mu_{k+1}(p) = \mu_k(p) \quad (35)$$

where q_j^k is the batch firing index of the transition t_j at the μ -marking μ_k such that:

$$q_j^k = \begin{cases} 1 & \text{if } \bullet t_j \cap P_b \neq \emptyset \\ b / w(p, t) & b \in \mu_k(p) \text{ if } \bullet t_j \cap P_b \neq \emptyset \end{cases} \quad (36)$$

Similar to classical Petri nets, the incidence matrix W of a BDSPN with m places and n transitions is an $n \times m$ matrix $W = [w_{t,p}]$ whose entries are defined as:

$$w_{t,p} = w(t, p) - w(p, t) \quad (37)$$

where $w(t, p)$ is the weight of the arc from transition t to its output place p and $w(p, t)$ is the

weight of the arc to transition t from its input place p . The equations (33), (34), and (35) can then be written in matrix form (38), called state equation, as:

$$\mu_{k+1} = \mu_k + W \times Q \quad (38)$$

where μ_{k+1} is the μ -marking obtained (reached) after the firing of transition t_j from μ -marking μ_k . μ_{k+1} and μ_k are both $m \times 1$ column vector. The entry $\mu_{k+1}(p)$ is the number of discrete tokens if p is a discrete place or the multiset of batch tokens (positive integers) if p is batch place. Q is an $n \times 1$ column vector with $(n - 1)$ zero entries and one nonzero entry. A nonzero entry q_j^k in the j^{th} position indicates that transition t_j is fired at the k^{th} firing. Q is called firing count vector defined as:

$Q = [q_j^k]_{n \times 1}$ such as :

$$q_j^k = \begin{cases} 1 & \text{if } (\bullet t_j \cap P_b) \neq \emptyset \\ b / w(p, t_j) & b \in \mu_k(p) \text{ if } (\bullet t_j \cap P_b) \neq \emptyset \end{cases} \quad (39)$$

4.2 The q-firing of a transition

The q -firing of a q -enabled transition t consists of firing the transition q times simultaneously. In a classical Petri net, a transition t_j is said to be q -enabled at a marking M if and only if:

$$\forall p_i \in \bullet t_j, M(p_i) \geq q \times w(p_i, t_j) \quad (40)$$

$$\forall p_i \in \circ t_j, M(p_i) < w(p_i, t_j) \quad (41)$$

From the above definition, a transition t_j is q -enabled if and only if there is at least $q \times w(p_i, t_j)$ tokens in each input place of t_j and the number of tokens in each inhibitor place p_i does not exceed $w(p_i, t_j)$. The q -firing of a q -enabled transition t_j consists of a sequential firing of t_j where t_j is fired q times.

The matrix equation which characterizes a q -firing of transition t_j is as follows:

$$M_{k+1} = M_k + W \times U \times q \quad (42)$$

where $U[i] = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$

We can also write the equation (42) as:

$$M_{k+1} = M_k + W \times Q \tag{43}$$

where $Q[i] = \begin{cases} 0 & \text{if } i \neq j \\ q & \text{if } i = j \end{cases}$

Note that for a BDSPN the equation (43) is similar to the state equation (38) if we consider the M-marking. This leads us to a certain dynamic analogy between a BDSPN and its associated discrete Petri net.

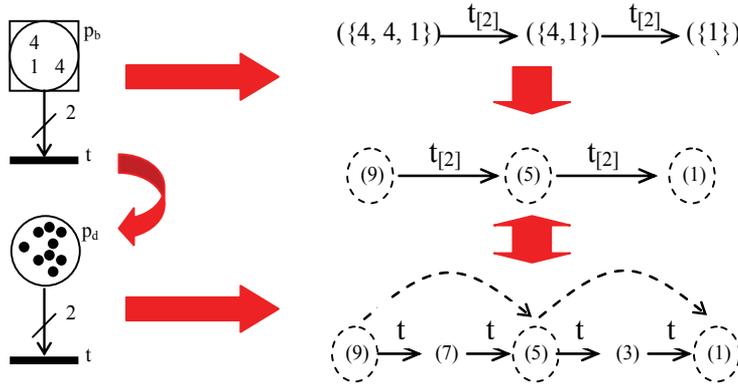


Fig. 9. An illustration of the q-firing concept in a BDSPN and a classical Petri net

The basic idea of the analogy is illustrated by an example given in Fig. 9, where the M-marking graph of a BDSPN is presented in an unfolded form in the marking graph of its associated discrete Petri net. As shown in the previous section, the discrete enabling and firing rules of a discrete transition t can be regarded as a special case of the batch enabling and firing rules of a batch transition. For $q = 1$, the batch rules are reduced to the discrete rules. A close look of the enabling conditions (2) and (3) finds that they are actually the q-enabling conditions for a standard Petri net. In other words, in a standard Petri net, a transition t is said to be q-enabled at a marking M if and only if (2) and (3) are satisfied (i.e., there is at least $q \times w(p, t)$ tokens in each input place of t and the number of tokens in each inhibitor place p does not exceed $w(p, t)$).

4.3 Associated discrete Petri net

Any BDSPN can be associated with a discrete Petri net, denoted by DPN , which is obtained by:

- Transforming each batch place into a discrete place with the initial number of tokens in the discrete place equal to the initial total size of the batch tokens in the batch place, i.e.,

$$\forall p_i \in P_b, M_0(p_i) = \sum_{\mu_p \in \mu_b(p_i)} \mu_p$$

- Keeping all its transitions and arcs unchanged in the associated discrete Petri net.

An illustrative example of the transformation of a BDSPN into its associated discrete Petri net is given in Fig. 10, where an order treatment system is modelled by both the BDSPN model (a) and its associated classical discrete Petri net (b). Each transition t_j (t_1 and t_2), in the two Petri nets, represents the same operation. Contrary to those in the classical model (b), the places p_2 and p_3 in the BDSPN are batch places with batch tokens.

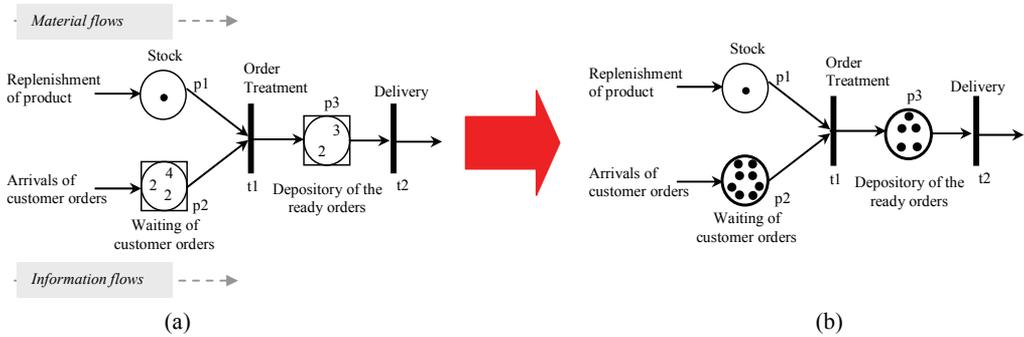


Fig. 10. Synchronization and coordination of material and information flows

In the BDSPN model (a), customer orders with different sizes arrive and are recorded in the batch place p_2 where they wait for treatment according to their dates of arrival. In the example, three customer orders currently arrived in the batch place p_2 : two customer orders with size 2 and one customer order with size 4 (i.e., $\mu(p_2) = \{2, 4, 2\}$). The customer orders will be filled, according to their orders of arrival in batch place p_2 , from the stock of finished products, represented by the discrete place p_1 , if the stock is sufficient. At the current μ -marking of the BDSPN, there is only one unit of finished product available in the stock (i.e., $\mu(p_1) = 1$). The transition t_2 is not enabled and the customer orders must keep waiting until stock is sufficient to fill any of the orders. Contrary to the behaviour of the BDSPN, in the classical model (b), the orders are recorded in the discrete place p_2 . In this representation, there are no informations about the various sizes of the customer orders, except for their total size $M(p_2) = 8$ which corresponds to the M-marking of the place p_2 in the BDSPN model (sum of the sizes of its batch tokens). In this case the transition t_2 is enabled at the current marking of the net. The place p_3 in the two models corresponds to the customer orders ready for delivery. The delivery is carried out in batch mode in the BDSPN by firing the transition t_2 and each customer will receive his/her order as it is ordered. This is not the case in the discrete Petri net model where the delivery is carried out by multiple firings of the discrete transition t_2 .

This illustration clearly shows that the BDSPN and its associated discrete Petri nets behave differently. However, despite this difference, the BDSPN and its associated discrete Petri net have several common qualitative properties. The rest of this section is thus dedicated to establish formal relationships between the two models.

4.4 Relationships between a BDSPN and its associated discrete Petri net

Before conducting a formal analysis of a BDSPN and its associated discrete Petri net to explore the relationships, we first examine an illustrative example given in Fig. 11.

Important observations can be obtained by a careful investigation of the two marking graphs in the figure. The M-marking graph of the BDSPN is reproduced in an unfolded form in the marking graph of its associated discrete Petri net. In fact, each one-step state transition $M_k[t_{j|q}] \rightarrow M_{k+1}$ appeared in the M-marking graph of the BDSPN has its corresponding multi-step state transition presented as $M_k[t_j t_j \dots t_j \rightarrow M_{k+1}$ in the marking graph of its associated discrete Petri net. For instance, the batch firing of the transition t_1 with batch firing index $q = 2$ (i.e, firing of $t_{1|2}$) at the M-marking M_0 which leads to a new M-marking M_2 (i.e., $M_0[t_{1|2}] \rightarrow M_2$) in the M-marking graph of the BDSPN has its corresponding multi-step transition firing presented as $M_0[t_1 t_1] \rightarrow M_2$ in the marking graph of the discrete Petri net. In other words, each batch firing of a batch transition t_j with a batch firing index q at a given M-marking M_k can be interpreted as q discrete firings occurred concurrently in the associated discrete Petri net. Consequently, the set of reachable M-marking of the BDSPN is included in the set of reachable marking of its associated discrete Petri net.

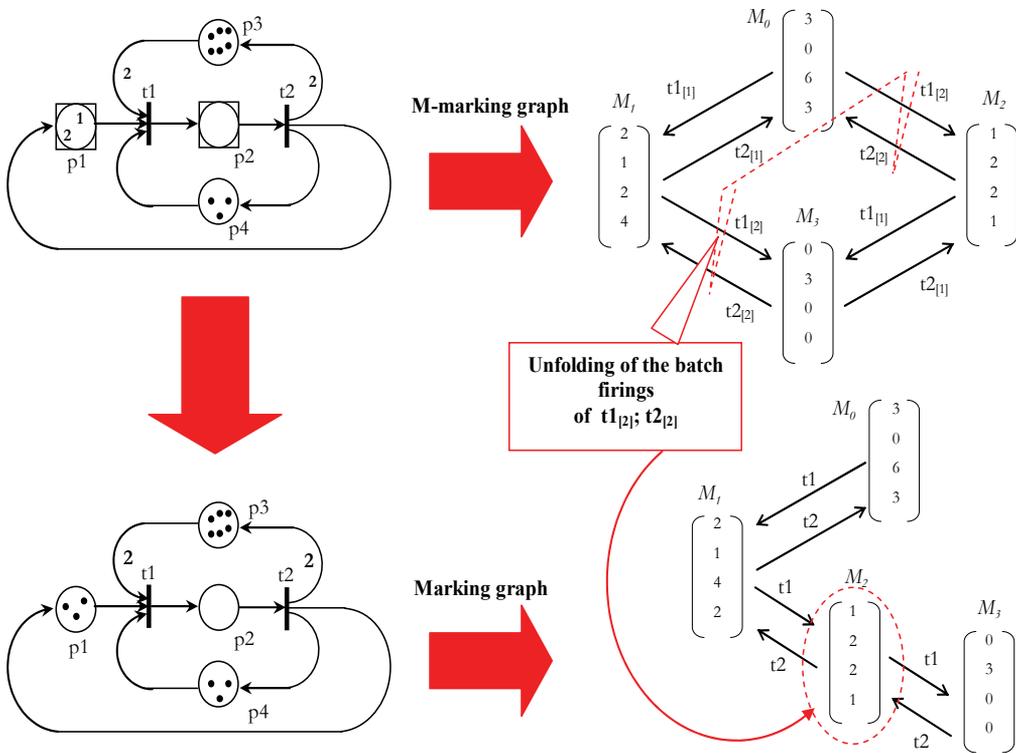


Fig. 11. An illustration of the behaviours of a BDSPN and its associated discrete Petri net

As shown in Fig. 12, due to the unfolding operation of batch firings, the associated discrete Petri net may generate some intermediate states (markings) between two states (markings) in the BDSPN model. In general case, the intermediate states may be evaluated to other states which do not appear in the BDSPN model.

As shown by an example given in Fig. 13, when the BDSPN contains an inhibitor arc, some of its reachable M-markings are not reproduced by its associated discrete Petri net. It is

simply due to the different effect of the inhibitor arc on the behaviours of the two nets. Contrary to the BDSPN, in the associated discrete Petri net, the inhibitor arc prevents the occurrence of the sequence of firings of the transition t_1 which corresponds to the batch firings of $t_{1[3]}$ and $t_{1[2]}$ in the BDSPN.

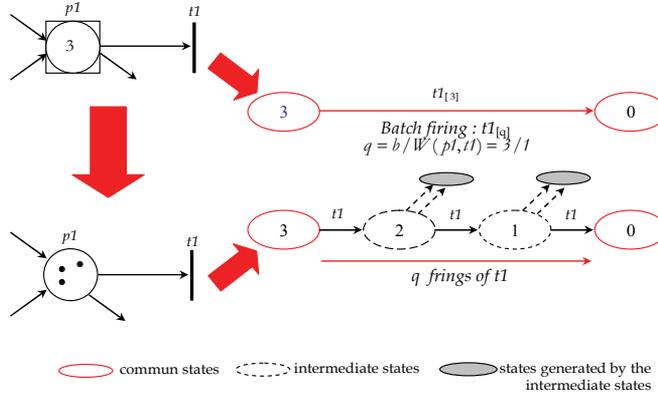


Fig. 12. Common and intermediate states of a BDSPN and its associated discrete Petri net

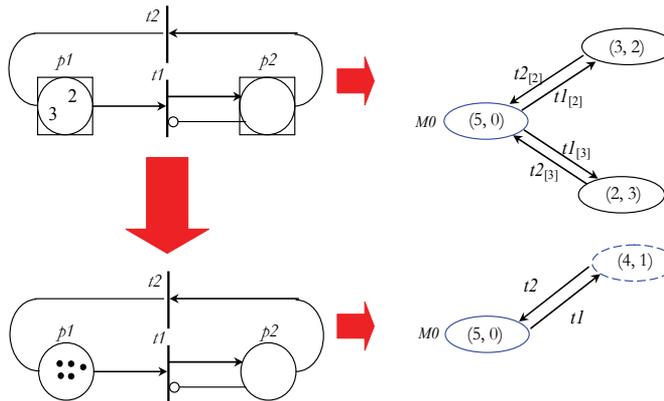


Fig. 13. Illustration of the case of a BDSPN with inhibitor arcs.

The above discussion leads to some important properties about the behaviours of the BDSPN and its associated discrete Petri. In the following, it is assumed that the BDSPN model has no inhibitor arc. We use $G_\mu(\text{BDSPN}, \mu_0)$, $G_M(\text{BDSPN}, M_0)$ and $G_M(\text{DPN}, M_0)$ to denote the μ -marking reachability graph of the BDSPN, the M-marking reachability graph of the BDSPN, and the reachability graph of the associated discrete Petri net DPN , respectively.

• **Property 1.** A BDSPN and its associated discrete Petri net have the same initial M-marking M_0 and the same incidence matrix W .

Proof. This property is a direct consequence of the definition of the associated discrete Petri net.

• **Property 2.** The set of the reachable M-markings of a BDSPN, M^*_{BDSPN} , is included in the set of the reachable markings, M^*_{DPN} , of its associated discrete Petri net, i.e.,

$$M_{BDSPN}^* \subseteq M_{DPN}^* \Leftrightarrow \forall M \in M_{BDSPN}^* \Rightarrow M \in M_{DPN}^*$$

Proof: The property 1 implies that: $\forall M_0 \in M_{BDSPN}^* \Rightarrow M_0 \in M_{DPN}^*$

Let t_j be a transition enabled at the initial μ -marking μ_0 (its corresponding M-marking M_0) in the BDSPN. The transition is also enabled at the marking M_0 in the associated discrete Petri net.

i) If t_j is a discrete transition, after the firing of the transition t_j in the two nets, their M-marking (marking) will be changed to the same marking M_1 . Thus, the marking M_1 appears in the two reachability graphs $G_M(BDSPN, M_0)$ and $G_M(DPN, M_0)$. We then have:

$$M_1 \in M_{BDSPN}^* \Rightarrow M_1 \in M_{DPN}^*$$

ii) If t_j is a batch transition in the BDSPN, then it can be fired at M_0 with a batch firing index q . This batch firing corresponds to q consecutive firings of the transition in the associated discrete Petri net as represented by:

$$M_0 \Big| t_{j[q]} \rightarrow M_k \Leftrightarrow M_0 \Big| \underbrace{t_j t_j \dots t_j}_{q \text{ times}} \rightarrow M_k$$

We then have: $M_k \in M_{BDSPN}^* \Rightarrow M_k \in M_{DPN}^*$

Continuing this process, for each new reachable marking M_k in the BDSPN, it is reachable in the associated discrete Petri net.

• **Property 3.** All feasible sequences of firings (multiple transitions which can be fired consecutively) in a BDSPN are also feasible in its associated discrete Petri net, i.e.,

$$\forall S \in G_\mu(BDSPN, \mu_0) \Rightarrow S \in G_M(DPN, \mu_0)$$

Proof: This property is an implication of the property 2 which says that all reachable M-markings of the BDSPN are included in the set of reachable markings of its associated discrete Petri net. For example, consider $S = t_{3[2]} t_1 t_{2[4]}$ as a feasible sequence by a given BDSPN from M to M' .

$$M[t_{3[2]}t_1t_{2[4]} \rightarrow M' \in G_\mu(BDSPN, \mu_0) \Rightarrow M[t_3t_3t_1t_2t_2t_2t_2 \rightarrow M' \in G_M(DPN, M_0)$$

• **Property 4.** If the associated discrete Petri net of a BDSPN is bounded then the BDSPN is also bounded (DPN is bounded \Rightarrow BDPN is bounded).

Proof: By definition, a DPN is bounded if all its places are bounded. Formally:

$$\forall p \in P, \forall M \in G_M(DPN, M_0) \Rightarrow M(p) \leq k, (k \in \mathbb{IN})$$

However, all the M-markings of the BDSPN are included in the set of reachable markings of its associated discrete Petri net, then: $\forall p \in P, \forall M \in G_M(BDSPN, M_0) \Rightarrow M(p) \leq k$. Hence the BDSPN is bounded.

• **Property 5.** The boundness of a BDSPN does not imply the boundness of its associated discrete Petri net.

Proof. As illustrated in Fig. 13, the associated discrete Petri net of a BDSPN may generate some intermediate states (markings) which do not appear in the BDSPN. These intermediate markings, which are not included in the states of the BDSPN, may be unbounded.

• **Property 6.** All P-invariants (resp. T-invariants) of the associated discrete Petri net of a BDSPN are also P-invariants (resp. T-invariants) of the BDSPN.

Proof. By definition, every P-invariant (resp. T-invariant) of a net is a solution of the equation $Y^T \times W = 0$ (resp. $W \times X = 0$), where W is the incidence matrix of the net. Since the BDSPN and its associated discrete Petri net have the same incidence matrix W , they have the same P-invariants and T-invariants.

- **Property 7.** The liveness (resp. the reversibility) of a BDSPN can not be concluded from the liveness (resp. the reversibility) of its associated discrete Petri net.

Proof. As we know, intermediate markings may appear in the reachability set of the associated Petri net. At these intermediate markings, transitions may be fired, leading to other markings (see Fig. 13). For the reversibility of the associated discrete Petri net, it is possible that the return to the initial marking starting from a given marking is made through a sequence of transitions which is not feasible in the BDSPN. Moreover, because of possible transition firings at the intermediate states, the liveness of the associated discrete Petri net does not imply the liveness of the BDSPN due to the existence of some sequences of transitions which are not feasible in the BDSPN.

- **Property 8.** A BDSPN is reversible if its associated discrete Petri net is reversible. In this case, all reachable markings of the two nets are identical.

Proof. By definition, the associated discrete Petri net is reversible if for every reachable marking M there exists a sequence of transitions whose firing reproduces M_0 . That is,

$$\forall M \in M_{DPN}^*, M_0 = M \times W$$

Since the BDSPN and its associated discrete Petri net have the same incidence matrix, they have the same set of reachable markings. Hence, the BDSPN is reversible.

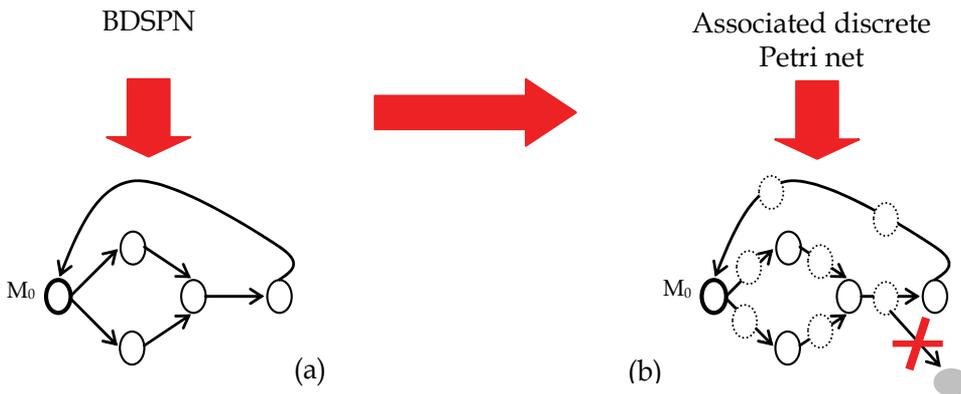


Fig. 14. Illustration of the ideal configuration between a BDSPN and its associated discrete Petri net

From the above discussion, we know that although a BDSPN and its associated discrete Petri net behave differently, formal relationships between the two models and their common qualitative properties can be explored. More results can be derived from the properties presented above. Intuitively, a case with common qualitative properties (liveness, reversibility, boundness, ...) of the BDSPN and its associated discrete Petri net is illustrated in Fig. 14.

5. Conclusion

In the first part of this chapter, a new class of Petri nets, called batch deterministic and stochastic Petri nets (BDSPN), is presented as a powerful modelling and performance

evaluation tool for discrete event dynamic systems with batch behaviours. The model enhances the modelling and analysis power of the existing discrete Petri nets. It is particularly adapted for the modelling of flow evolution in discrete quantities (variable batches of different sizes) and is capable of describing activities such as customer order processing, stock replenishment, production and delivery in a batch mode. The model and its associated analysis methods are particularly suitable for the modelling and analysis of industrial and manufacturing systems where materials are processed in batches and operations are performed in batch modes to take advantages of the economies of scale or because of the batch nature of customer orders. As demonstrated in our previous applications (Amodeo et al., 2007; Labadi et al., 2007, Chen et al., 2005), the model is able to describe essential characteristics of logistics and inventory control systems (batch behaviours, batch operational policies, synchronization of various flows, randomness) and more generally discrete event dynamic systems with batch behaviours. The second part of this chapter contributes to the structural and behavioral analysis of the model by using the transformation approach. The first method proposed transforms a BDSPN into an equivalent classical discrete Petri net under some conditions. In this case, the corresponding transformation procedures are presented. For other cases, especially for BDSPNs with variable arc weights depending on their marking, the transformation is impossible. The second method analyzes a BDSPN based on its associated discrete Petri net which is obtained by converting the batch components (batch places, batch tokens, batch transitions) of the BDSPN into discrete components of the discrete Petri net. We show that although a BDSPN and its associated discrete Petri net behave differently, they have several common qualitative properties. This study establishes a relationship between BDSPNs and classical discrete Petri nets and demonstrates the necessity of the introduction of the BDSPN model. In the future, we intend to develop new efficient methods for analysis of the BDSPN model based on the obtained formal relationships between the BDSPNs and classical discrete Petri nets and by exploring existing results for classical Petri nets. With the formal relations obtained in this work, we think that some results of classical Petri nets can be adapted and extended for the BDSPN model. On the other hand, we want to develop reduction rules for the BDSPNs following the methodology of the transformation approach. In this case, the BDSPN model will not be converted into a classical Petri net as shown in this chapter, but the size of the model will be reduced. The reduced model can be used for deriving qualitative and/or quantitative properties of the original BDSPN model and the analysis of the original model is thus simplified. This expected research aims at the development of an automated toolset which can help us to efficiently model and analyze untimed, timed and stochastic discrete systems with batch behaviours.

6. References

- Ajmone Marsan, M, Balbo, G., Conte, G., Donatelli, S. & Franceschinis, G. (1995). Modelling with Generalized Stochastic Petri Nets, *John Wiley and Sons, ISBN: 0-471-93059-8, 1995.*
- Amodeo, L., Chen, H. & El Hadji, A. (2007). Multiobjective Supply Chain Optimization: An Industrial Case Study, *Lecture Notes in Computer Science, Applications of Evolutionary Computing, ISBN: 978-3-540-71804-8, Vol. 4448, pp. 732-741, Springer Berlin Heidelberg, 2007.*

- Chen, H., Amodeo, L. & Boudjeloud, L. (2003). Supply chain optimization with Petri Nets and genetic algorithms, *Proceedings of IEEE International Conference on Industrial Engineering and Production Management*, ISBN: 2-930294-13-02, vol. 2, pp. 49-58, *Proceedings FUCAM Editors, Porto, Portugal, May 2003*.
- Chen, H., Amodeo, L., Chu, F., and Labadi, K. (2005). Modelling and performance evaluation of supply using batch deterministic and stochastic Petri nets, *IEEE transactions on Automation Science and Engineering*, ISSN: 1545-5955, Vol.2, N°2, pp. 132-144, April 2005.
- Ehrig, H., Engels, G., Krewski, H.-J. & Rozenberg G., (editors). Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications, Languages and Tools. *World Scientific*, ISBN 981-02-4020-1, 1999.
- Haddad, S., "A reduction theory for colored Nets", European Workshop on applications and theory in Petri Nets", *Lecture Notes in Computer Science, Advances in Petri Nets 1989*, ISBN 978-3-540-52494-6, pp. 209-235, Springer Berlin / Heidelberg, 1989.
- Jensen, K. (1997). Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Vol. I, Basic Concepts. Monographs in Theoretical Computer Science, *Springer-Verlag*, 2nd corrected printing, ISBN: 3-540-60943-1, London, UK, 1997.
- Juan, E.Y.T., Tsai, Jeffrey J.P., Murata, T. & Zhou, Yi., (2001). Reduction Methods for real-Time Systems Using Delay Time Petri nets, *IEEE Transactions on Software Engineering*, ISSN: 0098-5589, Vol. 27, N°5, pp. 422-448, May 2001.
- Labadi, K., Chen, H. & Amodeo, L. (2005). Application des BDSPNs à la Modélisation et à l'Evaluation de Performance des Chaînes Logistiques", *Journal Européen des Systèmes Automatisés*, DOI:10.3166/jesa.39.863-886, Vol.39/7, pp. 863-886, 2005.
- Labadi, K., Chen, H. & Amodeo, L. (2007). Modeling and Performance Evaluation of Inventory Systems Using Batch Deterministic and Stochastic Petri Nets. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, ISSN: 1094-6977, Vol. 37, N°6, pp. 1287-1302, 2007.
- Lee-Kwang, H. & Favrel, J., (1985). Hierarchical Reduction Methods for analysis and Decomposition of Petri Nets, *IEEE Transactions on Systems, Man, and Cybernetics*, ISSN 0018-9472, Vol. 15, pp. 272-280, March 1985.
- Lee-Kwang, H. & Favrel, J., and Paptiste, P. (1987). Generalized Petri Nets Reduction Method", *IEEE Transactions on Systems, Man, and Cybernetics*, ISSN: 0018-9472, Vol. 17, N°2, pp. 297-303, April 1987.
- Li, Yao. & Woodside, C. Murray., (1995). Complete Decomposition of Stochastic Petri Nets Representing Generalized Service Networks, *IEEE Transactions on Computers*, ISSN: 0018-9340, Vol. 44, N° 4, pp. 577-592, April 1995.
- Lindemann, C., (1998). Performance Modelling with Deterministic and Stochastic Petri Nets, *John Wiley and Sons*, ISBN: 0471976466, New York, USA, 1998.
- Ma, J. & Zhou, M.C., (1992). Performance Evaluation of Discrete Event Systems via Stepwise Reduction and Approximation of Stochastic Petri Nets, *Proceedings of the 31st IEEE Conference on decision and Control*, ISBN: 0-7803-0872-7, Vol.1, pp. 1210-1215, Tucson, Arizona, USA, 1992.
- Murata, T. (1989). Petri Nets: Properties, Analysis and Applications, *Proceedings of the IEEE*, ISSN: 0018-9219, Vol. 77, N° 4, pp. 541-580, April, 1989.
- Thapa, D., Dangol, S., & Wang, Gi-Nam. (2005). Transformation from Petri Nets Model to Programmable Logic Controller using One-to-One Mapping technique.

Computational Intelligence for Modelling, Control and Automation, ISBN: 0-7695-2504-0, Vol. 2, pp. 228-233, Nov. 2005.

Wang, J., Deng, Yi. & Zhou M.C., (2000). Compositional Time Petri Nets and Reduction Rules, *IEEE Transactions on Systems, Man, and Cybernetics, Part B, ISSN: 1083-4419, Vol. 30, N° 4, pp.562-572, August 2000.*

Automatic Estimation of Parameters of Complex Fuzzy Control Systems

Yulia Ledeneva^{1,2}, René García Hernández³ and Alexander Gelbukh¹

¹*National Polytechnic Institute, Center for Computing Research*

²*Autonomous University of the State of Mexico*

³*Toluca Institute of Technology, Computer Science Department
Mexico*

1. Introduction

Since the first fuzzy controller was presented by Mamdani in 1974, different studies devoted to the theory of fuzzy control have shown that the area of development of fuzzy control algorithms has been the most active area of research in the field of fuzzy logic in the last years. From 80's, fuzzy logic has performed a vital function in the advance of practical and simple solutions for a great diversity of applications in engineering and science. Due to its great importance in navigation systems, flight control, satellite control, speed control of missiles and so on, the area of fuzzy logic has become an important integral part of industrial and manufacturing processes.

Some fuzzy control applications to industrial processes have produced results superior to its equivalent obtained by classical control systems. The domain of these applications has experienced serious limitations when expanding it to more complex systems, because a complete theory does not yet exist for determining the performance of the systems when there is a change in its parameters or variables.

When some of these applications are designed for more complex systems, the number of fuzzy rules controlling the process is exponentially increased with the number of variables related to the system. For example, if there are n variables and m possible linguistic labels for each variable, m^n fuzzy rules would be needed to construct a complete fuzzy controller. As the number of variables n increases, the rule base quickly overloads the memory of any computing device, causing difficulties in the implementation and application of the fuzzy controller.

Sensory fusion and hierarchical methods are studied in an attempt to reduce the size of the inference engine for large-scale systems. The combination of these methods reduces more considerably the number of rules than these methods separately. However, the adequate fusion-hierarchical parameters should be estimated. In traditional techniques much reliance has to be put on the experience of the system designer in order to find a good set of parameters (Jamshidi, 1997).

Genetic algorithms (GA) are an appropriate technique to find parameters in a large search space. They have shown efficient and reliable results in solving optimization problems. For

these reasons, in this work we present a method that has proved to estimate parameters for the rule base reduction method using GAs.

The chapter is organized as follows. Section 2 summarizes the principles of rule base reduction methods. In Section 3, the sensory-fusion method, the hierarchical method and the combination of these methods are described. Section 4 proposes the GA which allows us to automatically find the parameters in order to improve the complex fuzzy control system performance. Inverted pendulum and beam-and-ball complex control systems are described and results are presented in Section 5. Finally, Section 6 concludes this chapter.

2. Complex Fuzzy Control Systems

A system may be called large-scale or complex, if its order is too high and its model is nonlinear, interconnected with uncertain information flow such that classical techniques of control theory cannot easily handle the system (Jamshidi, 1997). As the complexity of a system increases, it becomes more difficult and eventually impossible to make a precise statement about its behavior. Fuzzy logic is used in system control and analysis design, because it shortens the time for engineering development and sometimes, in the case of highly complex systems, is the only way to solve the problem.

Principle components of a fuzzy controller are: a process of coding numerical values to fuzzy linguistic labels (*fuzzification*), inference engine where the fuzzy rules (expert operator's experience) are implemented and decoding as the output fuzzy decision variables (*defuzzification*). Fuzzy control can be implemented by putting the above three stages on a computer device (chip, personal computer, etc.).

From a control theoretical point of view, fuzzy logic has been intermixed with all the important aspects of systems theory - modeling, identification, analysis, stability, synthesis, filtering, and estimation. One of the first complex system in which fuzzy control has been successfully applied is cement kilns, which began in Denmark. Today, most of the world's cement kilns are using a fuzzy expert system. However, the application of fuzzy control to large-scale complex systems is not, by no means, trouble-free. For such systems the number of the fuzzy IF-THEN rules as the number of sensory variables increases very quickly to an unmanageable level.

When a fuzzy controller is designed for a complex system, often several measurable output and actuating input variables are involved. In addition, each variable is represented by a finite number m of linguistic labels which would indicate that the total number of rules is equal to m^n , where n is the number of system variables. As an example, consider $n = 4$ and $m = 5$ then the total number of fuzzy rules will be $k = m^n = 5^4 = 625$. If there were five variables, then we would have $k = 3125$. From the above simple example, it is clear that the application of fuzzy control to any system of significant size would result in a dimensionality explosion.

3. Rule Base Reduction Methods

One of the most important applications of fuzzy set theory has been in the area of fuzzy rule based system. Rule base reduction is an important issue in fuzzy system design, especially for real time Fuzzy Logic Controller (FLC) design. Rule base size can be easily controlled in most fuzzy modeling and identification techniques.

The size of the rule base of complex fuzzy control systems grows exponentially with the number of input variables. Due to that fact, the reduction of the rule base is a very important issue for the design of this kind of controllers. Several rule base reduction methods have been developed to reduce the rule base size. For instance, fuzzy clustering is considered to be one of the important techniques for automatic generation of fuzzy rules from numerical examples. This algorithm maps data points into a given number of clusters (Klawonn, 2003). The number of cluster centers is the number of rules in the fuzzy system. The rule base size can be easily controlled through the control of the number of cluster centers. However, for control applications, often there is not enough data for a designer to extract a complete rule base for the controller. A designer has to build a generic rule base. A generic rule base includes all possible combinations of fuzzy input values. The size of the rule base grows exponentially as the number of controller input variables grows. As the complexity of a system increases, it becomes more difficult and eventually impossible to make a precise statement about its behavior.

A simple and probably most effective way to reduce the rule base size is to use Sliding Mode Control. The motivation of combining Sliding Mode Control and Fuzzy Logic Control is to reduce the chattering in Sliding Mode Control and enhance robustness in Fuzzy Logic Control. The combination also results in rule base size reduction. However, this approach has its disadvantages as the parameters for the switch function have to be selected by an expert or designed through classical control theory (Hung, 1993).

Anwer (Anwer, 2005) proposed a technique for generation and minimization of the number of such rules in case of limited data sets. Initial rules for each data pairs are generated and conflicting rules are merged on the basis of their degree of soundness. The minimization technique for membership functions differs from other techniques in the sense that two or more membership functions are not merged but replaced by a new membership function whose minimum and maximum ranges are the minimum value of the first and maximum of the last membership function and bisection point of the two or more will be the peak of the new membership function. This technique can be used as an alternative to develop a model when available data may not be sufficient to train the model.

A neuro-fuzzy system (Ajith, 2001; Kasabov, 1998; Juang, 1998; Jang, 1993; Halgamuge, 1994) is a fuzzy system that uses a learning algorithm derived from, or inspired by, neural network theory to determine its parameters (fuzzy sets and fuzzy rules) by processing data samples. Modern neuro-fuzzy systems are usually represented as special multilayer feedforward neural networks (for example, models like ANFIS (Jang, 1993), FuNe (Halgamuge, 1994), Fuzzy RuleNet (Tschichold-German, 1994), GARIC (Berenji, 1992), HyFis (Kim, 1999) or NEFCON (Nauck, 1994) and NEFCLASS (Nauck, 1995)). A disadvantage of these approaches is that the determination of the number of processing nodes, the number of layers, and the interconnections among these nodes and layers are still an art and lack systematic procedures.

Jamshidi (Jamshidi, 1997) proposed to use sensory fusion to reduce a rule base size. Sensor fusion combines several inputs into one single input. The rule base size is reduced since the number of inputs is reduced. Also, Jamshidi (Jamshidi, 1997) proposed to use the combination of hierarchical and sensory fusion methods. The disadvantage of the design of hierarchical and sensory fused fuzzy controllers is that much reliance has to be put on the experience of the system designer to establish the needed parameters. To solve this problem, we automatically estimate the parameters for the hierarchical method using GAs.

3.1 Sensory Fusion Method

This method consists in combining variables before providing them to input of the fuzzy controller (Ledeneva, 2006b). These variables are often fused linearly. For example, we want to fuse two input variables y_1 and y_2 (see Figure 1). The fused variable Y will be calculated as $Y = ay_1 + by_2$. Here, it is considered that the input variables of the fuzzy controller are represented by $m = 5$ linguistic labels. Therefore, in this case, the number of rules will be thus reduced from 25 to 5. As we can observe, more variables has the fuzzy controller, more reduction can be obtained (see Figure 4).

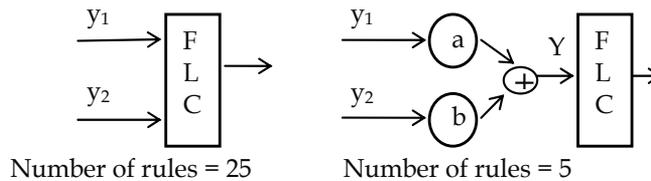


Fig. 1. Rule base reduction of sensory fusion fuzzy controller (when two variables are fused).

As another example, consider that a fuzzy controller has three inputs variables y_1 , y_2 and y_3 . The total number of rules will be 125. In this case, we look into combining three variables in one of these four possible ways:

1. Variables y_1 and y_2 are fused in the new variables Y_1 and Y_2 :

$$\begin{aligned} Y_1 &= ay_1 + by_2 \\ Y_2 &= y_3 \end{aligned}$$

2. Variables y_1 and y_3 are fused in the new variables Y_1 and Y_2 :

$$\begin{aligned} Y_1 &= ay_1 + by_3 \\ Y_2 &= y_2 \end{aligned}$$

3. Variables y_2 and y_3 are fused in the new variables Y_1 and Y_2 :

$$\begin{aligned} Y_1 &= ay_2 + by_3 \\ Y_2 &= y_1 \end{aligned}$$

4. Variables y_1 , y_2 and y_3 are fused in the new variable Y :

$$Y = ay_1 + by_2 + cy_3$$

The number of rules will be thus reduced by 125 to 25 if two variables are fused or from 125 to 5 if the three variables are combined.

The reduction of the number of rules is optimal if one can fuse all the input variables in only one variable associated. In this case, the number of rules is equal to the definite number of linguistic labels for this variable. But it is obvious that all these variables cannot be fused arbitrarily, any combination of variables has to be reasoned and explained. In practice only

two variables are fused: generally the error and the change of error. The fusion can be done through the following rule

$$E = ae + b\Delta e \quad (1)$$

where e and Δe are error and its rate of change, E is the fused variable, and a and b found manually (Jamshidi, 1997).

We want to point out that the manually selection of the parameters a and b convert into fastidious routine. Below, we describe a new method (Ledeneva, 2006a), which permits to estimate these parameters automatically.

3.2 Hierarchical Method

In the hierarchical fuzzy control structure from (Ledeneva, 2007a), the first-level rules are those related to the most important variables and are gathered to form the first-level hierarchy. The second most important variables, along with the outputs of the first-level, are chosen as inputs to the second level hierarchy, and so on. Figure 2 shows this hierarchical rule structure.

$$\begin{aligned} &\text{IF } y_1 \text{ is } A_{1i} \text{ and } \dots \text{ and } y_1 \text{ is } A_{1i} \text{ THEN } u_1 \text{ is } B_i \\ &\text{IF } y_2 \text{ is } A_{2i} \text{ and } \dots \text{ and } y_2 \text{ is } A_{2i} \text{ THEN } u_2 \text{ is } B_i \\ &\dots \\ &\text{IF } y_{N_i+1} \text{ is } A_{N_i+1} \text{ and } \dots \text{ and } y_{N_i+n_j} \text{ is } A_{N_i+n_j} \text{ THEN } u_i \text{ is } B_i \end{aligned} \quad (2)$$

where $i, j = 1, \dots, n$; y_i are output variables of the system, u_i are control variables of the system, A_{ij} and B_i are linguistic labels; $N_i = \sum_{j=1}^{i-1} n_j \leq n$ and n_j is the number of j -th level system variables used as inputs.

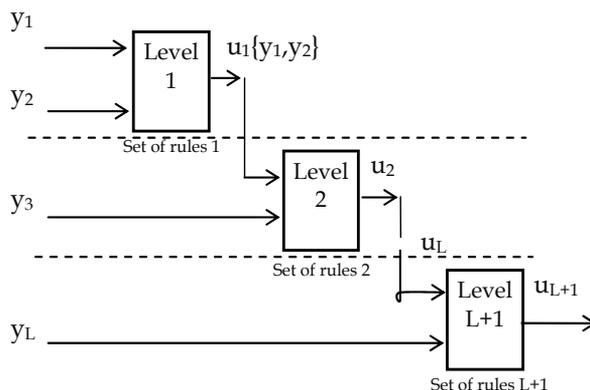


Fig. 2. Schematic representation of a hierarchical fuzzy controller.

The goal of this hierarchical structure is minimize the number of fuzzy rules from exponential to linear function. Such rule base reduction implies that each system variable

provides one parameter to the hierarchical scheme. Currently, the selection of such parameters is manually done.

3.3 Combination of Methods

The more number of input variables of the fuzzy controller we have, the more it is interesting to combine the methods presented above with a goal to reduce more number of rules. We want to quote, as an example, the combination of the sensory fusion method (section 3.1) and the hierarchical method (section 3.2) for five variables as in Figure 3. Initially, the variables are fused linearly, as in Figure 1, and then are organized hierarchically according to a structure similar to that of Figure 2.

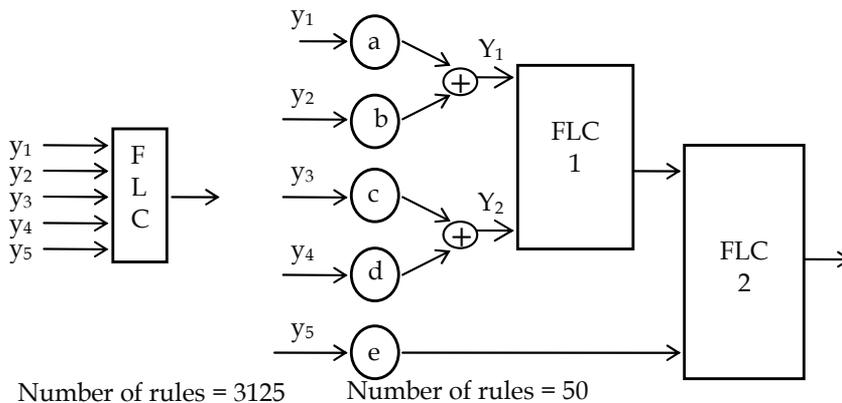


Fig. 3. Rule base reduction for the combination of sensory fusion and hierarchical methods (for $n = 5$ and $m = 5$).

The number of rules and the comparison of the sensory fusion method, the hierarchical method and the combination of these rule base reduction methods are presented in Table 1 and Figure 4 correspondingly. Take into account that the variables are fused here per pair and that on each level of the hierarchy one and only one variable is added. The most significant reduction can be obtained when the sensory fusion and hierarchical methods are combined (Ledeneva, 2007b).

Method used to reduce the number of rules	The number of variables $n > 1$	
	Even	Odd
Sensory Fusion	$m^{n/2}$	$m^{(n+1)/2}$
Hierarchical	$(n-1) \cdot m^2$	
Combination of methods	$((n/2)-1) \cdot m^2$	$((n+1)/2)-1$

Table 1. - The number of rules for the different reduction methods.

4. Genetic Optimization of the Parameters

Firstly, we give some basic definitions of GAs, than we present the proposed method to estimate the parameters of the sensory fusion method, the hierarchical method, and the combination of these rule base reduction methods.

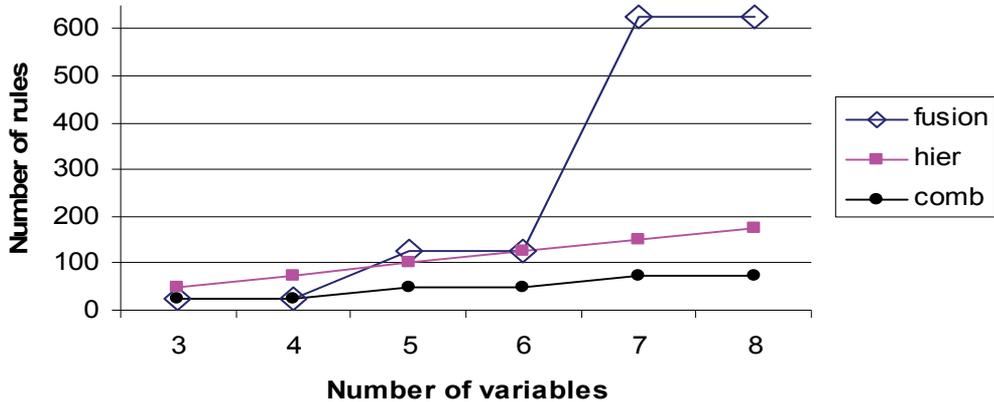


Fig. 4. Comparison of various rule base reduction methods with $m = 5$.

4.1 Step Response Characteristics

A fuzzy control system can be evaluated with the step response characteristics. We consider the following step response characteristics (see Figure 5):

Overshoot (%) is the amount by which the response signal can exceed the final value. This amount is specified as a percentage of the range of steps. The range of steps is the difference between the final value and initial values.

Undershoot (%) is the amount by which the response signal can undershoot the initial value. This amount is specified as a percentage of the range of steps. The range of steps is the difference between the final value and initial values.

Settling time is time taken until the response signal settles within a specified region around the final value. This settling region is defined as the step value plus or minus the specified percentage of the final value.

Settling (%) is the percentage used in the settling time.

Rising time is time taken for the response signal to reach a specified percentage of the range of steps. The range of steps is the difference between the final value and initial value.

Rise (%) is the percentage used in the rising time.

4.2 Genetic Algorithms

GA uses the principles of evolution, natural selection, and genetics from natural biological systems in a computer algorithm to simulate evolution (Goldberg, 1989). Essentially, the genetic algorithm is an optimization technique that performs a parallel, stochastic, but directed search to evolve the fittest population. GAs encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to

these structures so as to preserve critical information. GAs are often viewed as function optimizers, although the range of problems to which genetic algorithms have been applied is quite broad. The more common applications of GAs are the solution of optimization problems, where efficient and reliable results have been shown. That is the reason why we will use these algorithms to find parameters for the rule base reduction methods.

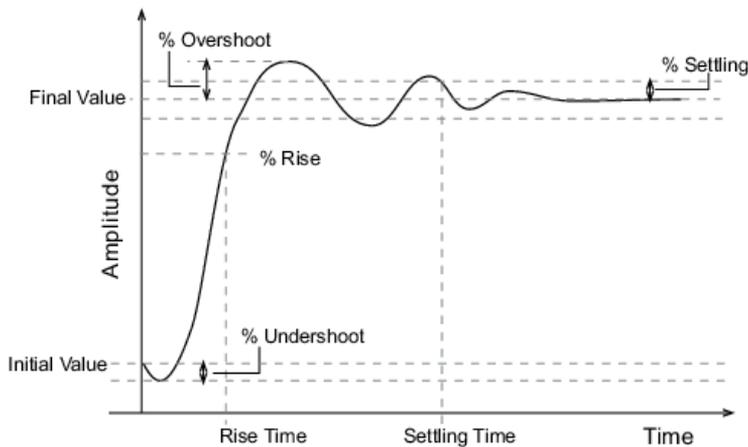


Fig. 5. Step response characteristics.

In the early 1970s, John Holland introduced the concept of genetic algorithms. His aim was to make computers do what nature does. Holland was concerned with algorithms that manipulate strings of binary digits. Each artificial “chromosome” consists of a number of “genes” and each gene is represented by 0 or 1:

0	0	0	0	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---

Nature has an ability to adapt and learn without being told what to do. In other words, nature finds good chromosomes blindly. GAs do the same. Two mechanisms link a GA to the problem it is solving: encoding and evaluation. The GA uses a measure of fitness of individual chromosomes to carry out reproduction. As reproduction takes place, the crossover operator exchanges parts of two single chromosomes, and the mutation operator changes the gene value in some randomly chosen location of the chromosome.

4.2 Method for the Estimation of Parameters

The scheme of the proposed method is shown in Figure 5. We have three modules: System Module, Fuzzy Controller Module, and Genetic Algorithm Module. These three modules interconnect in two loops: an internal loop to control a system and an external loop to modify the fusion-hierarchical parameters. The internal loop comprises the fuzzy controller module and the system module. In other words, this loop represents a closed-loop control scheme. The external loop is composed of the genetic algorithm module, the fuzzy controller module, and the system module. The objective of the genetic algorithm module is to

estimate the fusion-hierarchical parameters of the fuzzy controller through the minimization of the error between the design specifications and the output of the process. Below we discuss each module of the proposed method.

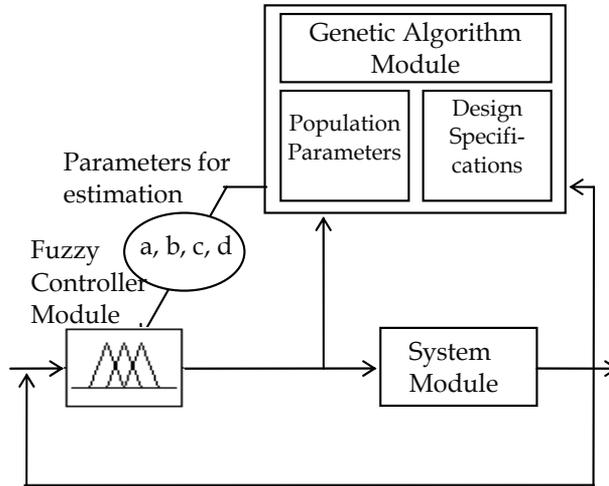


Fig. 5. Scheme of the proposed method.

4.2.1 Control System Module

The control system is defined as a complex system with p inputs and q outputs:

$$\begin{aligned} u &= [u_1, \dots, u_p] \\ y &= [y_1, \dots, y_q] \end{aligned} \quad (3)$$

4.2.2 Fuzzy Controller Module

The fuzzy controller module is represented by the fuzzy controller of reduced complexity which results after the application of the sensory fusion method, the hierarchical method, and the combination of these rule base reduction methods correspondingly such that it uses the combination of the fusion-hierarchical parameters.

Generally, the fuzzy controller is composed of one or several fuzzy controllers (depending on the number of variables). These controllers are of the Takagi-Sugeno type and each has a two inputs. The variation of these inputs results from the design of the sensory fusion method, the hierarchical method, and the combination of these methods; or the output variables of another fuzzy controller.

For example, let us describe general fuzzy controller with two input variables (see Figure 6) which are the vector of error $\varepsilon = y_d - y$ and variation of error $\Delta\varepsilon$, where y_d is the desirable system output. $K\varepsilon = [K\varepsilon_1, \dots, \varepsilon_q]$ and $K\Delta\varepsilon = [K\Delta\varepsilon_1, \dots, \Delta\varepsilon_q]$ are the gain input vectors. The output gain vector is noted as $Ku = [Ku_1, \dots, Ku_q]$. The vector containing the resulting variables from the fusion module is noted as $X = [x_1, \dots, x_q]$. So, for this example we have the output

$$X_i = K\varepsilon_i \cdot \varepsilon_i + K\Delta\varepsilon_i \cdot \Delta\varepsilon_i \quad (4)$$

where $i = 1, \dots, q$.

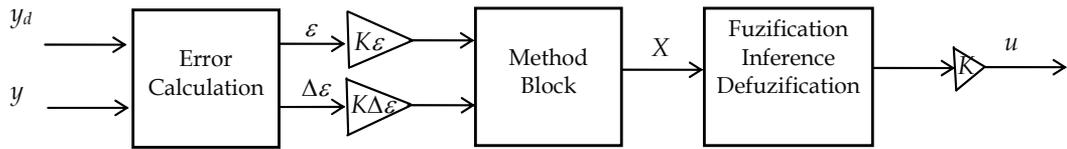


Fig. 6. General fuzzy controller structure.

4.2.3 Genetic Algorithm Module

Genetic Algorithm Module represents a genetic algorithm that maintains a population of chromosomes where each chromosome represents a combination of candidate parameters. This genetic algorithm uses data from the system to evaluate the fitness of each parameter in the population. The evaluation is done at each time step by simulating out with each combination of the parameters and forming a fitness function based on the design specifications which characterize the desired performance of the system. Using this fitness evaluation, the genetic algorithm propagates parameters into the next generation via the combination of the genetic operations proposed below. The combination of the parameters that is the fittest one in the population is used in the sensory fusion fuzzy controller.

This allows the proposed method to evolve automatically the combination of parameters from generation to generation (i.e., from one time step to the next, but of course multiple generations could occur between time steps), and hence to tune the combination of the parameters in response to changes in the system or due to user changes of the specifications in the fitness function of the GA.

The proposed procedure of estimating the combination of parameters by GA is summarized as follows:

1. Determine the rule base reduction method and the number of parameters it is necessary to find.
2. Construct an initial population.
3. Encode each chromosome in the population.
4. Evaluate the fitness value for each chromosome.
5. Reproduce chromosomes according to the fitness value calculated in Step 4.
6. Create offspring and replace parent chromosomes by the offspring through crossover and mutation.
7. Go to 3 until the maximum number of iterations is reached.

4.2.3.1 Representation

To encode the combination of parameters, chromosomes of length $N \cdot B$ are used, where N is the number of parameters and B the number of bits which we use to encode the parameters. To decide how many bits to use for each parameter, we should consider the range of all possible values for each of them. For example, suppose that the parameters we want to obtain are positive with one decimal after the dot. To encode all possible values of each parameter we will use 8 bits. In Figure 7, there is one chromosome, representing the combination of parameters, which has $N = 4$ parameters with $B = 8$ bits each. So, the total range of the parameters will be in the interval $[0, 256]$. To obtain the required precision (one

decimal after the dot), we multiply the output values of the parameters by 0.1. As a result, the searching parameters will be in the interval $[0, 25.6]$.

4.2.3.2 Population

The initial population is randomly generated. Its size is fixed and equal to 50 individuals.

N		B							
a	= 1.5	0	0	0	0	1	1	1	1
b	= 4.7	0	0	1	0	1	1	1	1
c	= 20.3	1	1	0	0	1	0	1	1
d	= 3	0	0	0	1	1	1	1	0

Fig. 7. Example of representation of one chromosome (or one combination of parameters) which has $N = 4$ parameters with $B = 8$ bits each.

4.2.3.3 Fitness Function

The genetic algorithm maintains a population of chromosomes. Each chromosome represents a different combination of parameters. It also uses a fitness measure that characterizes the closed-loop specifications. Suppose, for instance, that the closed-loop specifications indicate that the user want, for a step input, a (stable) response with a rise-time of t_r^* , a percent overshoot of s_p^* , and a settling time of t_s^* . We propose the fitness function so that it measures how close each individual in the population at time t (i.e., each parameter candidate) is to meet these specifications. Suppose that t_r , s_p , and t_s denote the rise-time, the overshoot, and the settling time, respectively, for a given chromosome (we compute them for a chromosome in the population by performing a simulation of the closed-loop system with the candidate combination of the parameters and a model of the system). Given these values, we propose (for each chromosome and every time step)

$$J = w_1 (t_r - t_r^*)^2 + w_2 (s_p - s_p^*)^2 + w_3 (t_s - t_s^*)^2 \quad (4)$$

where $w_i > 0$, $i = 1, 2, 3$, are positive weighting factors. The function J characterizes how well the candidate combination of the parameters meets the closed-loop specifications; if $J = 0$ it meets the specifications perfectly. The weighting factors can be used to prioritize the importance of meeting the different specifications (e.g., a high value of w_2 relative to the other values indicates that the percent overshoot specification is more important to meet than the others).

Now, we would like to minimize J , but the genetic algorithm is a maximization routine. To minimize J with the genetic algorithm, we propose the fitness function

$$J_{\text{res}} = 1/J \quad (5)$$

Then, after knowing the design specifications of the system, and once we can obtain the step response characteristics for each chromosome in the population (rise-time, overshoot, and settling time), the fitness function is calculated in 2 steps:

1. We ask if the results coming from the GA is in the range of the design specifications of the system. If they are, we go to step 2. Else, the fitness value of this chromosome is set to 1000.
2. The fitness function is defined as described above (equations 4, 5).

4.2.3.4 Genetic Operators

In this section, we determine some genetic operators that we will use below (in Table 4).

Crossover: is a genetic operator that combines two chromosomes (parents) to produce one or two chromosomes (offspring). The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. First, the crossover operator randomly chooses a crossover point where two parent chromosomes “break”, and then exchanges the chromosome parts after that point with a user-definable crossover probability. As a result, two new offspring are created (Melanie, 1999). The most common forms of crossover are one-point and two-point.

Mutation: represents a change in the gene. Its role is to provide and guarantee that the search algorithm is not trapped on a local optimum. The mutation operator uses a mutation probability denoted as p_m previously set by the user, which is quite small in nature, and it is kept low for GAs, typically in the range 0.001 and 0.01. According with this probability, the bit value is changed from 0 to 1 or vice versa (Melanie, 1999).

Elitism: copies the best individual (% of most fit individual) from the actual population to a new population and the rest of the new population is constructed according to the genetic algorithm.

Half Uniform Crossover (HUX): In this operator, bits are randomly and independently exchanged, but exactly half of the bits that differ between parents are swapped (see Figure 8). The HUX operator (Eshelman, 1991 ; Gwiazda, 2006) ensures that the offspring are equidistant between the two parents. This serves as a diversity preserving mechanism.

Truncation selection: implies that duplicate individuals are removed from population (Melanie, 1999).

In **roulette selection:** parents are selected according to their fitness. The better is the fitness, the bigger chance to be selected.

Parent A	1	1	0	1	1	0	1	1
Parent B	0	0	1	1	1	0	1	0
* Different Allels	*	*	*	1	1	0	1	*
x Allels to Interchange	x	*	x	1	1	0	1	*
Offspring A	0	1	1	1	1	0	1	1
Offspring B	1	0	0	1	1	0	1	0

Fig. 8. Example of Half Uniform Crossover.

5. Simulation Results

5.1 Inverted Pendulum System

The inverted pendulum control system (Messner, 1998; Aguilar, 2005; Aguilar, 2007) is used to test the proposed methods. The objective of this control system is, on one hand, to maintain the stem of the pendulum in high driving position, on the other hand, to bring the cart towards a given position x_0 . The scheme in Figure 9 shows the main components of the system.

The basic variables are:

- the angular position of the stem θ ;
- the angular velocity of the stem $\Delta\theta$;
- the horizontal position of cart x ;
- the velocity of the cart Δx .

The design specifications of the inverted pendulum system are:

- the objective position of the cart is 30 cm;
- the overshoot of no more than 5 %;
- the settling time of no more than 5 sec.

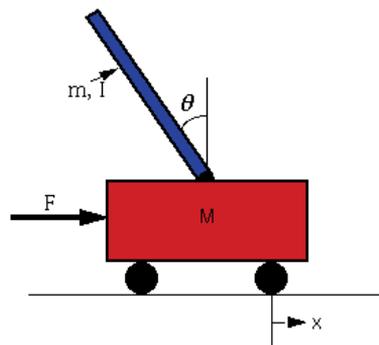


Fig. 9. Inverted pendulum, where $M = 1$ kg – mass of the cart, $m = 0.1$ kg – mass of the pendulum, $l = 1$ kg – length to pendulum, F – force applied to the cart, x – cart position coordinate, θ – pendulum angle with vertical.

5.1.1 Design of the Sensory Fusion Method

The design of sensory fusion on a fuzzy controller is described in this section. First the sensory fusion of the input variables is done as follows:

$$\begin{cases} X_{\theta} = a\theta + b\Delta\theta \\ X_e = ce + d\Delta e \end{cases} \quad (6)$$

where a , b , c , and d are positive.

So, if X_e is null, that means that the cart reached its position of reference ($e = 0$ and $\Delta e = 0$), or that it moves towards this one ($ce = -d\Delta e$). Reasonably it is identical for X_{θ} . If X_{θ} is null, the

angular position of the pendulum is stabilized to zero. Consequently, the stabilization of X_θ and X_e makes it possible to bring the pendulum towards a position of reference and ensure the maintenance of the stem of the pendulum in high driving position. The more absolute value of X_θ more the horizontal position of the pendulum is critical. And the more absolute value of X_e , more the angular position of the pendulum is critical. The variables X_θ and X_e represent respectively the critical angular position and the critical horizontal position of the pendulum.

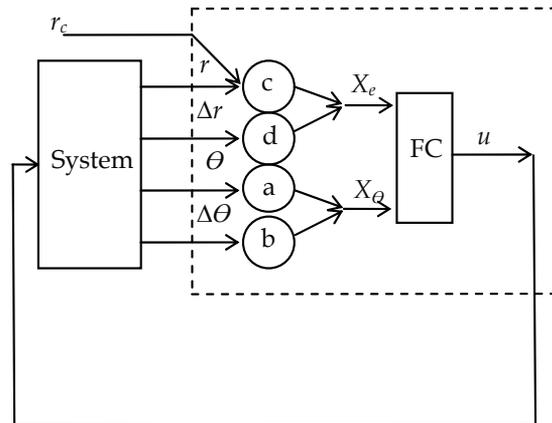


Fig. 10. Scheme of the sensory fusion fuzzy controller.

This control problem is far from being commonplace because two variables are to be controlled but only one has an action of control. The dynamics of θ being much faster than that of r , the adopted strategy is as follows: we initially seek to balance the pendulum (high driving position) then to gradually bring it towards its position of reference by unbalancing it on the "good side".

We can then write the five following rules in order to control the pendulum at the same time horizontally and vertically:

- R1: IF X_θ is Negative THEN u is Negative,
 - R2: IF X_θ is Positive THEN u is Positive,
 - R3: IF X_θ is Zero y X_e is Negative THEN u is Negative,
 - R4: IF X_θ is Zero y X_e is Zero THEN u is Zero,
 - R5: IF X_θ is Zero y X_e is Positive THEN u is Positive,
- (7)

These five rules interpret well the priority objective which is the vertical stabilization of the pendulum: X_e is considered only when X_θ is null. Now let us examine the third rule (R3): "IF X_θ is Zero and X_e is Negative THEN u is Negative". This negative control involves X_θ positive. The second rule R2 is then activated: $u_s = ku$ becomes positive in order to balance the pendulum and the position of the cart increases as wished. The reasoning is similar for the fifth rule.

The five rules (7) can be written in a more compact way in the form of table (Table 2):

X_θ		N	Z	P
Xe	N	N	N	P
	Z		Z	
	P		P	

Table 2. Rule base of the sensory fusion fuzzy controller.

5.1.2 Design of the Hierarchical Method

The design of the hierarchical fuzzy controller is described in this section. The structure of the hierarchical fuzzy controller is represented in Figure 11.

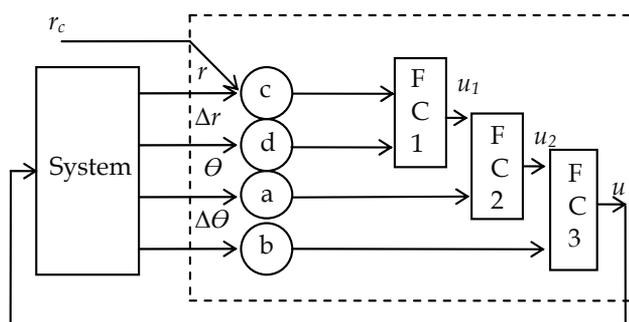


Fig. 11. - Hierarchical fuzzy controller.

The hierarchical fuzzy controller is composed of three fuzzy controller series connected. The corresponding rule bases are represented in Tables 3, 4 and 5. The total number of rules is $9 + 5 + 9 = 23$ rules.

The objective of the first fuzzy controller (FC1) is to bring the cart towards its position of reference r_c . The first action u_1 consists in unbalancing the pendulum in the "good direction". This imbalance must have as a consequence the displacement of the cart in the desired direction.

$d\Delta e$		N	Z	P
ce	N	N	N	Z
	Z	N	Z	P
	P	Z	P	P

Table 3. Rule base of the FC1.

The objective of second fuzzy controller (FC2) is to balance the pendulum if this one is not so yet. The first decision of an action u_1 is preserved if the angular position of the pendulum is zero, but if the pendulum is not balanced, the new action u_2 is such as the pendulum converges towards a high driving position.

As for the third fuzzy controller (FC3) it aims to refine the preceding control by considering an additional variable $\Delta\theta$, the angular velocity of the pendulum.

If $\Delta\theta$ is zero, it does not have a reason there to modify the preceding control u_2 . In the same way, if $\Delta\theta$ is negative (respectively positive) and u_2 is negative (respectively positive) then

the preceding control does not have to be revised since it balances the pendulum. On the other hand, if the control u_2 is zero and the pendulum tends to be unbalanced then it is necessary to choose a control consequently.

5.1.3 Design of the Fusion-Hierarchical Method

The objective position where we must to bring a cart is x_o . The variables to fuse are θ and $\Delta\theta$, e and Δe , where e is the error in position given by $e = x - x_o$ and $\Delta e = \Delta x$. The sensory fusion of the error in position and its variation $X_e = ce + d\Delta e$ combined with the hierarchical method led to the fuzzy controller represented in Figure 12. The first fuzzy controller (FC1) calculates the first control action according to X_e and the angular position θ . In the second fuzzy controller (FC2), it refines the value of preceding control by considering an additional variable $\Delta\theta$. The fuzzy controller based on fusion-hierarchical combination is represented in the Figure 12. The rule bases of FC1 and FC2 are represented in Tables 6-7.

$a\theta$		N	Z	P
u_1	N	N	N	Z
	Z		Z	
	P		P	

Table 4. Rule base of the FC2.

$d\Delta\theta$		N	Z	P
u_2	N	N	N	Z
	Z	N	Z	P
	P	Z	P	P

Table 5. Rule base of the FC3.

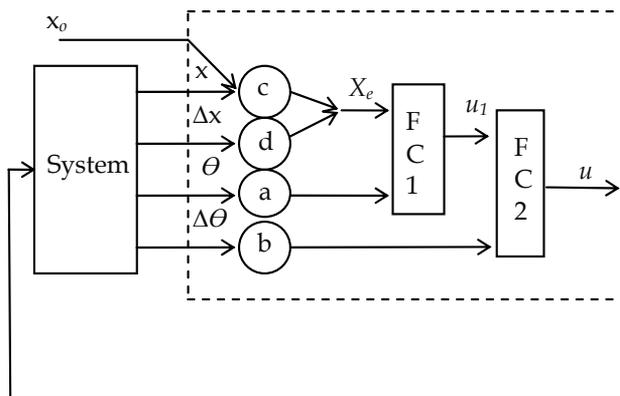


Fig. 12. Fuzzy controller based on the combination of the sensory fusion and hierarchical methods.

$a\theta$		N	Z	P
Xe	N	N	N	P
	Z		Z	
	P		P	

Table 6. Rule bases of the fuzzy controllers FC1.

$b\theta$		N	Z	P
u_1	N	N	N	Z
	Z	N	Z	P
	P	Z	P	P

Table 7. Rule bases of the fuzzy controllers FC2.

The simulation of the inverted pendulum is performed in *Simulink*, *Matlab* (Figure 13) starting from the nonlinear equations (Messner, 1998). The fuzzy controller is implemented in *Matlab's FIS Editor*. The input fuzzy sets are represented by triangular functions (N, Z and P) regularly distributed on the universe of discourse $[-1, 1]$. The output fuzzy sets are singletons regularly distributed on $[-1, 1]$.

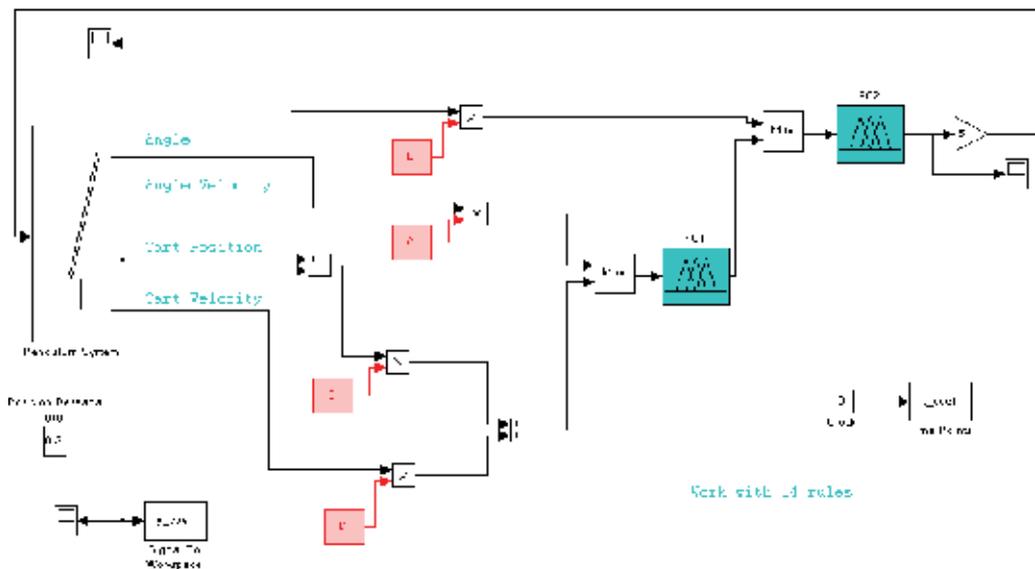


Fig. 13. Inverted pendulum control problem for the combination of methods implemented in Simulink.

5.1.4 Results

We apply the proposed method in order to find the parameters a , b , c , and d . The experiments were realized with the combination of some genetic operators in Table 8. The results of obtained parameters for each combination of genetic operators are presented in Tables 9-11. The best result is highlighted (Tables 9-11). The time response graphics are

illustrated for the best experiment from Tables 8-10 in Table 12. In these experiment, the weighting factors of overshoot, settling time and rising time are $w_1 = 1$, $w_2 = 1$, and $w_3 = 0$ respectively.

Num.	Selection Operator	Num. of Generations	Crossover Operator	Mutation Operator	Elitism
1.	Roulette	50	Two-point with $p_c=0.8$	$p_m=0.01$	6 %
2.	Roulette	100	Two-point with $p_c=0.8$	$p_m=0.01$	3 %
3.	Roulette	50	Two-point with $p_c=0.8$	$p_m=0.15$	3 %
4.	Truncation	30	HUX	-	-
5.	Truncation	50	HUX	-	-
6.	Truncation	100	HUX	-	-

Table 8. The combinations of genetic operators for realize the experiments.

For the reduction with the sensory fusion method we obtained the following parameters: $a = 23$, $b = 8$, $c = 1.3$ and $d = 2.8$ (see experiment 6, Table 9). With these parameters the horizontal position of the cart is stabilized in 4.95 seconds with overshoot of 0%. The design specifications of the inverted pendulum system are totally satisfied.

Num.	a	b	c	d	Overshoot (%)	Settling Time (sec.)	Rising Time (sec.)
1.	13.1	4.3	3	2.6	26	4.7	0.9
2.	19.7	4.1	1.3	2.6	0.17	4.95	2.85
3.	25	7	4	6	0	4.75	2.1
4.	24.8	5.5	6	7	0	4	1.7
5.	20.5	9.6	6.3	8.5	0	4.5	2.6
6.	23	8	1.3	2.8	0	4.95	2.7

Table 9. The results obtained for the sensory fusion fuzzy controller.

For the reduction with hierarchical method we obtained the following parameters: $a = 19.2$, $b = 6.4$, $c = 1.1$ and $d = 2.3$ (see experiment 6, in Table 10). With these parameters the horizontal position of the cart is stabilized in 4.7 seconds with overshoot 0%.

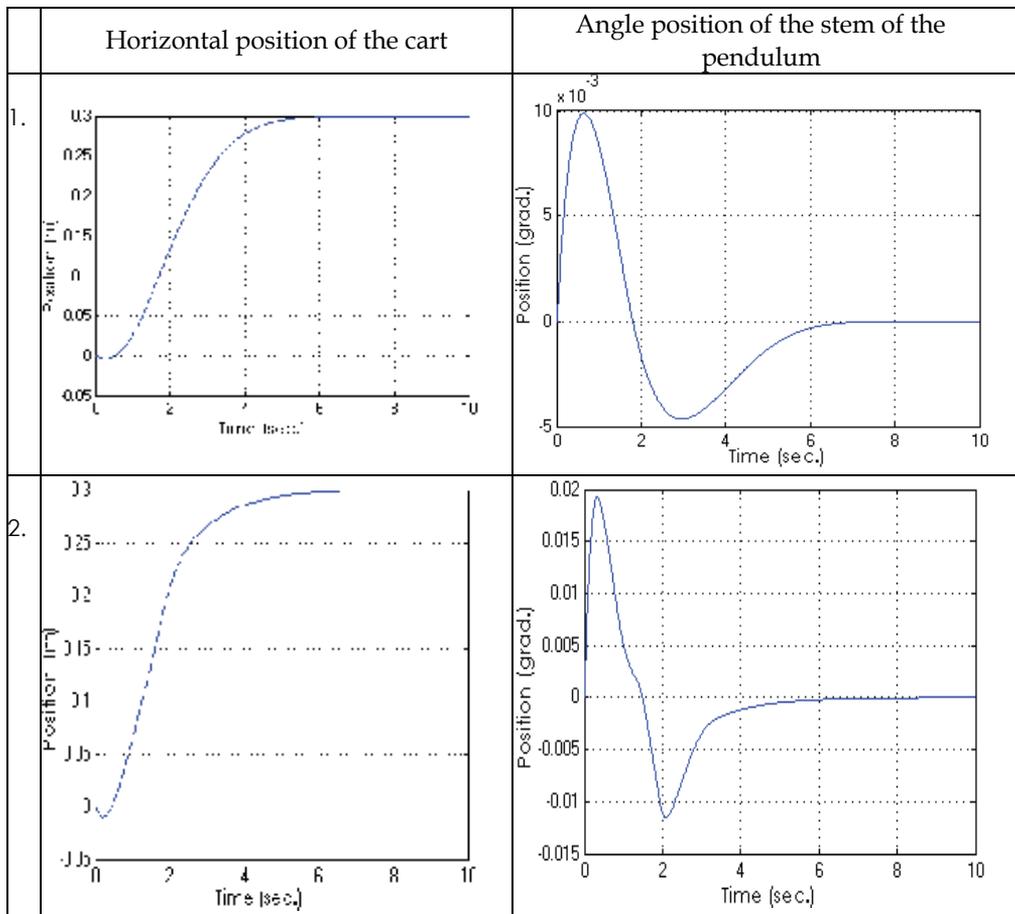
Num.	a	b	c	d	Overshoot (%)	Settling Time (sec.)	Rising Time (sec.)
1.	20.6	11.2	3.3	5.5	0	5	1.7
2.	24.5	8.3	5.4	6.9	0	4	1.8
3.	17	6	4	5	0	4	2
4.	19.6	9.4	1	2.3	0	5.16	2.7
5.	23.5	7.1	5.2	7.5	0	5	2.4
6.	19.2	6.4	1.1	2.3	0	4.7	2.7

Table 10. The results obtained for the hierarchical fuzzy controller.

For the reduction with the combination of the sensory fusion and the hierarchical methods we obtained the following parameters: $a = 25.3$, $b = 10.1$, $c = 3.4$, and $d = 5.5$. With these parameters the horizontal position of the cart is stabilized in 5 seconds with overshoot 0% (see experiment 6 in Table 11), and the behavior of the angle position of the stem of pendulum is shown in Table 12, experiment 6, the third column.

Num.	a	b	c	d	Overshoot (%)	Settling Time (sec.)	Rising Time (sec.)
1.	20.9	7.6	1.9	2.9	5.4	5	1.8
2.	14.7	5.1	2.7	3.1	3.5	3	1.2
3.	19.2	6.5	5.7	6.4	0	4	1.5
4.	24.7	7.2	3.1	4.8	0	4	2
5.	19.6	7	1.1	2.3	0	4.5	2.6
6.	25.3	10.1	3.4	5.5	0	5	2

Table 11. The results obtained for the fusion-hierarchical fuzzy controller.



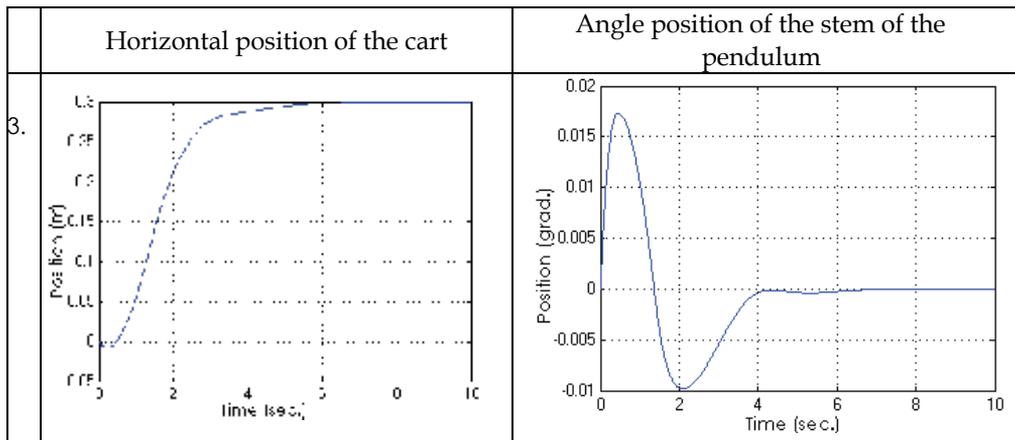


Table 12. The time response graphics obtained for the fusion-hierarchical fuzzy controller.

The fitness value convergence diagrams calculated for 30 generations are presented in Figure 14. The main observation is that the best combination of parameters can be met around 25 generations.

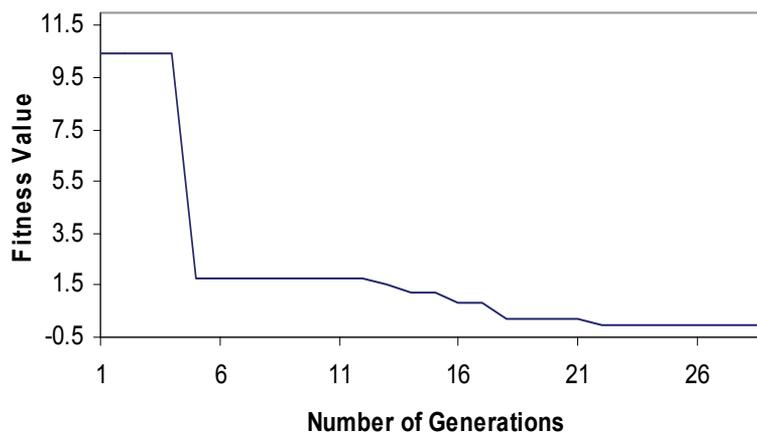


Fig. 14. The fitness value convergence diagram (30 generations).

5.2 Beam-and-Ball System

A representation of the beam-and-ball system is given in Figure 15 (Messner, 1998). A ball of mass M placed on a beam of length L is allowed to roll along the length of the beam. A lever arm of negligible mass mounted onto a gear and driven by a servomotor is used to tilt the beam in either direction. The beam angle α is controlled by a rotational motion of the servomotor, shown as θ . With α initially zero, the ball is in a stationary position. When α is positive (in relation to the horizontal) the ball moves to the left due to gravity, and when α is negative the ball moves to the right. The objective is to design a controller for this system so that the ball position can be controlled to any position r along the beam.

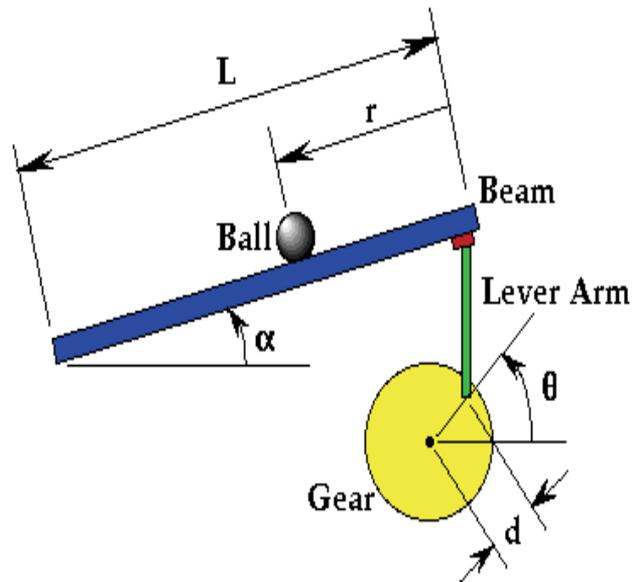


Fig. 15. Beam-and-ball system.

For this problem, we will assume that the ball rolls without slipping and friction between the beam and ball is negligible. The constants and variables for this example are defined as follows: $M = 1\text{ kg}$ - mass of the ball, $r = 0.015\text{ m}$ - radius of the ball, $d = 0.03\text{ m}$ - lever arm offset, g - gravitation acceleration, $L = 1.0\text{ m}$ - length of the beam, J - inertia moment of ball, α - beam angle coordinate, θ - pendulum angle with vertical.

The design specifications of this problem are:

- Carry the ball to the position of 50 cm;
- Overshoot no more than 5%;
- The settling time no more than 3 seconds.

The basic variables of the ball-and-beam system are:

- Angular position θ ,
- Angular velocity $\Delta \theta$,
- Horizontal position of the ball r ,
- Velocity of the ball Δr .

The equation of motion for the ball is given by the following (Messner, 1998):

$$\left(\frac{J}{R^2} + m\right)r'' + mg \sin \alpha - mr(\alpha')^2 = 0 \quad (6)$$

Linearization of this equation about the beam angle, $\alpha = 0$, gives us the following linear approximation of the system:

$$\left(\frac{J}{R^2} + m\right)r'' = -mg\alpha \quad (7)$$

The equation which relates the beam angle to the angle of the gear can be approximated as linear by the equation below:

$$\alpha = \frac{d}{L}\theta \quad (8)$$

Substituting this in the previous equation, we get:

$$\left(\frac{J}{R^2} + m\right)r'' = -mg\frac{d}{L}\alpha \quad (9)$$

The linearized system equations can also be represented in state-space form as shown below:

$$\begin{bmatrix} r' \\ r'' \\ \alpha' \\ \alpha'' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-mg}{\left(\frac{J}{R^2} + m\right)} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ r' \\ \alpha \\ \alpha' \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u; \quad y = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ r' \\ \alpha \\ \alpha' \end{bmatrix} \quad (10)$$

For this system the gear and lever arm would not be used, instead a motor at the center of the beam will apply torque to the beam, to control the position of the ball.

5.2.1 Design of the Sensory Fusion Method

The sensory fusion fuzzy controller is designed as show in Figure 16. The sensory fusion of the input variables is done as follows:

$$\begin{cases} X_\theta = a\theta + b\Delta\theta \\ X_e = cr + d\Delta r \end{cases} \quad (11)$$

where a, b, c, and d are positive.

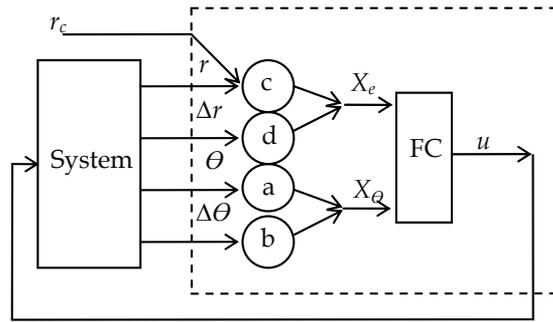


Fig. 16. Design of the sensory fusion fuzzy controller for beam-and-ball system.

We obtain the nine following rules in order to control the ball to any position r_c . These nine rules interpret well the priority objectives which are the horizontal stabilization to the desired position of the ball and the horizontal stabilization of the beam (see Table 13).

		X_θ		N	Z	P
X_e	N		P	P	Z	Z
	Z		N	Z	Z	P
	P		Z	N	N	N

Table 13. Rule base for the sensory fuzzy controller.

5.2.2 Design of the Hierarchical Method

The design of a hierarchical fuzzy controller is represented in Figure 17.

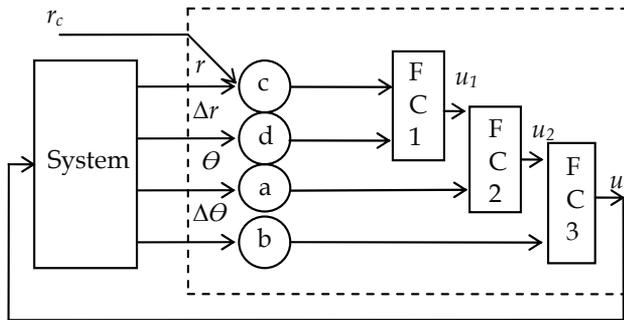


Fig. 17. Hierarchical fuzzy controller.

The hierarchical fuzzy controller is composed of three connected fuzzy controller. The corresponding rules are represented in Tables 14, 15, and 16. The total number of rules is $9 + 9 + 9 = 27$ rules.

The objective of the first fuzzy controller (FC1) is to bring the ball towards its position of reference r_c . The first action u_1 consists in unbalancing the beam in the right direction. This imbalance must have as a consequence the displacement of the ball in the desired direction.

		$d\Delta e$	N	Z	P
ce	N	N	N	Z	
	Z	N	Z	P	
	P	Z	P	P	

Table 14. Rule base for FC1.

The objective of second fuzzy controller (FC2) is to balance the beam if it is not balanced. The new action u_2 is such as the beam moves towards horizontal position.

		$a\theta$	N	Z	P
u_1	N	P	P	Z	
	Z	N	Z	P	
	P	Z	N	N	

Table 15. Rule base for FC2.

The third fuzzy controller (FC3) aims to refine the preceding control by considering an additional variable: $\Delta\theta$ - the angular velocity of the beam.

		$d\Delta\theta$	N	Z	P
u_2	N	P	P	Z	
	Z	N	Z	P	
	P	Z	N	N	

Table 16. Rule base for FC3.

5.2.3 Design of the Fusion-Hierarchical Method

The sensory fusion method combined with the hierarchical method led to the fuzzy controller represented in Figure 18. The first fuzzy controller FC1 calculates the first control according to X_e and the angular position θ . The corresponding rule base is similar to that written for the fuzzy controller based on the sensory fusion only. In the second fuzzy controller FC2, it refines the value of preceding control by considering the additional variable $\Delta\theta$. Its rule base is obtained following the same reasoning as that which guided the writing of the third rule base for the fuzzy controller based on the hierarchical method.

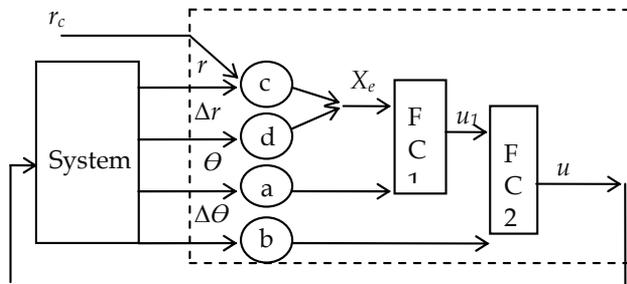


Fig. 18. FC based on the fusion-hierarchical combination.

The rule bases of two fuzzy controllers FC1 and FC2 are represented in Tables 17-18.

$a\theta$		N	Z	P
X_e	N	P	P	Z
	Z	N	Z	P
	P	Z	N	N

Table 17. Rule bases of the fuzzy controllers FC1.

$b\Delta\theta$		N	Z	P
u_1	N	P	P	Z
	Z	N	Z	P
	P	Z	N	N

Table 18. Rule bases of the fuzzy controllers FC2.

5.2.4 Results

Now we apply the proposed method in order to find the parameters a , b , c , and d for the beam-and-ball system. The experiments were realized with the combination of some genetic operators (Table 8). In all experiments the population size is 50 chromosomes. The results of obtained parameters for each combination of genetic operators are presented respectively in Tables 19-21. The time response graphics are illustrated for the best experiment from Tables 19-21 in Table 22. In these experiment, the weighting factors of overshoot, settling time and rising time are $w_1 = 1$, $w_2 = 1$, and $w_3 = 0$ respectively.

For the reduction with the sensory fusion method we obtained the following parameters: $a = 3$, $b = 4$, $c = 9.6$ and $d = 1.9$ (see experiment 6, Table 19). With these parameters the horizontal position of the ball is stabilized in 3 seconds with overshoot of 0 % (see Table 19, experiment 6, the second column), and the behavior of the angle position of the beam is shown in Table 19, experiment 6, the third column. The design specifications of the beam-and-ball system are totally satisfied.

Num.	a	b	c	d	Overshoot (%)	Settling Time (sec.)	Rising Time (sec.)
1.	3.9	5.5	8.2	3.1	0	4.8	2.3
2.	1.7	3.7	25.1	4.3	0	4.9	3.2
3.	3.4	3.9	9.1	1.5	0	3.1	1.8
4.	3	4.1	10.6	3.4	0	2.8	1.93
5.	2.8	4.2	12.1	4.8	0.8	3	1.96
6.	3	4	9.6	2.9	0	3	2

Table 19. The results obtained for the sensory fusion fuzzy controller.

For the reduction with hierarchical method we obtained the following parameters: $a = 8.9$, $b = 6.4$, $c = 1.1$ and $d = 2.3$ (see experiment 6 in Table 20). With these parameters the horizontal position of the cart is stabilized in 3 seconds with overshoot 0%.

For the reduction with the combination of methods we obtained the following parameters: $a = 8.6$, $b = 0.9$, $c = 2.8$ and $d = 3.3$ (see experiment 6 in Table 21). With these parameters the horizontal position of the ball is stabilized in 3 seconds with overshoot of 0.4% (see Table 21, experiment 6, the second column), and the behavior of the angle position of the beam is shown in Table 21, experiment 6, the third column. The design specifications of the beam-and-ball system are totally satisfied.

Num.	a	b	c	d	Overshoot (%)	Settling Time (sec.)	Rising Time (sec.)
1.	19.6	8.1	3.8	6.3	0	4.24	2.57
2.	8.7	1.2	3.9	4.3	0	3.69	2
3.	14.4	5.7	3.3	5.1	0	3.57	2.1
4.	6.9	1.2	2.7	3	0	2.99	1.62
5.	10.9	2.8	2.6	3.5	0	3	1.8
6.	8.9	2.2	2.4	3.2	0	3	1.7

Table 20. The results obtained for the hierarchical fuzzy controller.

Num.	a	b	c	d	Overshoot (%)	Settling Time (sec.)	Rising Time (sec.)
1.	8.7	3	3.1	3.4	15.6	5	1.2
2.	25.2	11.2	3.2	6.7	0	4.4	2.5
3.	12.3	3.1	2.5	3.7	0.97	3.24	1.93
4.	6.9	1.4	2.7	3.1	0	2.82	1.5
5.	9	0.9	3	3.5	0.5	3	1.8
6.	8.6	0.9	2.8	3.3	0.4	3	1.7

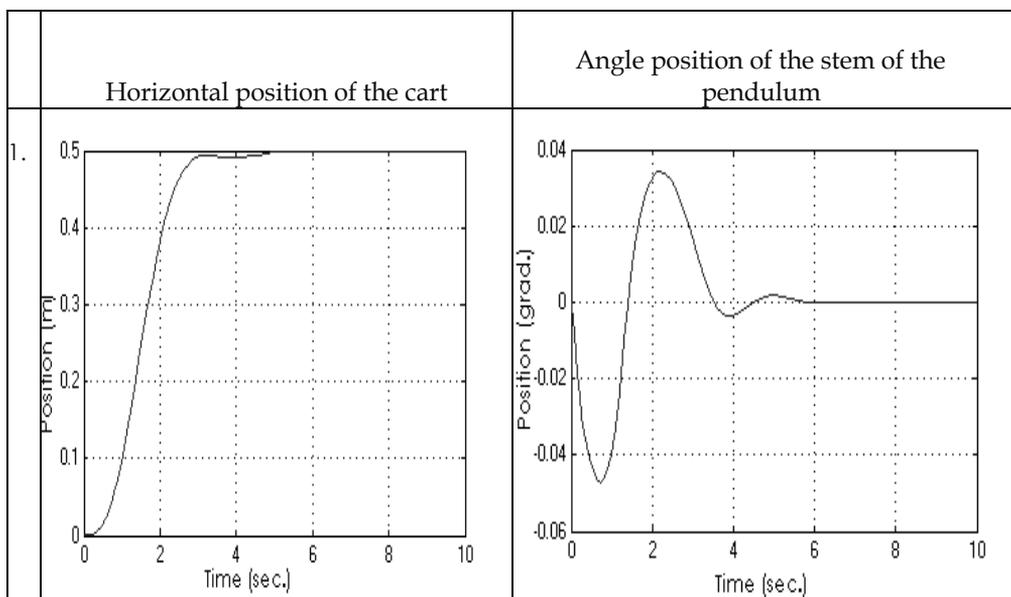
Table 21. The results obtained for the fusion-hierarchical fuzzy controller.

The fitness value convergence diagrams calculated for 30 generations are presented in Figure 19. The main observation is that the best combination of parameters can be met around 25 generations.

The sensory fusion controller, the hierarchical controller, and the fusion hierarchical controller were designed for the beam-and-ball control problem. The rule base of the sensory fusion controller was reduced from 625 to 9 rules. The rule base of the hierarchical controller was reduced from 625 to 27 rules. The rule base of the fusion hierarchical controller was reduced from 625 to 18 rules.

6. Conclusions

The sensory fusion method, the hierarchical method and the combination of these methods makes it possible to reduce the dimensionality of the control problem. In our approach, the problem of manually search for the required parameters was solved with an optimization algorithm (genetic algorithm). The proposed algorithm was tested by simulation of the inverted pendulum and beam-and-ball control problems. In both systems the fusion, the hierarchical, and the fusion-hierarchical parameters for the design specifications of this problem were adequately found. We conclude that manually search (can last several month if all of combination of parameters would be tried) are no more needed; instead the genetic estimation can be used. Due to the fact that the fitness function is based on the design specification of the system, we have the advantage to apply it to any combination of fusion-hierarchical variables. Another very important advantage is that when the user changes the design specifications, we can obtain the necessary fusion-hierarchical parameters very quickly by using the proposed GA. GA helped us not only to automatically estimate the fusion-hierarchical parameters, but also to improve the results obtained using fusion-hierarchical methods.



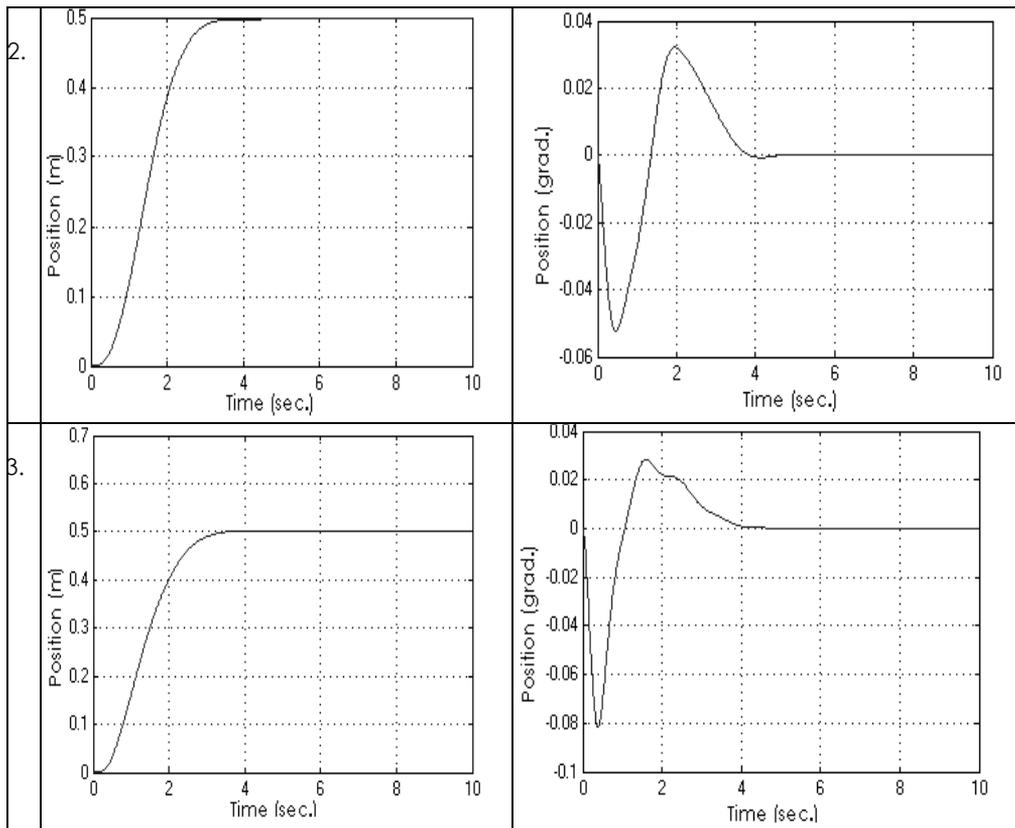


Table 22. The time response graphics obtained for the fusion-hierarchical fuzzy controller.

7. References

- Aguilar Ibañez, C., Gutiérrez Frias, O., and Suarez Castañon, M. (2005). Lyapunov-Based Controller for the Inverted Pendulum Cart System. In: *Nonlinear Dynamics 40*: 367–374 Springer.
- Aguilar Ibañez, C., Gutiérrez Frias, O. (2007). Controlling the inverted pendulum by means of a nested saturation function. In: *Nonlinear Dynamics* DOI: 10.1007/s11071-007-9224-3.
- Anwer, M.J, et al. (2005). A Simple Technique for Generation and Minimization of Fuzzy Rules. *IEEE International Conference on Fuzzy Systems*, CD-ROM Memories, Nevada.
- Ajith, Abraham (2001). Neuro Fuzzy Systems: State-of-the-art. Modeling Techniques. Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence, In : *Jose Mira and Alberto Prieto* (Eds.), *Lecture Notes in Computer Science*, Springer-Verlag, Spain, vol. 2084, pp. 269–276.
- Berenji, H. R., Khedkar, P. (1992). Learning and tuning fuzzy logic controllers through reinforcements. In: *IEEE Trans. Neural Networks*, vol. 3, pp. 724–740.
- Eshelman, Larry J. (1991). The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination, In: *Gregory J. E. Rawlins*

- (Ed.), Proceedings of the First Workshop on Foundations of Genetic Algorithms, pp. 265–283, Morgan Kaufmann.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley.
- Gwiazda, Tomasz D. (2006). Genetic Algorithms Reference vol.1 Crossover for single-objective numerical optimization problems, Tomasz Gwiazda, Lomianki, ISBN 83-923958-3-2.
- Halgamuge, S. K., Glesner, M. (1994). Neural networks in designing fuzzy systems for real world applications. In: *Fuzzy Sets and Systems*, vol. 65, pp. 1–12.
- Hung, Jonh Y., et al. (1993). Variable Structure Control: A Survey. In: *IEEE Trans. on Industrial Electronics*, vol. 40, no.1, pp. 2–21.
- Jamshidi, M. (1997). *Fuzzy Control Systems*. Springer-Verlag, chapter of Soft Computing, pp. 42–56.
- Jang, Jyh-Shing Roger (1993). ANFIS: Adaptive-Network-Based Fuzzy Inference Systems, In: *IEEE Trans. System Man & Cybernetics*, vol. 23, pp. 665–685.
- Juang, Chia-Feng and Lin, Chin-Teng. (1998). An On-Line Self-Constructing Neural Fuzzy Inference Network and Its Applications. In: *IEEE Transaction on Fuzzy Systems*, vol. 6, no.1, pp. 12–32.
- Kasabov, N., Kozma, R., and Duch W. (1998). Rule Extraction from Linguistic Rule Networks and from Fuzzy Neural Networks: Propositional versus Fuzzy Rules. In: *Proceedings of the Conference on Neural Networks and Their Applications (NEURAP)*, France, pp. 403–406.
- Kim J., Kasabov N. (1999). HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems. In: *Neural Networks*, no. 12, pp. 1301–1319.
- Klawonn, F., Höppner, F. (2003). What is Fuzzy About Fuzzy Clustering? – Understanding and Improving the Concept of the Fuzzifier. In: *M.R. Berthold, et al. (Eds.): Advances in Intelligent Data Analysis*. Springer Berlin, pp. 254–264.
- Ledeneva, Y. (2006a). Automatic Estimation of Parameters to Reduce Rule Base of Fuzzy Control Complex Systems. *Master thesis*, INAOE Mexico.
- Ledeneva, Y., Reyes García, C. A. (2006b). Automatic Estimation of Fusion Method Parameters to Reduce Rule Base of Fuzzy Control Complex Systems. In: *Gelbukh A., et al. (Eds.): MICAI 2006*, Springer-Verlag Berlin Heidelberg, LNAI 4293, pp. 146–155.
- Ledeneva, Y., Reyes García, C. A. (2007a). Automatic Estimation of Parameters for the Hierarchical Reduction of Rules of Complex Fuzzy Controllers. In: *Proceedings of ICINCO*, France, pp. 398–401.
- Ledeneva, Y., Reyes García, C.A., Gelbukh, A., García Hernández, R.A. (2007b). Genetic Optimization of the Parameters of Fuzzy Control Complex Systems. In: *Torres S. et al (Eds.): CORE 2007*, Research in Computing Science ISSN: 1870-4069, pp. 37–48.
- Melanie, Mitchell (1999). *An introduction to Genetic Algorithms*, MIT Press.
- Messner, W. C., Tilbury Dawn, M. (1998). Control Tutorials for Matlab and Simulink: A Web-Based Approach, Addison-Wesley.
- Nauck D., Kruse, R. (1994). NEFCON-I: An X-Window Based Simulator for Neural Fuzzy Controllers. In: *IEEE-ICNN, WCCI*.

- Nauck, D., Kruse, R. (1995). NEFCLASS - A Neuro-Fuzzy Approach for the Classification of Data. In : *Symposium on Applied Computing (SAC)*, Nashville.
- Tschichold-German, N. (1996). *RuleNet - A New Knowledge--Based Artificial Neural Network Model with Application Examples in Robotics*. PhD thesis, ETH Zurich.

Edited by Aleksandar Lazinica

This book represents the contributions of the top researchers in the field of robotics, automation and control and will serve as a valuable tool for professionals in these interdisciplinary fields. It consists of 25 chapter that introduce both basic research and advanced developments covering the topics such as kinematics, dynamic analysis, accuracy, optimization design, modelling , simulation and control. Without a doubt, the book covers a great deal of recent research, and as such it works as a valuable source for researchers interested in the involved subjects.

Photo by ABIDAL / iStock

IntechOpen

