IntechOpen

MATLAB

A Fundamental Tool for Scientific Computing and Engineering Applications - Volume 3

Edited by Vasilios N. Katsikis

MATLAB – A FUNDAMENTAL TOOL FOR SCIENTIFIC COMPUTING AND ENGINEERING APPLICATIONS – VOLUME 3

Edited by **Vasilios N. Katsikis**

MATLAB - A Fundamental Tool for Scientific Computing and Engineering Applications - Volume 3
http://dx.doi.org/10.5772/3339
Edited by Vasilios N. Katsikis

Contributors

Marian Gaiceanu, Rafael Cuerda Monzani, Afonso José Do Prado, José Pissolato Filho, Sérgio Kurokawa, Luiz Fernando Bovolato, Bhar Kisabo Aliyu, Charles Attah Osheku, Adetoro M.A.L, Funmilayo A. Aliyu, Leonardo Da Silva Lessa, Rodrigo Cleber Silva, Michal Blaho, Peter Fodrek, Martin Foltin, Ján Murgaš, Fatima El Guezar, Hassane Bouzahir, Cheng Siong Chin, Weal A. Altabey, Charis Harley, Hassan Al- Haj Ibrahim, Xiao-Guang Zhou, Vedran Vajnberger, Semir Silajdzic, Nedim Osmic, Cyril Belavý, Gabriel Hulkó, Karol Ondrejkovič, Ian Andrew Grout, Eric Anterrieu, Marcelo Reggio, Sébastien Leclaire, Maud El-Hachem, Nourdine Aliane, Rafael Pastor Vargas, Javier Fernandez Andrés, Antonio Napolitano, Sara Ungania, Vittorio Cannatà, Vasilios N. Katsikis

Notice

Statements and opinions expressed in the chapters are these 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, 2012 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

MATLAB - A Fundamental Tool for Scientific Computing and Engineering Applications - Volume 3
Edited by Vasilios N. Katsikis

p. cm.

ISBN 978-953-51-0752-1

eBook (PDF) ISBN 978-953-51-5708-3

Meet the editor

Dr Vasilios N. Katsikis received his Diploma of Mathematics from the University of Athens, his M.Sc. in Applied Mathematics and his Ph. D. in Mathematics from the National Technical University of Athens. He also worked as a post-doc researcher in Applied and Computational Mathematics under the financial support of the State Scholarship Foundation (IKY). During the years 1999-2009 he worked in several Greek universities and Technological Education Institutes and from September 2009 he belongs to the teaching and research stuff of the Department of Mathematics at the Technological Education Institute of Piraeus as an assistant professor of Mathematics. His research interests lie in the areas of Computational Mathematics, Functional Analysis, Computational Finance, Matrix Analysis and Applied Linear Algebra, Image and Signal Processing. He has published several articles in high quality journals concerning the above areas and he serves as a reviewer for many journals and congresses while he belongs to the editorial board of three journals.

Contents

Preface

"Πάντα κατ' αριθμόν γίγνονται. (All is number)"
Pythagoras

This excellent book represents the final part of three-volumes regarding MATLAB-based applications in almost every branch of science. The present textbook contains a collection of 19 high quality articles. In particular, the book consists of three sections, the first one is devoted to mathematical methods in the applied sciences by using MATLAB, the second is devoted to MATLAB applications of general interest and the third one discusses MATLAB for educational purposes. In what follows, we present a short summary focusing on the key concepts of each chapter.

Section 1: Mathematical Methods in the Applied Sciences

In **chapter 1**, the authors introduce a MATLAB toolbox for accurate and fast simulation of generic planar Piece-wise Affine Hybrid System (PWAHS). In particular, the authors propose a generic planar method to simulate PWAHSs with periodic and state dependent events. Using analytical expressions, their approach can reach arbitrate accuracy in event detections without any loss.

Chapter 2, demonstrates possibilities, how to exploit MATLAB based software support for analysis and design of robust control of Distributed Parameter Systems (DPS) with respect to uncertainty of models obtained by evaluation of measured data. The presented approach is based on the general decomposition of controlled DPS dynamics, represented by transient and impulse characteristics, into time and space components. Starting out from this dynamics decomposition a methodical framework is presented for the analogous decomposition of control synthesis into the space and time subtasks. In space domain approximation problems are solved, while in the time domain control synthesis is performed by lumped parameter SISO control loops, where various well-known methods for design of controller is possible to utilize. The advantage of this approach is the relatively simple LDS model of DPS, which is directly suitable for control purposes and can be easily identified from input-output data by means of classical techniques.

In **chapter 3,** fouling on heat exchanger surfaces is studied which it remains today one of the major unresolved problems in Thermal Science, although it has been recognised

for a long time and great technical advance in the design and manufacture of heat exchangers has been achieved.

Chapter 4, investigates the optimality of the solution to matrix Riccati algebraic equation which traditionally provides solution for Kalman gain with that of matrix Riccati differential equation. Both solutions were used in the synthesis of a Linear Quadratic Gaussian controller for an autopilot of a highly aerodynamically unstable Expendable Launch Vehicle model in pitch plane. It was proved here via simulation in MATLAB/SIMULINK that the solution obtained by solving a matrix Riccati differential equation gave an optimal result as regards time-domain response characteristics of the pitch controller.

In **chapter 5,** the efficient solution of the Frank-Kamenetskii partial differential equation through the implementation of parallelized numerical algorithms in MATLAB is discussed. Moreover, numerical algorithms and their ability to be structured in a manner suitable for their implementation on a GPU (Graphics Processing Units) are investigated. The computing time of these algorithms on the CPU is compared to those obtained when running the code on the GPU and these results discussed within the context of the numerical methods employed.

Chapter 6, is focused on how to solve a typical fuzzy decision making problem, that is, Fuzzy Analytical Network Process (FANP) by MATLAB software. According to Fuzzy Preference Programming method (FPP), local weights of fuzzy pairwise comparison matrixes can be achieved. Then an unweighted and weighted supermatrix based on its network structure can be formed. For FANP, the key step is to calculate the limit supermatrix until it is convergent. During the process, local weights and the limit supermatrix can be solved by MATLAB software. A case is given through the proposed method and MATLAB codes are provided as well.

In **chapter 7,** the authors define the most widely used and robust methods for fractal dimension estimation as well as their performances. Different methods and algorithms to estimate fractal dimension are described. To this end, flow charts and some part of MATLAB key code are included to provide an easy tool for the reader to reproduce the results. Limits and strong points of the described algorithms are outlined and a benchmark on their performance in a number of cases has also been reported. The fractal dimension estimation methods are applied to biomedical images.

Chapter 8, presents computational methods for option replication based on vector lattice theory. It is well accepted that the lattice theoretic ideas are one of the most important technical contributions of the large literature on modern mathematical finance. In this chapter, an incomplete market of primitive securities is presented, meaning that some call and put options need not be marketed. The objective is to describe an efficient method for computing maximal submarkets that replicate any option. Even though, there are several important results on option replication they cannot provide a method for the determination of the replicated options. By using the

theory of vector-lattices and positive bases it is provided a procedure in order to determine the set of securities with replicated options. Moreover, in this work the author determine those subspaces of the marketed subspace that replicate any option by introducing a MATLAB function, namely mrsubspace. The results of this work can give us an important tool in order to study the interesting problem of option replication of a two-period security market in which the space of marketed securities is a subspace of $\mathbb{R}^m$. This work is based on a recent work by the author, regarding computational methods for option replication.

In **chapter 9**, the authors present a description of isotropic gradient discretizations and convolution products. These isotropic gradients are useful, and superior to anisotropic discretizations especially in the field of flow simulation, when the lattice Boltzmann method is used. However, high order isotropic gradients are computationally expensive. To address this issue, the authors combined the convolution product wih the Jacket toolbox in MATLAB and GPU hardware, which enabled them to achieve high computational speedups, compared to plain MATLAB computations using a CPU. Moreover, the design of discrete operators or filters for the calculation of gradients is a classical topic in scientific computing. Typical applications are gradient reconstruction relevant in computational fluid dynamics, edge detection in computer graphics and biomedical imaging, and phase boundaries definition in the modeling of multiphase flows. In this chapter the authors describe in detail regarding isotropic and anisotropic discretizations that will and will not conserve the isotropic property of the differentiated function, respectively. This will be followed by a description on how convolution can be used to reduce computer time on the gradient calculation. Then a MATLAB implementation of these ideas, complemented with speedup comparisons applying convolution is performed.

Section 2: MATLAB General Applications

In **chapter 10,** it is discussed the development of technological knowledge, based on MATLAB/SIMULINK, related to grid connected power systems for energy production by using Renewable Energy Sources (RES), as clean and efficient sources for meeting both the environment requirements and the technical necessities of the grid connected power inverters. Moreover, another objective of this work is to promote the knowledge regarding RES. The chapter strategy follows two directions: the first, knowledge developments (a study and implementation of a high performance grid-power inverter; the fuel cells technology and the photovoltaic panels, as RES; the control methods; specific modelling and simulation methods); the second, the applicative research (a real time implementation with dSPACE platform is provided).

Chapter 11, presents the modeling and simulation of the solar tracker system consisting of the photovoltaic system under a constant load using MATLAB/SIMULINK. In particular, the overview of the electro-mechanical design of the single-axis solar tracker is described and this is followed by the description of the

proposed MATLAB/SIMULINK models. The chapter concludes with experimental results.

In **chapter 12,** the authors investigate the kinematics and dynamics of six degrees of freedom micro-robot intended for surgery applications. The kinematic model is based on Denavit-Hartenberg representation and the workspace of the end-effector is defined by solving the inverse kinematics problem. Four different methods were used to derive the trajectory planning for the six joints and were designed and employed to calculate the torque history for the six actuators. The dynamic equations of motion in symbolic form were derived using the Lagrange-Euler technique and the torque history was obtained using MATLAB for each joint. The proposed algorithm is flexible and can be extended to any robot configuration provided that the Denavit-Hartenberg presentation was available and the physical limits of joints are defined. The intent of this work is to show the convenience of MATLAB for micro-robots analysis.

Chapter 13, presents the usage and the benefits of remote control, and how to realize it using MATLAB. In particular, the design of a remotely controlled stepper motor and robotic arm via web server is presented. Operating algorithms and GUI were realized for both systems. Through the GUI for the stepper motor user can operate the motor in two modes: velocity and positional. The feedback received from the encoder is sent through the established connection from server to the client. Experimental results demonstrate the effectiveness of the remotely controlled stepper motor. Using the GUI for the robotic arm user can operate each joint of robotic arm separately. The feedback received from the camera is sent through the established connection from server to the client. Experimental results are not shown in this chapter, because as stated before, this model of robotic arm does not possess encoders. That is the reason why camera was used as visual feedback. The system operates in real time and visual feedback provides us information about current state of robotic arm. System is based on microcontroller and its development is not expensive, unlike the systems which are based on other technologies i.e., PLC. These systems are used in environments which are dangerous for humans.

Section 3: MATLAB for Educational Purposes

In **chapter 14**, the educational use of MATLAB and concepts for lecturers for the improvement of their courses are discussed. Main and biggest part of this chapter covers the most used MATLAB toolboxes.

Chapter 15, presents the use of MATLAB within a university education context and in particular the integration of MATLAB into the teaching and learning of semiconductor device fundamentals. In this work, consideration was given to the use of MATLAB in three teaching and learning scenarios; (i) at-presence "traditional" laboratory experiments; (ii) at-presence computer aided learning laboratories; and (iii), distance based remote access to laboratory experiments. Each scenario was introduced and the development of the laboratory experiments discussed. The physical infrastructure

(electronic hardware and software) was identified and the role in which technology is utilised in the education environment was presented.

In **chapter 16,** it is presented an interactive learning module focused exclusively on servo systems, which is aimed at providing insight not only into fundamental concepts but also into practical issues. The development of a MATLAB/SIMULINK-GUI application for servo systems learning is described, and its use in the classroom is also addressed. The tool is based on exploiting interactivity as a pedagogical basis in learning activities. Although many interactive tools have been developed for general topics in control education, interactive tools focusing on servo systems are practically nonexistent.

Chapter 17, presents MATLAB codes for solving the ordinary differential equations that cannot be solved without the aid of the computer in the field of non-linear optics as well as in relativistic electrodynamics, two domains of modern physics with non intuitive theories. The code is fast, accurate and easy to use. It has to be fed with only functions supplied by the student for switching from one study to another, the initial conditions of the problem being changed accordingly. Since the results are obtained almost in real time, these initial settings can be changed at will so that an interactive study is often conceivable with the computers in a classroom. In a classroom setting, the time to be allocated between the underground physics, the modeling issues and the freewheel experimentation is of course left to the teacher and may vary from one student to another. Numerical examples have demonstrated the capabilities of these codes for illustrating in teaching conditions some amazing effects of modern physics that cannot be brought to the attention of students without the aid of the computer.

In **chapter 18,** the application of the MATLAB software in analyses and simulations of transient phenomena in transmission lines is described. Moreover, the authors present an analysis based on resources used by MATLAB to obtain similar results to those obtained by ElectroMagnetic Transients type programs (EMTP), detailing the types of programming used to analyze the three-phase circuits in MATLAB using transformation matrices and also to evaluate programming performance of numerical methods and programming developed in MATLAB for the simulation of electromagnetic transients. The shown analyses and results can be used by undergraduate students for learning about the important concepts of power systems, transmission lines and wave propagation.

Chapter 19, introduces the basic concepts of electromagnetic transient simulations for a simplified model of transmission lines for transient simulations. The transmission lines are represented as a single-phase circuit and modeled through π circuits, using state variable equations for this representation. The obtained linear system can be solved by trapezoidal integration techniques. Based on these assumptions, it is obtained a simplified numeric routine for the first contact of undergraduate students with traveling wave studies. This numeric routine can lead to satisfactory precision

and accuracy for transient phenomena simulations on a single-phase representation of transmission lines. The mentioned routine is performed using MATLAB.

At this point, I would like to thank the authors for their great contribution in this series of scientific books regarding MATLAB applications in Sciences. Also, I thank the InTech team for their significant support during the preparation of this book.

Vasilios N. Katsikis
Department of Mathematics,
Technological Education Institute of Piraeus,
Greece

Section 1

Mathematical Methods in the Applied Sciences

Chapter 1

Simulation of Piecewise Hybrid Dynamical Systems in Matlab

Fatima El Guezar and Hassane Bouzahir

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/48570

1. Introduction

A hybrid dynamical system is a system containing on the same time continuous state variables and event variables in interaction. We find hybrid systems in different fields. We cite robotic systems, chemical systems controlled by vans and pumps, biological systems (growth and division) and nonlinear electronics systems.

Because of interaction between continuous and discrete aspects, the behavior of hybrid systems can be seen as extremely complex. However, this behavior becomes relatively simple for piece-wise affine hybrid dynamical systems that can, in contrast, generate bifurcation and chaos. There are many examples such as power electronics DC-DC converters.

The common power electronics DC-DC converters are the buck converter and the boost converter. They are switching systems with time variant structure [9].

DC-DC converters are widely used in industrial, commercial, residential and aerospace environments. These circuits are typically controlled by PWM (Piece Wise Modulation) or other similar techniques to regulate the tension and the current given to the charges. The controller decides to pass from one configuration to another by considering that transitions occur cyclically or in discrete time. In order to make the analysis possible, most of mathematical treatments use some techniques that are based on averaging or discretization. Averaging can mean to wrong conclusions on operation of a system. Discrete models do not give any information on the state of the system between the sampled instants. In addition, they are difficult to obtain. In fact, in most cases, a pure analytic study is not possible. Another possible approach to analyze these converters can be done via some models of hybrid dynamical systems. DC-DC converters are particularly good candidates for this type of analysis because of their natural hybrid structure. The nature of commutations of these systems makes them strongly nonlinear. They present specific complex phenomena such as fractals structures of bifurcation and chaos.

The study of nonlinear dynamics of DC-DC converters started in 1984 by works of Brockett and Wood [4]. Since then, chaos and nonlinear dynamics in power electronics circuits have attracted different research groups around the world. Different nonlinear phenomena have been observed such as routes to chaos following the period doubling cascade [16], [5], [19], [20] and [23] or quasi-periodic phenomena [6], [7] and [8], besides border collision bifurcations [23] and [2].

Switched circuits behavior is mostly simulated by pure numerical methods where precision step is increased when the system is near a switching condition. Those numerical tools are widely used mainly because of their ease-of-use and their ability to simulate a wide range of circuits including nonlinear, time–variant, and non–autonomous systems.

Even if those simulators can reach the desired relative precision for a continuous trajectory, they can miss a switching condition and then diverge drastically from the trajectory as in figure 1. This could be annoying when one is interested by border collision bifurcations, or when local behavior is needed with a good accuracy. In those applications, an alternative is to write down analytical, or semi–analytical, trajectories and switching conditions to obtain a recurrence which is very accurate and fast to run. Building and adapting such *ad'hoc* simulators represent a lot of efforts and a risk of mistakes.

Generic and accurate simulators can be proposed if we are restricted to a certain class of systems. A simulation tool with no loss of events is proposed in [14] and [15] for PWAHSs defined on polytopes (finite regions that are bounded by hyperplanes). This class of PWA differential systems has been widely studied as a standard technique to approximate a range of nonlinear systems.

But closed polytopic partition of the state space does not allow simulation of most switching circuits where switching frontiers are mostly single affine constraints or time–dependent periodical events.

This chapter follows our previous study in [13], [12] and [11].

We focus on planar PWAHSs with such simple switching conditions which can model a family of switched planar circuits: bang–bang regulators, the Boost converter, the Charge-Pump Phase Locked Loop (CP-PLL), ...

This class of systems has analytical trajectories that help to build fast algorithms with no loss of events. We propose a semi-analytical solver for hybrid systems which provides:

- A pure numerical method when the system is nonlinear or non-planar;
- A pure analytic method when all continuous parts of the system and switching conditions can be solved symbolically. This can be the case for the boost converter [3], [21], the second order charge-pump phase locked loop [17], [22].
- A mixed method using analytical trajectories and numerical computation of the switching instant when those solutions are transcendent. This has been used for the third order CP-PLL [17]. It can also be the case for the Buck converter [10], [21], ...

This chapter is organized as follows. Section 2 contains our main results. We describe the problem to be deal with, we introduce a general algorithm to solve planar HSs, we

present the algorithm that detects events' occurrence and devote a subsection to our approach efficiency. Section 3 Illustrates the current-mode controlled Boost converter example. Finally, a conclusion is stated in Section 4.

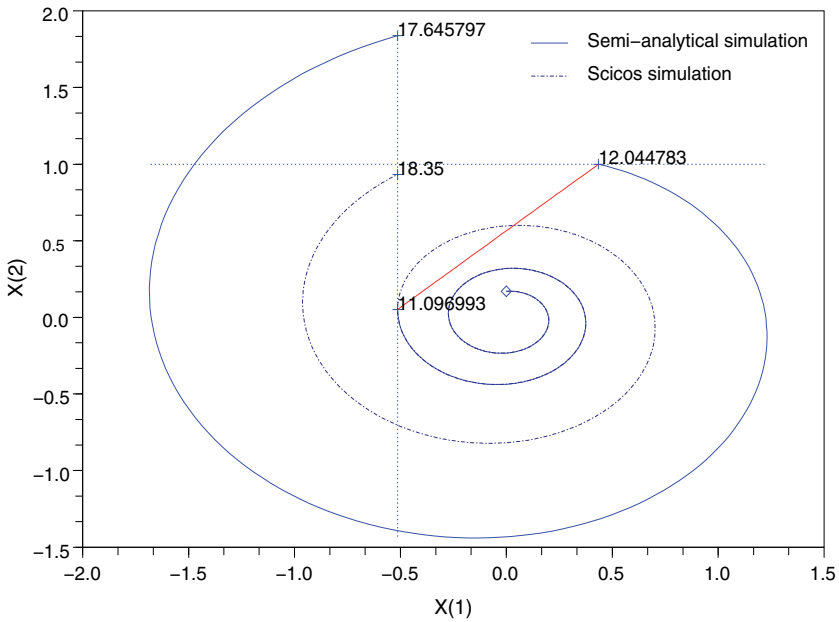


Figure 1. Numerical simulation versus semi–analytical simulation.

2. Main results

2.1. A HS (X, E, t): general definition

A general definition of HSs is presented here. This type of dynamical systems is characterized by the coexistence of two kinds of state vectors: continuous state vector $X(t)$ of real values, and discrete state vector $E(t)$ belonging to a countable discrete set $\mathcal{M}$.

Definition 1. *A continuous-time, autonomous HS is a system of the form:*

$$\begin{aligned} \dot{X}(t) &= F\left(X(t), E(t)\right), \; F : \mathcal{H} \to \mathbb{R}^n \\ E^+(t) &= \phi\left(X(t), E(t)\right), \; \phi : \mathcal{H} \to \mathcal{M} \end{aligned} \tag{1}$$

$\mathcal{H} = \mathbb{R}^n \times \mathcal{M}$ is called the hybrid state space. $X(t) \in \mathbb{R}^n$ is the continuous state vector of the HS at time instant t and $E(t) \in \mathcal{M} := \{1, \ldots, N\}$ is its discrete state. $E^+(t)$ denotes the updated discrete state right after time instant t. $\phi : \mathcal{H} \to \mathcal{M}$ describes the discrete dynamic, it is usually modeled by Petri Nets. A transition from $E(t) = i$ to $E^+(t) = j$ is valid when the state X reaches a switching set called $\mathcal{S}_{E_i,E_j}$. Such transitions are called state dependent events. A HS is called piece-wise affine if for each $E \in \mathcal{M}$, $F(X, E)$ can be defined by $F(X, E) = A_E X + B_E, \forall X$.

Remark — For non–autonomous HSs, the function ϕ can also depend on time $\phi(X, E, t) : \mathbb{R}^n \times \mathcal{M} \times \mathbb{R} \to \mathcal{M}$. Then time dependent events can occur and validate a transition, such as periodic events.

2.2. HSs class of interest

We consider a two dimensional PWAHS ($X(t) \in \mathbb{R}^2$). F has then the affine piece-wise form, $F(.,.)$ is defined for each $E \in \mathcal{M}$ and $X \in \mathbb{R}^2$ by $F(X(t), E(t)) = A_{E(t)}X + B_{E(t)}$, where $A_{E(t)} \in \mathbb{R}^{2\times 2}$ and $B_{E(t)} \in \mathbb{R}^2$ are two matrices that depend on the discrete state $E(t)$. Hence, a two dimensional PWAHS is a HS that take the form:

$$\begin{aligned} \dot{X}(t) &= A_{E(t)}X + B_{E(t)}, \\ E(t) &\in \mathcal{M} = \{1, 2, \ldots, N\} \end{aligned} \tag{2}$$

We consider two kinds of events: state dependent events and periodic events.

The state dependent event transition $\mathcal{S}_{E_iE_j}$ is defined by an affine state border of the form $N'_{ij}.X < l_{ij}$. In this case an event can occur when the continuous state reaches the border of the set $\mathcal{S}_{E_iE_j} = \left\{X(t) \in \mathbb{R}^2 : N'_{ij}.X \leq l_{ij}\right\}$.

Note that the set $\mathcal{S}_{E_iE_j}$ is not polytopic in the sense that the domain is not the interior of a closed bounded polytope.

Remark — We consider, without loss of generality, the case where a transition occurs at time $d\mathcal{S}_{E_iE_j}$if and only if the state $X(d\mathcal{S}_{E_iE_j})$ reaches a border of the set $\mathcal{S}_{E_iE_j}$ from outside. Figure 2 defines a transition with the complimentary set $\bar{\mathcal{S}}$, which allows to detect the event in both directions. Both transitions can be met with the set $B = \mathcal{S} \cup \bar{\mathcal{S}}$. Periodic

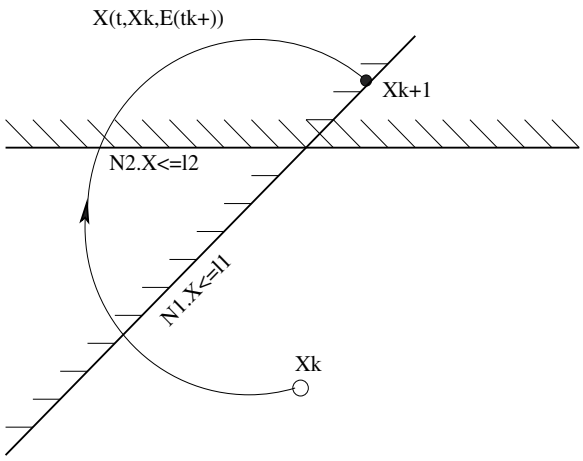


Figure 2. Oriented polytopic state dependent transitions.

events are simply defined by time instants $t = d\mathcal{P}_{E_iE_j}$, where $d\mathcal{P}_{E_iE_j}$ belongs to the set $\mathcal{P}_{E_iE_j} = \{t : t = kT + \varphi,\ k \in \mathbb{N}\}$. T is the period and φ is the phase of such periodic events.

2.3. Event–driven simulation of PWAHSs

The simulation will compute the hybrid state from event to event. Knowing the states $X(t_k)$ and $E(t_k^+)$, one can compute the trajectory $X(t > t_k) = \int_{t_k}^{t} f\left(X(t_k), E(t_k^+)\right) dt + X(t_k)$, assuming that the discrete state is constant $E(t > t_k^+) = E(t_k^+)$. Then the following algorithm runs the simulation determining the date at the next event as the smallest:

Data: t_k, X_k, E_k.
while $t < t_{fin}$ **do**
  Compute all events' dates $d\mathcal{S}_{E_iE_j}$ and $d\mathcal{P}_{E_iE_j}$;
  $t_{k+1} = \min(d\mathcal{S}_{E_iE_j}, d\mathcal{P}_{E_iE_j})$;
  $X_{k+1} = f\left(X_k, E_k, t_{k+1}\right)$;
  $E_{k+1} = \phi\left(X_k, E_k, t_{k+1}\right)$;
end

Algorithm 1: Algorithm computing the hybrid state at t_{k+1}.

2.4. Event detection occurrence: description and algorithm

We consider the affine Cauchy problem in $\mathbb{R}^2$:

$$\begin{cases} \dot{X}(t) = AX(t) + B,\ t > t_0 \\ X(t_0) = X_0 \end{cases} \tag{3}$$

where X_0 is the initial value. We compute the smallest strictly positive time t_i^* so that the trajectory of $X(t)$ intersects the fixed border B_i arriving from the part of the plan where $N_i'.X < l_i$. The function $f_i(t) = N_i'.X(t) - l_i$ defines the guard condition for a border B_i. Thus, the problem can be formulated as follows:

$$\text{Find the smallest } t_i^* \text{ such that } \begin{cases} f_i(t_i^*) = 0 \\ \exists\, \delta > 0, \forall t \in \left]t_i^* - \delta, t_i^*\right[,\ f_i(t) < 0 \end{cases} \tag{4}$$

If f_i does not have any strictly positive root or the last condition is not satisfied, t_i^* is given the infinite value.

2.4.1. Analytical trajectories

Definition 2. *For any square matrix A of order n and t in $\mathbb{R}$, the exponential matrix e^{tA} is defined by*

$$\mathrm{e}^{tA} = \sum_{k=0}^{\infty} \frac{t^k A^k}{k!} = \mathbb{I} + tA + \frac{t^2 A^2}{2!} + \frac{t^3 A^3}{3!} + \ldots \tag{5}$$

where $\mathbb{I}$ is the identity matrix.

It is well known that the analytical trajectory $X(t)$ of the initial value matrix differential equation (3) is given in terms of the exponential matrix and the variation of constants formula

by the general integral form:

$$X(t) = \mathrm{e}^{(t-t_0)A} X_0 + \int_{t_0}^{t} \mathrm{e}^{(t-s)A} B \mathrm{d}s. \tag{6}$$

When A is invertible, the above expression becomes linear:

$$X(t) + A^{-1}B = \mathrm{e}^{(t-t_0)A}(X_0 + A^{-1}B) \tag{7}$$

The analytical expression of the exponential matrix e^{At} takes two forms depending on whether the eigenvalues p_1 and p_2 of the matrix A are equal or not:

If $p_1 \neq p_2$, then

$$\mathrm{e}^{tA} = \frac{(p_1 \mathbb{I} - A^\circ)}{p_1 - p_2}\, \mathrm{e}^{p_1 t} - \frac{(p_2 \mathbb{I} - A^\circ)}{p_1 - p_2}\, \mathrm{e}^{p_2 t} \tag{8}$$

If $p_1 = p_2 = p$, then

$$\mathrm{e}^{tA} = (\mathbb{I} + (p\,\mathbb{I} - A^\circ)\; t)\; \mathrm{e}^{pt} \tag{9}$$

where $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, $A^\circ = \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$ and $\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

Using these expressions, we can determine the function $f(t)$ of the problem (4) as follows:

$$f(t) = a_1 + a_2\, t + a_3\, t^2 + (a_4 + a_5\, t)\, \mathrm{e}^{p_1 t} + a_6\, \mathrm{e}^{p_2 t}$$

where a_i are real scalars.

Depending on the eigenvalues p_1 and p_2, there are five cases that determine the values of the coefficients a_i as shown in Table 1. **Remark —** Coefficients a_i are real scalars that depend on

$f(t) = a_1 + \ldots$	$p_1 \in \mathbb{R}^*$	$p_1 = 0$
$p_2 \in \mathbb{R}^*$	$a_4\, \mathrm{e}^{p_1 t} + a_6\, \mathrm{e}^{p_2 t}$	$a_2\, t + a_6\, \mathrm{e}^{p_2 t}$
$p_2 = 0$	$a_2\, t + a_4\, \mathrm{e}^{p_1 t}$	$a_2\, t + a_3\, t^2$
$p_1 = \overline{p_2} \in \mathbb{C}^*$	$a_4 \mathrm{e}^{p_1 t} + a_5 \mathrm{e}^{p_2 t}$, with $a_5 = \overline{a_4} \in \mathbb{C}^*$	
$p_1 = p_2 \in \mathbb{R}^*$	$(a_4 + a_5\, t)\mathrm{e}^{p_1 t}$	

Table 1. Expressions of $f(t)$ depending on the eigenvalues p_1 and p_2.

the eigenvalues p_1 and p_2, the initial point X_k and the border parameters are N_i and l_i.

In some cases, ($p_1 = p_2 = 0$, gray cell in Table 1) roots of $f(t)$ can be found analytically and the problem is solved with machine precision.

In other cases, the solution can not be found with classical functions and then a numeric algorithm should be used. Using classical methods like Newton does not guaranty existence or convergence of the smallest positive root. To meet these conditions, let us use analytical

roots of the derivative function $f'(t)$ expressed in Table 2. We can then compute analytically

$f'(t) = \dots$	$p_1 \in \mathbb{R}^*$	$p_1 = 0$
$p_2 \in \mathbb{R}^*$	$a_4\, p_1\, \mathrm{e}^{\,p_1 t} + a_6\, p_2\, \mathrm{e}^{\,p_2 t}$	$a_2 + a_6\, p_2\, \mathrm{e}^{p_2 t}$
$p_2 = 0$	$a_2 + a_4\, p_1\, \mathrm{e}^{p_1 t}$	$a_2 + 2\, a_3\, t$
$p_1 = \overline{p_2} \in \mathbb{C}^*$	$a_4\, p_1\, \mathrm{e}^{p_1 t} + \overline{a_4 p_1} \mathrm{e}^{\overline{p_1} t}$, with $a_4 \in \mathbb{C}^*$	
$p_1 = p_2 \in \mathbb{R}^*$	$(a_4\, p_1 + a_5 + a_5\, p_1\, t)\ \mathrm{e}^{p_1 t}$	

Table 2. Expressions of $f'(t)$ depending on the eigenvalues p_1 and p_2

the set L of ordered roots of $f'(t)$, those roots determines monotone intervals of $f(t)$. The following algorithm is used to return the solution t^* when it exists or the value ∞ if not.

Remark — When $(p_1, p_2) \in \mathbb{C}^* \times \mathbb{C}^*$ the set L is infinite: when the real part of p_i is positive, the algorithm

```
Data: N_i, l_i, A, B, X_k
Result: construct the set L, compute t*
T ← {0, L, ∞};
t* ← ∞;
for i ← 1 to (card(T) − 1) do
    if f(T(i)) < 0 & f(T(i+1)) > 0 then
        t* ← solve [T(i), T(i+1)];
        Break;
    end
end
```

Algorithm 2: Algorithm computing t^* when a solution is transcendent.

will end by finding a root. In the other case, the set L should be reduced to its three first elements, to find a crossing point when it exists.

3. Matlab modelling

Our semi-analytical solver is composed of different main programs that define the studied affine system. First, we create the affine system given with a specifically chosen name. Then, we define the matrices A_i and B_i. After that, we give the switching borders with the sign of transitions and all necessary elements or we give the period if it is about a periodic event. Finally, we execute the simulation by specifying the initial state and the time of simulation.

3.1. Application: Current-mode controlled Boost converter

A current-mode controlled Boost converter in open loop consists of two parts: a converter and a switching controller. The basic circuit is given in figure 3.

This converter is a second-order circuit comprising an inductor, a capacitor, a diode, a switch and a load resistance connected in parallel with the capacitor.

The general circuit operation is driven by the switching controller. It compares the inductor current i_L with the reference current I_{ref} and generates the on/off driving signal for the switch S. When S is on, the current builds up in the inductor.

When the inductor current i_L reaches a reference value, the switch opens and the inductor current flows through the load and the diode. The switch is again closed at the arrival of the next clock pulse from a free running clock of period T.

The Boost converter controlled in current mode is modeled by an affine piece-wise hybrid system defined by the same sub-systems given in equation as follows:

$$\begin{aligned} S_1 : \dot{X}(t) &= \begin{bmatrix} \frac{-1}{RC} & 0 \\ 0 & 0 \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ \frac{V_{in}}{L} \end{bmatrix} \\ S_2 : \dot{X}(t) &= \begin{bmatrix} \frac{-1}{RC} & \frac{1}{C} \\ \frac{-1}{L} & 0 \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ \frac{V_{in}}{L} \end{bmatrix}, \end{aligned} \tag{10}$$

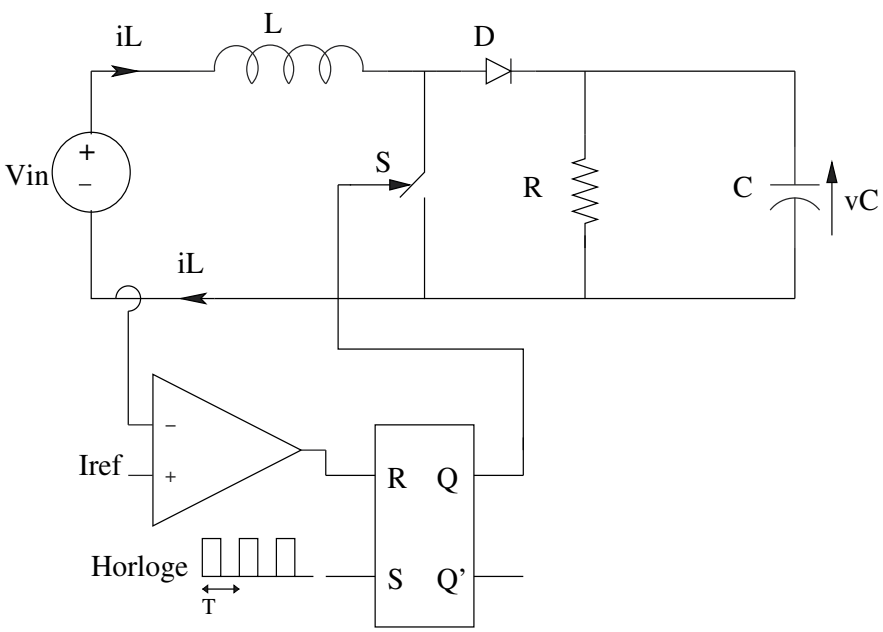


Figure 3. Boost converter controlled in current mode.

In the case of the Boost converter controlled in current mode, there are two types of events:

A state event defined by a fixed border of the set $\mathcal{S}_{E_1E_2}$:

$$\mathcal{S}_{E_1E_2} = \left\{ X \in \mathbb{R}^2 : [0\ 1]\, X < I_{ref} \right\} \tag{11}$$

and another periodic event defined by the dates $t = d\mathcal{P}_{E_iE_j}$, where $d\mathcal{P}_{E_iE_j}$ belongs to the set:

$$\mathcal{P}_{E_2E_1} = \{ t \in \mathbb{R} : t = nT,\ n \in \mathbb{N} \} \tag{12}$$

where T is the period of this periodic event. The different simulations are obtained using our planar PWA solver.

Before performing any study of the observed bifurcations in this circuit, a numerical simulation in the parametric plane is needed.

The following program calcule_balayage_mod.m is used to obtain the parametric plane:

```
%%%%----------------------------calcule_balayage_mod.m----------------
%
%
clear all;
close all;
%
%% File calculating points to display a figure of parametric plane
% Example of boost
% Save data in...
%
monfich=('data_balais');
% You should specify the path of hybrid_solver_matlab
%
addpath('.\hybrid_solver_matlab');
%
eps=1E-6;
ordre_max=15;
x_eps = 1e-5;
Xmax=100;
nb_trans=500;
nb_inits=1;
ta= 0.5:1.1/200:1.6;
tb= 5:15/200:20;
a=ta(1);
b=tb(1);
%% Definition of BOOST
%Iref changes and noted a
%Vin changes and noted b
%
L=1.5e-3;
T=100e-6;
R=40;
Vin=b;
C=5e-6;
Iref=a;
AE1=[
    -1/R/C  0 ;
    0    0
    ];
```

```
AE2=[
    -1/R/C  1/C ;
    -1/L    0
    ];
B1=[0;
    b/L];
B2=B1;
N1 = [0 1];
S1 = '<';
[p1,p2]=racines(AE1);
H=create_hybrid_system('affine');
H=add_state(H,1,'On',AE1,B1);
H=add_state(H,2,'Off',AE2,B2);
H=add_event(H,1,'Iref',N1,a,S1);
H=add_periodic_event(H,1,'Clock',T,0);
H=add_transition(H,1,2,1);
H=add_periodic_transition(H,2,1,1);
%
%% initial state
Xi.t=T/1000;
Xi.E=1;
Xi.Xc=[16.5;0.47];
mape=colormap(ma_color);
na=length(ta)
tic
for ia = 1 : na
    a=ta(ia);
    for ib= 1 : length(tb)
        b=tb(ib);
        %% update of the equation with a new a
        % here only Iref that changes and modifies a border
        B1=[0;
            b/L];
        H=add_state(H,1,'On',AE1,B1);
        H=add_state(H,2,'Off',AE2,B2);
        H=add_event(H,1,'Iref',N1,a,S1);
        H=update_transition(H,1,2,1);
        ordres(ib,ia)=-2;
        for init = 1:nb_inits
            X=Xi;
             or = ordre_max;
            for i = 1 : nb_trans
                [X]=recu(H,X); %1->2;
                [X]=recu(H,X); %2->1;
                if (X.t==Inf)
                    or=-1;
```

```
                    break;
                end
                if (max(abs(X.Xc))>Xmax)
                    or=0;
                    break;
                end
            end
            if (or==ordre_max)
        %% check if we have periodic event state E=1
                if (X.E ~= 1)
                   [X]=recu(H,X);
                end
                it=1;
                X0 = X;
                tt0=X.t;
                while (it<ordre_max) & (or == ordre_max)
                    [X]=recu(H,X); %1->2;
                            tt=X.t-tt0;
                            ii=1;
                            while (ii*T<tt)
                                it=it+1;
                                ii=ii+1;
                            end
                    [X]=recu(H,X);
                    tt0=X.t;
                    if (max(abs(X.Xc-X0.Xc))<x_eps)
                        or=it;
                        break;
                    else
                        it = it + 1;
                    end
                end
            end
            or;
            ordres(ib,ia)=max(or,ordres(ib,ia));
        end
    end
    fprintf('About %2.1f %% done, still about %5.0f secondes to be...
     %3.0f minutes\n',ia/na*100,toc/ia*(na-ia),toc/ia*(na-ia)/60)
end
temps=toc
save(monfich)
affiche_balayage
```

After calculating the necessary points of the parametric plane saved in the file named dat_balais, the next program affiche_balayage.m plots the figure given in Fig.4

```
%%%%----------------------------affiche_balayage_mod.m-------------
% Used in general after calcule_balayage

%% Charge the saved 2-D bifurcation scan
%if the file was not executed
if (exist('ordres')==1)
    disp('use the points matrix of the workspace');
elseif (exist('data_balais.mat')==2)
    disp('charge the points that are in data_balais.mat');
    load data_balais.mat
else
    disp('There are no points or files of points: try ordres.mat...
           insha ALLAH! It may be long...')
    load ordres.mat
end
%
%% Display the bifurcation scan diagram
da=(ta(2)-ta(1))/2;
db=(tb(2)-tb(1))/2;
colormap(mape)
for ia = 1 : length(ta)
    a=ta(ia);
    for ib= 1 : length(tb)
        b=tb(ib);
        if (ordres(ib,ia)<0)
            %plot(a,b,'.w');
            fill([a-da a-da a+da a+da],[b-db b+db b+db b-db],...
            'w','EdgeColor','none')
        elseif (ordres(ib,ia)==0)
            %plot(a,b,'+w');
            fill([a-da a-da a+da a+da],[b-db b+db b+db b-db],...
            'w','EdgeColor','none')
        else
            %plot(a,b,'s','color',mape(ordres(ib,ia),:),...
            'MarkerFaceColor',mape(ordres(ib,ia),:),'MarkerSize');
            fill([a-da a-da a+da a+da],[b-db b+db b+db b-db],...
            mape(ordres(ib,ia),:),'EdgeColor','none')
        end
       hold on
    end
end
colormap(mape)
colorbar %(mape)
colorbar('YTickLabel',...
        {'O1','O2','O3','O4','O5','O6','O7',...
         'O8','O9','10','11','12','13','14','O+'})
```

The figure 4 of the parametric plane allows to emphasize the parameters values for which there exists at least one attractor (fixed point, cycle of order k, strange attractor).

Figure 5 shows a bifurcation diagram (Feigenbaum type) in the plane (I_{ref}, i_L). However, figure 6 shows the bifurcation diagram in the space plane (I, i_L, v_C).

To draw these two figures we use programs: calcule_figuier.m and affiche_figuier.m

```
%%%%----------------------------calcule_figuier.m----------------
%
clear all;
close all;
%
%% Code that calculates then displays the points of a bifurcation
tree
% Boost converter example
% Save the data in ...
monfich=('data_points');
%
addpath('../hybrid_solver_matlab');
%
eps=1E-6; % precision of the solver
nb_trans=400;%400 %Number of iterations to pass the transient phase
ordre_max=100; %100% nombre de points affichés après le transitoire
ta= 0.5:0.0025:1.6; % values of the parameter a to be calculated
%ta= 1.22:0.001:1.4;%0.5:0.001:1.6;
points=zeros(2,length(ta),ordre_max);
% points (x or y, index a, the ordre_max of the last trajectory
points)
a=ta(1);
%
%% Definition of the Boost converter
%Iref is a variable denoted a
L=1.5e-3;
T=100e-6;
R=40;
Vin=10;
C=5e-6;
Iref=a;
AE1=[
    -1/R/C  0 ;
       0    0
    ];
AE2=[
    -1/R/C  1/C ;
    -1/L    0
    ];
```

```
B1=[0;
    Vin/L];
B2=B1;
N1 = [0 1];
S1 = '<';
[p1,p2]=racines(AE1)
H=create_hybrid_system('affine');
H=add_state(H,1,'On',AE1,B1);
H=add_state(H,2,'Off',AE2,B2);
H=add_event(H,1,'Iref',N1,a,S1);
H=add_periodic_event(H,2,'Clock',T,0);
H=add_transition(H,1,2,1);
H=add_periodic_transition(H,2,1,2);
%
%% Initial condition
X0.t=T/1000;
X0.E=1;
X0.Xc=[16.4549;0.4648];
%
%% Vary a and memorize the points for the bifurcation tree
na=length(ta);
tic;
for ia = 1 : na
    a=ta(ia);
    if a==1.3
    end
    %% Update of the equation with a new a
     vi=X.Xc;
     cc=ia;
    % Here only Iref varies and the corresponding border is then
    modified
    H=add_event(H,1,'Iref',N1,a,S1);
    H=update_transition(H,1,2,1);
    X=X0;
    %
    %% transient zone
    for i = 1: nb_trans
        [X]=recu(H,X);
          [X]=recu(H,X);
    end
    %% Assure that we are on a periodic event, state E=1
    %
     if (X.E ~= 1)
         [X]=recu(H,X);
     end
     %
```

```
      Xin=X.Xc;
       tt0=X.t;
       it=1;
    %% memorize ordremax points issued from the periodic transition
    2-> 1
    %
     while (it<(ordre_max+1))
        [X]=recu(H,X);%1->2
        tt=X.t-tt0;
        ii=1;
        while (ii*T<tt)
            points(:,ia,it)=traj_ni(AE1,B1,p1,p2,Xin,ii*T);
            Xin=traj_ni(AE1,B1,p1,p2,Xin,ii*T);
            it=it+1;
            ii=ii+1;
        end
      %
      [X]=recu(H,X);%2->1

      points(:,ia,it)=X.Xc;
      Xin=X.Xc;
      tt0=X.t;
      it=it+1;
    end
   %
  fprintf('Approximately %2.1f %% are done, yet approximately %5.0f
  seconds...
          that is %3.0f minutes\n',ia/na*100,toc/ia*(na-ia),toc/ia*
          (na-ia)/60)
end
%scan the values of 'a'
temps_ecoule=toc
save(monfich)
%%
cc
vi
affiche_figuier

%%%%----------------------------affiche_figuier.m----------------
%% Charges the file containing the saved points
% if the file "figuier" is not executed
if (exist('points')==1)
   disp('use the points matrix of the workspace');
elseif (exist('data_points.mat')==2)
   disp('charge the points that are in data_points.mat');
   load data_points
```

```
else
  disp('There are no points or files of points: try points.mat insh
  ALLAH!')
  load points
end
%% bifurcation tree depending on the dimensions x then y
% % for dim=1:2
% %
% %     plot(points(dim,1,1));
% %     hold on;
% %     for ia=1:length(ta)
% %         for io=2:ordre_max
% %             plot(ta(ia),points(dim,ia,io));
% %         end
% %     end
% %     xlabel('a');
% %     figure
% % end
%  figure
%
%     plot(points(1,1,1));
%     hold on;
%     for ia=1:length(ta)
%         for io=2:ordre_max
%             plot(ta(ia),points(1,ia,io));
%         end
%     end
%     xlabel('Vin(V)');
%     ylabel('vC(V)');
    plot(points(2,1,1));
    hold on;
    for ia=1:length(ta)
        for io=2:ordre_max
            plot(ta(ia),points(2,ia,io));
        end
    end
    xlabel('Iref(A)');
    ylabel('iL(A)');
%% bifurcation tree in 3D
% z = variable parameter denoted a
% x the dimension x
% y the dimension y of the point

  plot3(points(1,1,1),points(2,1,1),ta(1));
  plot3(ta(1),points(2,1,1),points(1,1,1));
  hold on;
```

```
for ia=1:length(ta)
    for io=1:ordre_max
        plot3(ta(ia),points(2,ia,io),points(1,ia,io));
    end
end
xlabel('Iref(A)');
ylabel('iL(A)');
zlabel('vC(V)');
```

In these two figures, the voltage V_{in} is fixed to 10V and the current I_{ref} varies in the interval $[0.5, 1.6]$. We observe a period cascade doubling leading to a chaotic regime, interrupted by a border collision bifurcation at $I_{ref} = 1.23$A (see figure 7). In this figure, a distinction is given between the attractors of attractive cycle type of the order 1 to 14. Each cycle of order k is associated with one color.

For example, the blue area O1 represents the parameters' values for which there exists an attractive fixed point (fundamental periodic regime). The red area O2 represents the existence of an attractive cycle of order 2. The yellow area O4 represents the existence of an attractive cycle of order 4 and so on until getting the cycles O14 of order $k = 14$. The black area O+ corresponds to parameters values (I_{ref}, V_{in}) for which there exist cycles of order $k \geq 15$ or other types of attractors. In this last area, a chaotic phenomenon could be observed. This bi-dimensional diagram shows some bifurcation curves. In fact, for the rectangle defined by the interval of parameter $V_{in} \in [7, 15]$ and the parameter $I_{ref} \in [0.5, 1.6]$, we observe an area of blue color (existence of attractive fixed point) followed by an area of red color (existence of cycle of order 2), an area of yellow color (existence of cycle of order 4) and another area of black color (existence of cycle of order $k \geq 15$ or another attractor type); this succession of zones corresponds to the existence of period doubling cascade.

This representation of the parametric plane is not enough to establish a bifurcation structure of the hybrid model of the Boost converter, but it is useful for the initialization of programs to draw bifurcation curves.

The simulation results (temporal domain and voltage-current plane (v_C, i_L)) are obtained using the planar PWH solver in the case of the Boost converter controlled in current mode for periods: $1T$ (figure 8) for $I_{ref} = 0.7$A , $2T$ (figure 9) for $I_{ref} = 1$A, $4T$ (figure 10) for $I_{ref} = 1.3$A and the chaotic regime (figure 11) for $I_{ref} = 1.5$A. For these plots we used the code below by choosing the bifurcation parameter I_{ref} corresponding to each period case.

```
%%
% Define an affine system ii at random and simulate it
% by use of our Matlab toolbox solver
%
%detection of errors manually
%
% Warning
% In the affine case '=' is not supported yet !
addpath('.\hybrid_solver');
%
```

```
L=1.5E-3;
T=100E-6;
R=40;
E=10;
C=5E-6;
%Bifurcation parameter
Iref=0.7; %1;%1.3;%1.5;
% X = [vc ; iL]
%
%System 1 On
A1=[ -1/R/C,  0 ;
     0        0  ];
B1=[0 ; E/L];
N1 = [0 1];
Lim1= Iref;

% System 2 Off
A2=[ -1/R/C,  1/C ;
     -1/L      0  ];
B2=[0 ; E/L];
%
%% Initial condition
X0.t=0;
X0.E=1;
X0.Xc=[16.4549;0.4648];%[13.6097 ;0.3435];
%
%% Boost converter
clear H;
H=create_hybrid_system('Boost Converter');
H=add_state(H,1,'On',A1,B1);
H=add_state(H,2,'Off',A2,B2);
%
H=add_event(H,1,'Iref',N1,Lim1,'<');
H=add_transition(H,1,2,1);
H=add_periodic_event(H,2,'Clock',T,0);
H=add_periodic_transition(H,2,1,2);
Han=H;
%
Xan = hsim(Han,X0,4*T);
%
[XcAn,EAn,tAn]=split_state(Xan);
%
 trajplane(Xan,Han)
     figure;
    subplot(211);
    trajplot(Xan,Han,1);
    subplot(212);
    trajplot(Xan,Han,2);
```

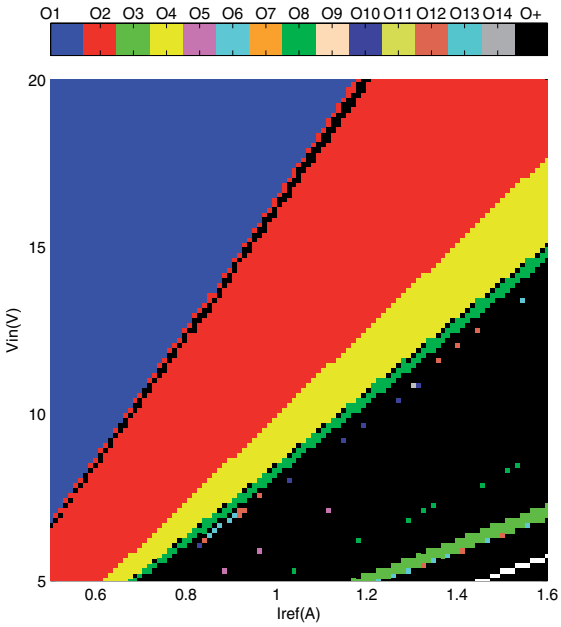


Figure 4. Parametric diagram of the Boost converter in the plane (I_{ref}, V_{in}) for $I_{ref} \in [0.5, 1.6]$A and $V_{in} \in [5, 20]$V.

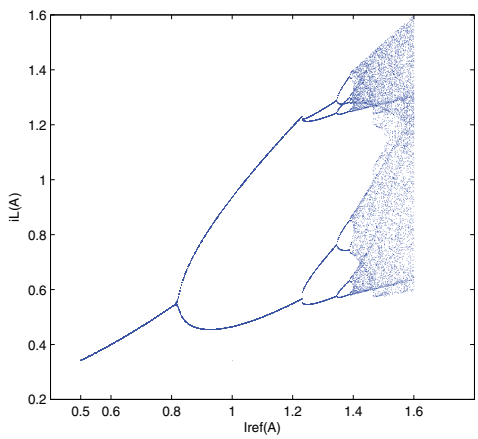


Figure 5. Bifurcation diagram of the Boost in the plane (I_{ref}, i_L) for $I_{ref} \in [0.5, 1.6]$A and $V_{in} = 10$V fixed.

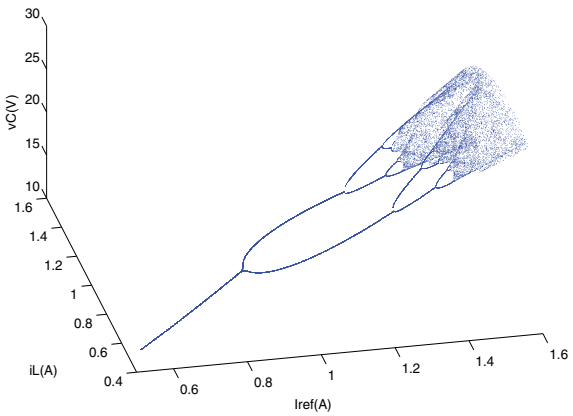


Figure 6. Bifurcation diagram of the Boost in the space (I_{ref}, i_L, v_c) for $I_{ref} \in [0.5, 1.6]$A and $V_{in} = 10$V fixed.

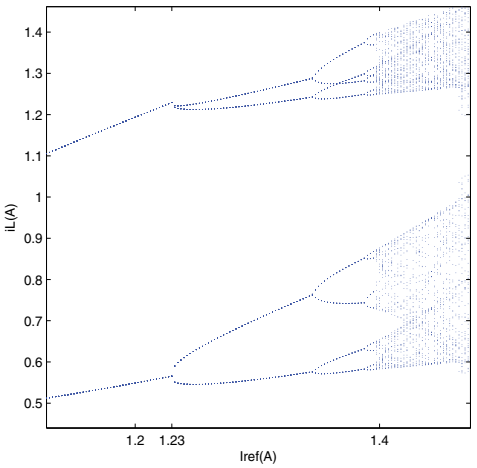
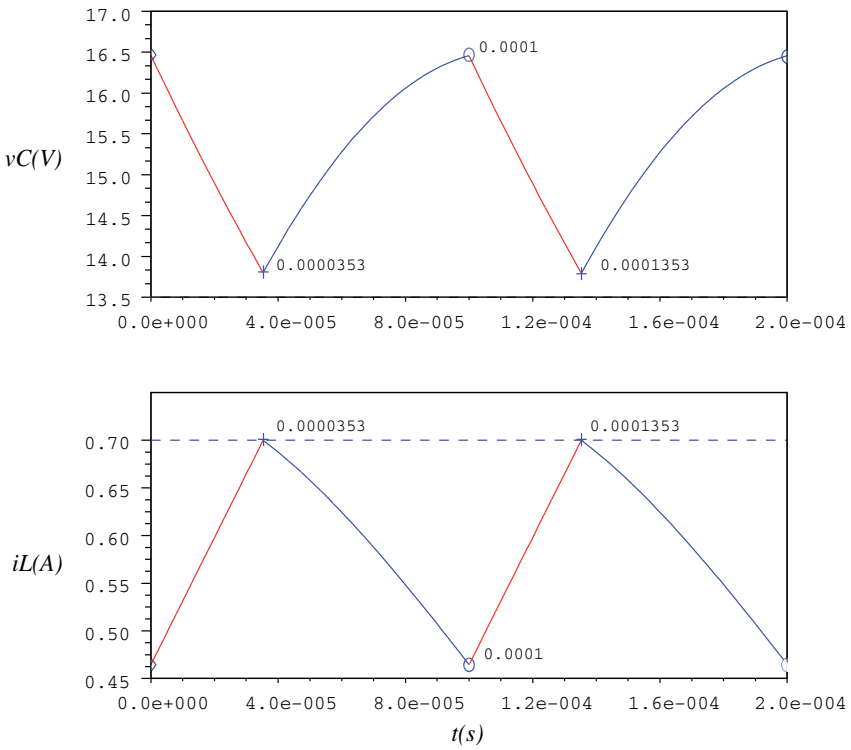
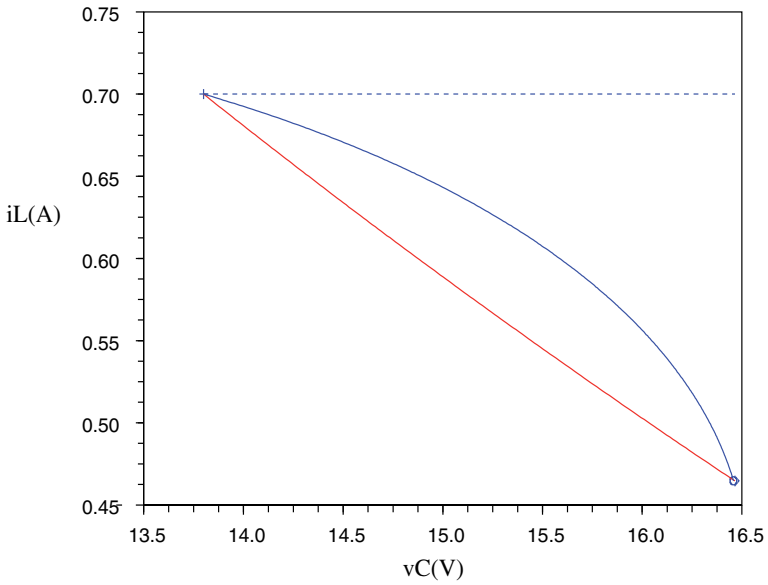


Figure 7. Zoom of figure 5: Border collision for $I_{ref} = 1.23$A.

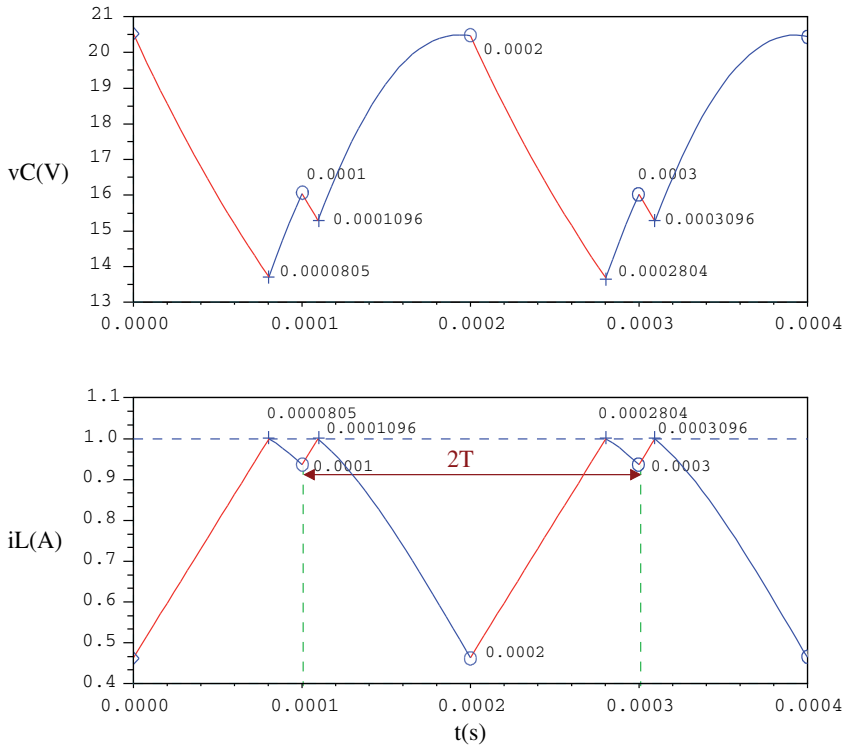


(a)

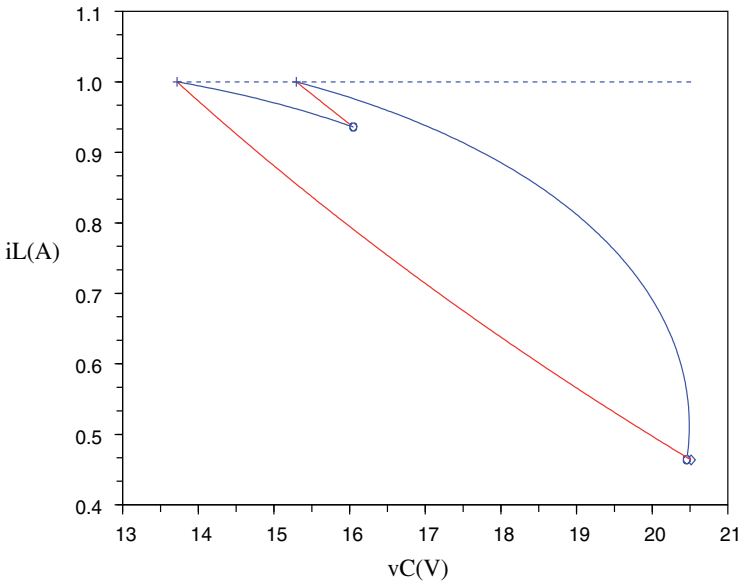


(b)

Figure 8. Fundamental periodic regime for $I_{ref} = 0.7$A: (a) (up) temporal waveform of the voltage v_C, (down) temporal waveform of the current i_L; (b) phase plane (v_C, i_L).

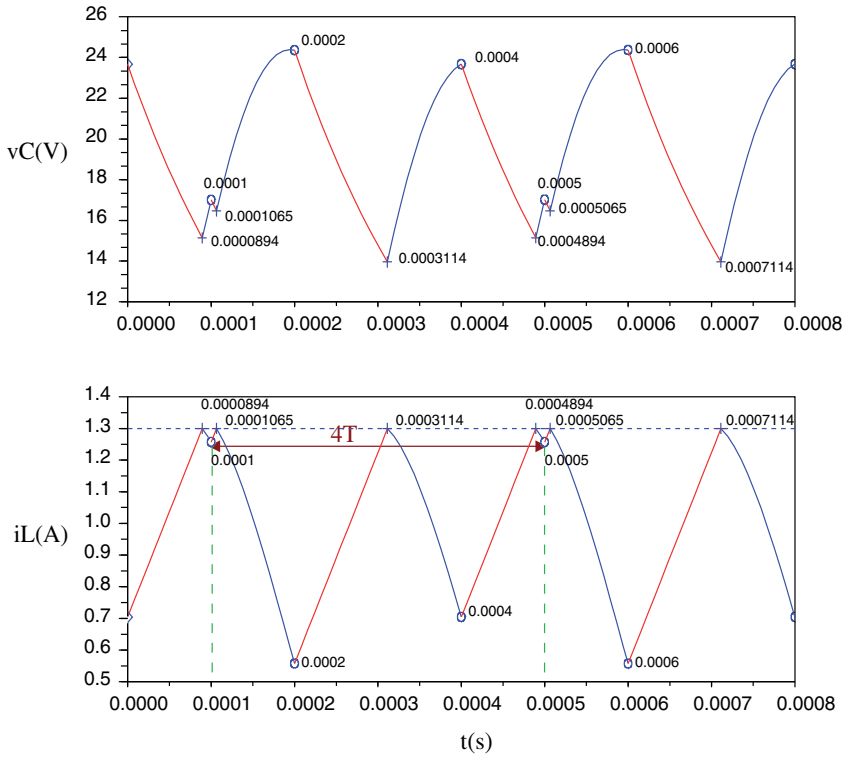


(a)

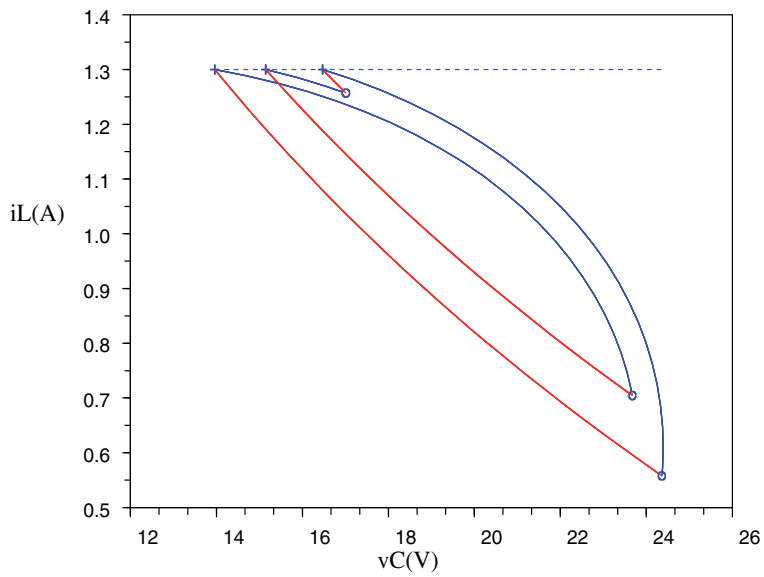


(b)

Figure 9. Cycle of order 2 for $I_{ref} = 1$A:(a) (up) temporal waveform of the voltage v_C, (down) temporal waveform of the current i_L; (b) phase plane (v_C, i_L).

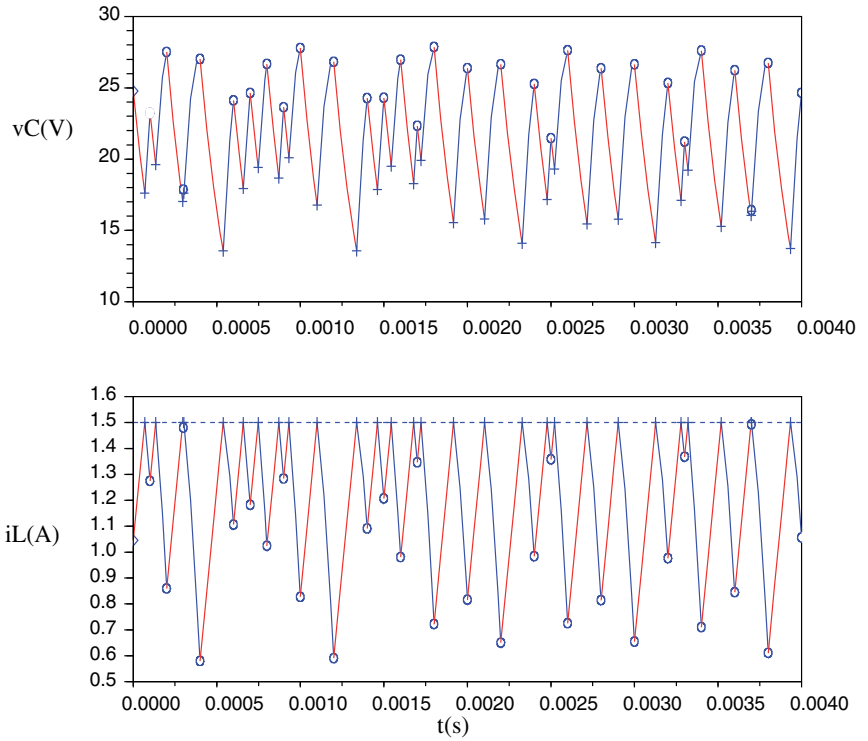


(a)

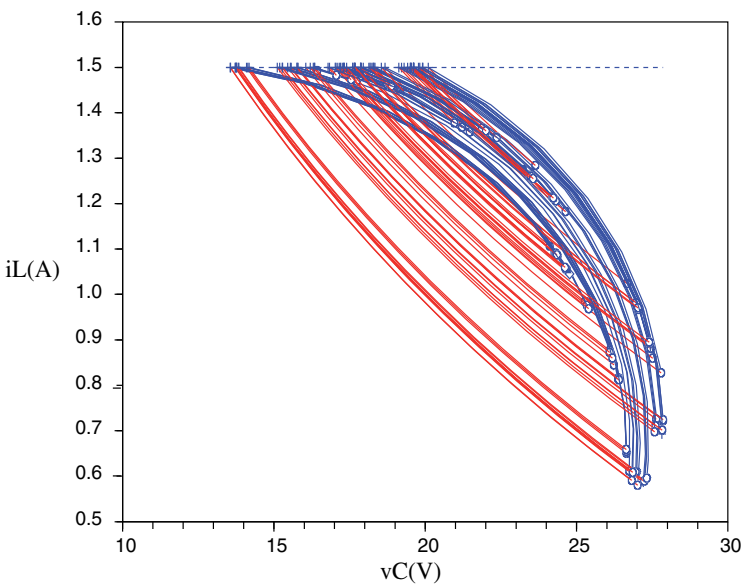


(b)

Figure 10. Cycle of order 4 for $I_{ref} = 1.3$A: (a) (up) temporal waveform of the voltage v_C, (down) temporal waveform of the current i_L; (b) phase plane (v_C, i_L).



(a)



(b)

Figure 11. Chaotic regime for $I_{ref} = 1.5$A: (a) (up) temporal waveform of the voltage v_C, (down) temporal waveform of the current i_L; (b) phase plane (v_C, i_L).

4. Conclusion

In this chapter, we have showed an accurate and fast method to determine events' occurrence for planar piece-wise affine hybrid systems. As a result, we have implemented our algorithm in Matlab toolbox version (free downloadable on http://felguezar.000space.com/).

This toolbox has also been completed by analysis tools such as displaying the bifurcation and parametric diagrams. The algorithm takes the advantage of the analytical form that appears in the planar case. Our approach can not be extended to a higher dimension. DC-DC converters like Boost converter are known to be simple switched circuits but very rich in nonlinear dynamics. As application, we have chosen the example of Boost converter controlled in current mode.

Acknowledgments

The authors would like to thank Pascal Acco and Danièle Fournier–Prunaret for crucial discussions on the original version of our work on this subject.

Author details

Fatima El Guezar
ESSI & ERMAGIM, Ibn Zohr University, EST, PO Box 32/S, Agadir, Morocco

Hassane Bouzahir
ESSI & ERMAGIM, Ibn Zohr University, EST, PO Box 32/S, Agadir, Morocco
Faculty of Engineering, AlHosn University, PO Box 38772, Abu Dhabi, United Arab Emirates

5. References

[1] Acco, P. (December 2003). Etude de la boucle à verrouillage de phase par impulsions de charge: Prise en compte des aspects hybrides. *Ph D thesis*, Institut National des Sciences Appliquées de Toulouse, France.

[2] Banerjee, S. (2000). Bifurcations in two-dimensional piecewise smooth maps - theory and applications in switching circuits, *IEEE Trans. on Circuits and Systems-I*, Vol.47, pp. 633-647.

[3] Bouzahir H.; El Guezar, F. & Ueta, T. (2007). On Scicos simulation of a hybrid dynamical system. *Proceedings of the 15th IEEE International Workshop on Nonlinear Dynamics of Electronic Systems*, Tokushima, Japan, pp. 62–65.

[4] Brockett R. W. & Wood J. R. (1984). Understanding power converter chaotic behavior mechanisms in protective and abnormal modes. *Proceedings of POWERCON 11*, pp. E–14.

[5] Deane, J. H. B. & Hamill D. C. (1990). Instability, subharmonics, and chaos in power electronic systems. *IEEE Trans. Power Electronics*, Volume 5, pp. 260–268, 1990.

[6] El Aroudi, A.; Benadero, L.; Toribio, E. & Machiche, S. (2000). Quasiperiodicty and chaos in the DC-DC buck-boost converter. *International Journal of Bifurcation and Chaos*, Vol. 10, pp. 359-371.

[7] El Aroudi, A. (February 2000). Study of nonlinear phenomena and quasiperiodicity route to chaos in PWM DC/DC converters. *Ph D thesis*, Universitat Politécnica de Catalunya, Spain.

[8] El Aroudi, A. & Leyva, R. (2001). Quasi-periodic route to chaos in a PWM voltage-controlled DC-DC boost converter. *IEEE Trans. on Circuits and Systems*, Vol. 48, No. 8, pp. 967-978.

[9] El Aroudi, A.; Debbat, M.; Giral, R.; Olivar, G.; Benadero, L.& Toribio, E. (2005). Bifurcations in DC-DC switching converters: review of methods and applications. *International Journal of Bifurcation and Chaos*, Vol. 5, pp. 1549–1578.

[10] El Guezar, F. & Bouzahir, H. (2008). Chaotic behavior in a switched dynamical system. *Modelling and Simulation in Engineering*, Vol. 2008, Article ID 798395, 6 pages.

[11] El Guezar, F.; Acco, P.; Bouzahir, H. & Fournier-Prunaret, D. (2008). Accurate and Fast Event Detection Occurrence in Planar Piecewise Affine Hybrid Systems. *Proceedings of the International Symposium NOLTA (NOn Linear Theory and its Applications)*, September 7-10, Budapest-Hungary, pp. 341–344.

[12] El Guezar, F. (December 2009). Modélisation et simulation des systèmes dynamiques hybrides affines par morceaux. Exemples en électronique de puissance. *Ph D thesis*, Institut National des Sciences Appliquées de Toulouse, France.

[13] El Guezar, F.; Bouzahir, H. & Fournier-Prunaret, D. (2011). Event Detection Occurrence For Planar Piecewise Affine Hybrid Systems. *Nonlinear Analysis: Hybrid Systems*, Vol. 5, pp. 626–638.

[14] Girard, A. (2002). Detection of event occurrence in piece-wise linear hybrid systems. *Proceedings of the 4th International Conference on Recent Advances in Soft Computing*, Nottingham, United Kingdom, December, pp. 19–25.

[15] Girard, A. (September 2004). Analyse algorithmique des systèmes hybrides. *Ph D thesis*, Institut National Polytechnique de Grenoble, France.

[16] Hamill , D. C. & Jeffries, D. J. (1988). Subharmonics and chaos in a controlled switched-mode power converter. *IEEE Trans. Circuits Systs. I*, Vol. 35, No. 8, pp. 1059–1060.

[17] Hedayat, C. D.; Hachem, A.; Leduc, Y. & Benbassat, G. (March–April 1997). High-level modeling applied to the second-order charge-pump PLL circuit. *Technical report, Texas Instrument Technical Journal*. Vol. 14, No. 2.

[18] Mira, C. (1987). Chaotic dynamics. *World scientific Publishing*.

[19] Tse, C.K. (1994). Chaos from a Buck switching regulator operating in discontinuous mode. *IEEE Transactions on International Journal of Circuit Theory and Application*. Vol. 22, No. 7, pp. 262–278.

[20] Tse, C.K. (1994). Flip bifurcation and chaos in three–state boost switching regulators. *IEEE Transactions on Circuits and Systems I: Theory and Applications*, Vol. 41, No. 1, pp. 16–23.

[21] Tse, C.K. (2003). *Complex behavior of switching Power converters*, CRC Press.

[22] Van Paemel, M. (July 1994). Analysis of a charge pump PLL: a new model. *IEEE Transactions on Communications*, Vol. 42, No. 7, pp. 2490-2498.

[23] Yuan, G. H.; Banerjee, S.; Ott, E. & Yorke, J. A. (1998). Border-collision bifurcations in the buck converter, *IEEE Trans. on Circuits and Systems-I*, Vol. 45, pp. 707–715.

Chapter 2

Robust Control of Distributed Parameter Systems with Demonstration in Casting Technology and MATLAB/Simulink/DPS Blockset Software Support

Cyril Belavý, Gabriel Hulkó and Karol Ondrejkovič

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46460

1. Introduction

Most of the dynamical systems analysed in engineering practice have the dynamics, which depends on both position and time. Such systems are classified as distributed parameter systems (DPS). The time-space coupled nature of the DPS is usually mathematically described by partial differential equations (PDE) as infinite-dimensional systems. However, from point of view of implementation of DPS control in technological practice, where a finite number of sensors and actuators for practical sensing and control is at disposal, such infinite-dimensional systems need to be approximated by finite-dimensional systems. There are many dimension reduction methods, which can be used to solve this problem.

In the first mathematical foundations of DPS control, analytical solutions of the underlying PDE have been used (Butkovskij, 1965; Lions, 1971; Wang, 1964). That is the decomposition of dynamics into time and space components based on the eigenfunctions of the PDE. Continuous and approximation theories aimed to control of parabolic systems presents monograph (Lasiecka & Triggiani, 2000). Methodical approach from the view of time-space separation with model reduction is presented in (Li & Qi, 2010). Variety of transfer functions for systems described by PDE are illustrated by means of several examples in (Curtain & Morris, 2009). Well-known reduction methods based on finite difference method (FDM), or finite element method (FEM), spectral method require an accurate nominal PDE model and usually lead to a high-order model, which requires unpractical high-order controller.

An engineering approach for the control of DPS is being developed since the eighties of the last century (Hulkó et al., 1981, 1987, 1998, 2009a, 2009b). In the field of lumped parameters system (LPS) control, where the state/output quantities $x(t)/y(t)$ – parameters are given as

finite dimensional vectors, the actuator together with the controlled plant make up a controlled LPS. In this sense the actuators and the controlled plant as a DPS create a controlled lumped-input and distributed-parameter-output system (LDS).

In this chapter the decomposition of dynamics of controlled LDS into time and space components is introduced. Based on this decomposition a methodical framework of control synthesis decomposition into space and time tasks will be presented. In the space domain, approximation problems are solved. In the time domain, synthesis of control is performed by lumped parameter control loops, where robust controllers are used.

The casting technology is a typical case of the DPS. There in order to obtain the desired solidification structure, the casting process requires a specific temperature field of the mould, which is defined on complex-shape 3D definition domain. Modelling, simulation and evaluation of real-time experiments in this area is now widely accepted as an important tool in product design and process development to improve productivity and casting quality. For analysis of the casting process dynamics as DPS, especially temperature fields in the casting mould and control synthesis purposes, the benchmark casting plant with steel mould of complex-shape was designed at Faculty of Mechanical Engineering STU in Bratislava.

The main emphasis of this chapter is to present an engineering approach for the robust control of DPS with demonstration in the casting technology along with software support in the MATLAB & Simulink programming environment. This approach opens a wide space for novel applications of the toolboxes and blocksets of the MATLAB & Simulink software environment. For the software support of modelling, control and design of DPS, given on 1D-3D definition domains, the **Distributed Parameter Systems Blockset for MATLAB & Simulink (DPS Blockset),** a Third-Party Product of The MathWorks Company - www.mathworks.com/products/connections/ has been developed at the Institute of Automation, Measurement and Applied Informatics, Faculty of Mechanical Engineering STU, (Hulkó et al. 2003-2010). Also a web portal named **Distributed Parameter Systems Control** - www.dpscontrol.sk has been created for those, who are interested in solving problems of DPS control (Hulkó et al., 2003-2007). This web portal contains application examples from different areas of engineering practice, such as the control of technological and manufacturing processes, mechatronic structures, groundwater remediation, etc. In addition, this web portal offers the demo version of the **DPS Blockset** with the **Tutorial, Show, Demos** and **DPS Wizard** for download, along with the **Interactive Control** service for the interactive solution of model control problems via the Internet.

In this chapter, for the control synthesis purpose, LDS models of temperature fields in the casting mould were created by means of evaluation of real-time experiments. Robust control synthesis based on internal model control (IMC) structure in the time domain has been done. Designed robust controllers were used for the robust control of preheating casting mould in the real-time experiment in accordance to casting technology requirements. Identification, uncertainty analysis of the models, robust control synthesis and experiments were performed with the software support of DPS Blockset and toolboxes of the MATLAB & Simulink, especially the System Identification Toolbox, Control Systems Toolbox, Robust

Control Toolbox, Optimization Toolbox, System Identification Toolbox, Real-Time Windows Target and Simulink Design Optimization.

2. LDS/HLDS representation of DPS

In general, DPS are systems whose state or output quantities, $X(x,y,z,t)/Y(x,y,z,t)$ are distributed quantities or fields of quantities, where (x,y,z) are spatial coordinates in 3D. These systems are often considered as systems whose dynamics is described by PDE. In the input-output relation, PDE define distributed-input/distributed-output systems (DDS) between distributed input, $U(x,y,z,t)$ and distributed output quantities, $Y(x,y,z,t)$, at initial and boundary conditions given. Distributed parameter systems are very frequently found in various technical and non-technical branches with limited number of manipulated input quantities, or actuators. These lumped input quantities by means of interaction of fields and quantities generate distributed output of real DPS. Representation of such DPS is either in the form of LDS, Fig. 1 a), or in the form of LDS with zero-order hold unit H (HLDS), when discrete-time lumped input quantities are used, Fig. 1. b) , (Hulkó et al., 1981, 1987, 1998).

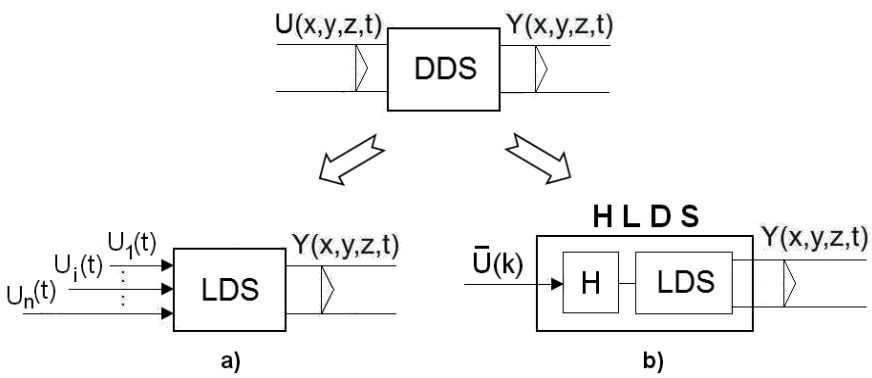


Figure 1. Representation of DPS: a) LDS - lumped-input/distributed-output system, b) HLDS –LDS with block of zero-order hold units H, $U(x,y,z,t)$ - distributed input quantity, $Y(x,y,z,t)$ - distributed output quantity, $\{U_i(t)\}_{i=1,n}$ - lumped input quantities, $\bar{U}(k)$ - vector of discrete-time lumped input quantities

2.1. Dynamics of LDS

Distributed output of the linear LDS from zero initial conditions either in continuous, or in discrete-time (DT) is in the form:

$$Y(\bar{x},t)=\sum_{i=1}^{n}Y_i(\bar{x},t)=\sum_{i=1}^{n}\mathcal{G}_i(\bar{x},t)\otimes U_i(t) \tag{1}$$

$$Y(\bar{x},k)=\sum_{i=1}^{n}Y_i(\bar{x},k)=\sum_{i=1}^{n}\mathcal{G}H_i(\bar{x},k)\oplus U_i(k) \tag{2}$$

where $\otimes$ denotes convolution product and $\oplus$ denotes convolution sum, $\bar{x}=(x,y,z)$ is position vector in 3D, $\mathcal{G}_i(\bar{x},t)$ - distributed parameter impulse response of LDS to the i-th input, $\mathcal{G}H_i(\bar{x},k)$ - DT distributed parameter impulse response of LDS with zero-order hold units H (HLDS) to the i-th input, $Y_i(\bar{x},t)$ - distributed parameter output quantity of LDS to the i-th input, $Y_i(\bar{x},k)$ - DT distributed parameter output quantity of HLDS to the i-th input, $U_i(t)$ - lumped input quantity, $U_i(k)$ - DT lumped input quantity, (Hulkó et al. 1998).

When $U_i(t)$ is a unit-step (Heaviside) function, $Y_i(\bar{x},t)$ is in the form of distributed step response function $\mathcal{H}_i(\bar{x},t)$. Similarly, for the unit-step function $U_i(k)$: $Y_i(\bar{x},k)\rightarrow \mathcal{H}H_i(\bar{x},k)$. For simplicity in this chapter distributed quantities are considered mostly as continuous scalar quantity fields with unit sampling interval in the time domain. Whereas DT distributed parameter step responses $\{\mathcal{H}H_i(\bar{x},k)\}_i$ of HLDS can be computed by common analytical or numerical methods, then DT distributed parameter impulse responses can be obtained as

$$\{\mathcal{G}H_i(\bar{x},k)=\mathcal{H}H_i(\bar{x},k)-\mathcal{H}H_i(\bar{x},k-1)\}_i \tag{3}$$

For points $\{\bar{x}_i=(x_i,y_i,z_i)\}_i$ located in surroundings of lumped input quantities $\{U_i(t)\}_i$, where partial distributed transient responses $\{\mathcal{H}_i(\bar{x}_i,t)\}_i$ attains maximal amplitudes, partial distributed output quantities are obtained in time-domain and next either continuous $\{S_i(\bar{x}_i,s)\}_i$, or discrete transfer functions $\{SH_i(\bar{x}_i,z)\}_i$ with sampling period T are identified.

$$\{Y_i(\bar{x}_i,t)=\mathcal{G}_i(\bar{x}_i,t)\otimes U_i(t)\}_i \quad\rightarrow\quad \{Y_i(\bar{x}_i,s)=S_i(\bar{x}_i,s)U_i(s)\}_i \tag{4}$$

$$\{Y_i(\bar{x}_i,k)=\mathcal{G}H_i(\bar{x}_i,k)\oplus U_i(k)\}_i \quad\rightarrow\quad \{Y_i(\bar{x}_i,z)=SH_i(\bar{x}_i,z)U_i(z)\}_i \tag{5}$$

For the space dependency and in the steady-state we can define reduced transient step responses between i-th input quantity at point $\bar{x}_i=(x_i,y_i,z_i)$ and corresponding partial distributed output quantity in the steady-state:

$$\left\{\mathcal{H}HR_i(\bar{x},\infty)=\frac{\mathcal{H}H_i(\bar{x},\infty)}{\mathcal{H}H_i(\bar{x}_i,\infty)}\right\}_i \tag{6}$$

for $\{\mathcal{H}H_i(\bar{x}_i,\infty)\}_i\neq 0$.

Dynamics of LDS/HLDS is decomposed to the time and space components:

Time Components of Dynamics $\{S_i(\bar{x}_i,s)\}_i$, or $\{SH_i(\bar{x}_i,z)\}_i$ - for given i and chosen $\bar{x}_i$

Space Components of Dynamics $\{\mathcal{H}HR_i(\bar{x},\infty)\}_i$ - for given i in ∞

2.2. Feedback control loop based on HLDS dynamics

Decomposition of dynamics enables also to decompose the control synthesis (CS) to time synthesis (TS) and space synthesis (SS) tasks in the feedback control loop of the distributed parameter system, Fig. 2.

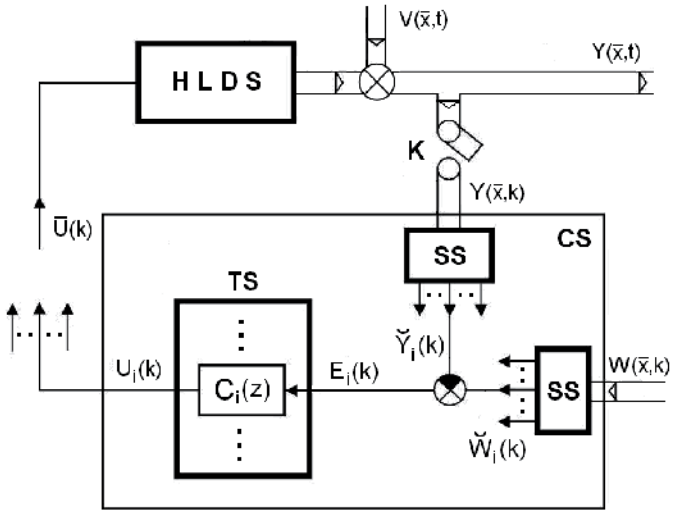


Figure 2. Distributed parameter feedback control loop: HLDS - LDS with zero-order holds $\{H_i\}_i$ on the input, CS - control synthesis, TS - control synthesis in time domain, SS - control synthesis in space domain, K - time/space sampling, $V(\bar{x},t)$ - disturbance quantity, $Y(\bar{x},t)$ - distributed controlled quantity, $Y(\bar{x},k)$ - sampled distributed controlled quantity, $\{\breve{Y}_i(k)\}_i$ - approximation parameters of controlled quantity, $W(\bar{x},k)$ - reference quantity, $\{\breve{W}_i(k)\}_i$ - approximation parameters of reference quantity, $\{E_i(k)\}_i$ - control errors, $\{C_i(z)\}_i$ - lumped parameter controllers, $\{U_i(k)\}_i$ - lumped control quantities

Let us consider a step change of distributed parameter control quantity $W(\bar{x},k)=W(\bar{x},\infty)$ and $V(\bar{x},t)=0$. The goal of the control synthesis is to generate a sequence of control inputs $\bar{U}(k)$ in such manner, that in the steady-state, for $k\to\infty$, the control error $E(\bar{x},k)$ will approach its minimal value $\|\breve{E}(\bar{x},\infty)\|$ in the quadratic norm:

$$\min\|E(\bar{x},\infty)\|=\min\|W(\bar{x},\infty)-Y(\bar{x},\infty)\|=\|\breve{E}(\bar{x},\infty)\| \tag{7}$$

First, in the SS blocks, the approximation both of sampled distributed controlled quantity $Y(\bar{x},k)$ and reference quantity $W(\bar{x},\infty)$, on the set of reduced steady-state distributed step responses $\{\mathcal{H}HR_i(\bar{x},\infty)\}_i$, are solved in following form:

$$\min_{\breve{Y}_i}\left\|Y(\bar{x},k)-\sum_{i=1}^{n}Y_i(\bar{x}_i,k)\mathcal{H}HR_i(\bar{x},\infty)\right\|=\left\|Y(\bar{x},k)-\sum_{i=1}^{n}\breve{Y}_i(k)\mathcal{H}HR_i(\bar{x},\infty)\right\| \tag{8}$$

$$\min_{\breve{W}_i}\left\|W(\bar{x},\infty)-\sum_{i=1}^{n}W_i(\bar{x}_i,\infty)\mathcal{H}HR_i(\bar{x},\infty)\right\|=\left\|W(\bar{x},\infty)-\sum_{i=1}^{n}\breve{W}_i\,\mathcal{H}HR_i(\bar{x},\infty)\right\| \tag{9}$$

Basis functions $\{\mathcal{H}HR_i(\bar{x},\infty)\}_i$ form a finite-dimensional subspace of approximation functions in the strictly convex normed linear space of distributed parameter quantities with

quadratic norm, where the approximation problem is solved. From approximation theory involves, that solution of the approximation problems (8), (9) is guaranteed as a unique the best approximation in the form $\sum_{i=1}^{n} \breve{Y}_i(k)\mathcal{H}HR_i(\bar{x},\infty)$ with the vector of optimal approximation parameters $\{\breve{Y}_i(k)\}_i$ in task (8) and the best approximation in the form $\sum_{i=1}^{n} \breve{W}_i \mathcal{H}HR_i(\bar{x},\infty)$ with the vector of optimal approximation parameters $\{\breve{W}_i\}_i$ for approximation task (9).

Let us formulate a DPS control problem for the distributed reference quantity $W(\bar{x},\infty)$. When $W(\bar{x},k)$ is assumed, the space control synthesis is performed in each time step *k*, which gives $\{\breve{W}_i(k)\}_i$ parameters. Graphical interpretation of the approximation problem (9) for HLDS defined on 1D space $\bar{x}$ is on Fig. 3.

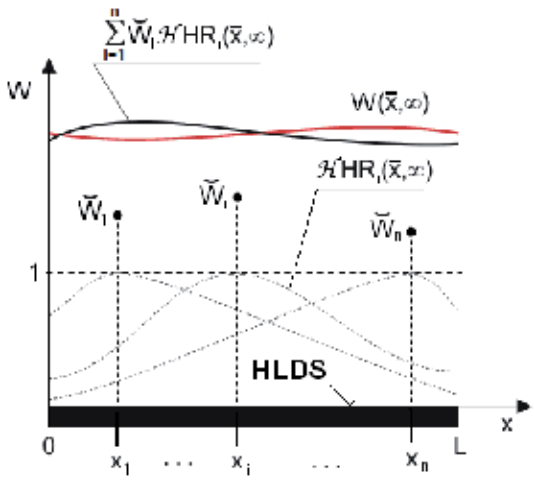


Figure 3. Solution of the approximation problem for the distributed reference quantity $W(\bar{x},\infty)$

Next, based on the solution of approximation problem, the vector of control error is created:

$$\bar{E}(k) = \{E_i(k)\}_i = \{\breve{W}_i - \breve{Y}_i(k)\}_i \tag{10}$$

The control errors vector $\bar{E}(k) = \{E_i(k)\}_i$ enters into the block TS, where the vector of control quantities, $\bar{U}(k) = \{U_i(k)\}_i$ is generated by controllers $\{C_i(z)\}_i$ in single-parameter control loops. During the control process, for $k \to \infty$ the control task (7) is accomplished.

Finally, we may state as a summary, that in the feedback control of DPS with dynamics represented in the form of HLDS, the control synthesis is performed as:

Space Tasks of Control Synthesis – as approximation tasks.
Time Tasks of Control Synthesis – on the level of lumped parameter control loops.

3. Robust control system

In general, a mathematical model for the plant dynamics is the basis for analysis and design of control systems. Also for LDS representation of DPS lumped and distributed models are used. However, in practice, no mathematical model describes exactly a physical process. It is obvious, that although no model represents the process exactly, some of them will do so with greater accuracy than others.

The theory of the robust control represents one of the possible approaches to the control system design in the presence of uncertainty. The goal of the robust system design is to retain a good quality of system performance in spite of model inaccuracies and changes. For the design techniques, the following requirements are supposed to be fulfilled: formulation of nominal plant model, different plant uncertainty models and requirements for both, robust stability and performance.

3.1. Sources of uncertainties in the LDS structure

LDS representation of DPS means decomposition of dynamics to space and time components. Uncertainties may occur in both, time and space components.

In distributed parameter control system, according to Fig. 2, single-input, single-output control loops in the block TS are tuned as closed feedback control loops using usual methods. In these loops, as models of the controlled system, transfer functions $\{S_i(\bar{x}_i, s)\}_i$ and/or $\{SH_i(\bar{x}_i, z)\}_i$ in the z-domain are used. These transfer functions describe the dynamics between sequences $\{U_i(k)\}_i$ and $\{Y_i(\bar{x}_i, k)\}_i$.

In this case, the sources of uncertainties are given by:

- procedure of dynamics modelling and possible change of parameters in models (4), (5)
- solution of approximation problem (8), (9), where lumped quantities are obtained

In order to treat uncertainties, it will be further assumed that the dynamic behaviour of a plant is described not by a single linear time invariant model, but by a family of linear time invariant models, Ψ_i. This family in the frequency domain, e.g. for models $\{S_i(\bar{x}_i, s)\}_i$, takes the following form:

$$\Psi_i = \left\{ S_i : \left| \; S_i(\bar{x}_i, j\omega) \; - \; \tilde{S}_i(\bar{x}_i, j\omega) \; \right| \le \bar{L}_{ai}(\omega) \right\} \tag{11}$$

where $\tilde{S}_i(\bar{x}_i, j\omega)$ is the nominal plant model. Any member of the family Ψ_i fulfils the conditions:

$$S_i(\bar{x}_i, j\omega) = \tilde{S}_i(\bar{x}_i, j\omega) \; + \; L_{ai}(j\omega) \tag{12}$$

$$\left| L_{ai}(j\omega) \right| \le \bar{L}_{ai}(\omega) \quad , \; \forall \; S_i \in \Psi_i \tag{13}$$

where $L_{ai}(j\omega)$ is an additive uncertainty and $\bar{L}_{ai}(\omega)$ is the bound of additive uncertainty. If we wish to work with multiplicative uncertainties, we define the relations:

$$L_{mi}(j\omega) = \frac{L_{ai}(j\omega)}{\tilde{S}_i(\bar{x}_i, j\omega)} \; ; \qquad \bar{L}_{mi}(\omega) = \frac{\bar{L}_{ai}(\omega)}{\left|\tilde{S}_i(\bar{x}_i, j\omega)\right|} \tag{14}$$

where $L_{mi}(j\omega)$ is a multiplicative uncertainty and $\bar{L}_{mi}(\omega)$ is the bound of multiplicative uncertainty.

3.2. Design of IMC robust controllers

A robust control system for HLDS will be designed using the Internal Model Control (IMC) strategy (Morari & Zafiriou, 1989) with the general structure depicted in Fig. 4. a). It is possible to transform this structure to the classical feedback control loop, Fig. 4. b) and to incorporate it into the TS block of the DPS feedback control system. The relationship between the classical feedback controller C and the IMC controller Q for the nominal model $\tilde{P}$ of the controlled process P is as follows, and vice-versa:

$$Q = \frac{C}{1+\tilde{P}C} \; ; \qquad C = \frac{Q}{1-\tilde{P}Q} \tag{15}$$

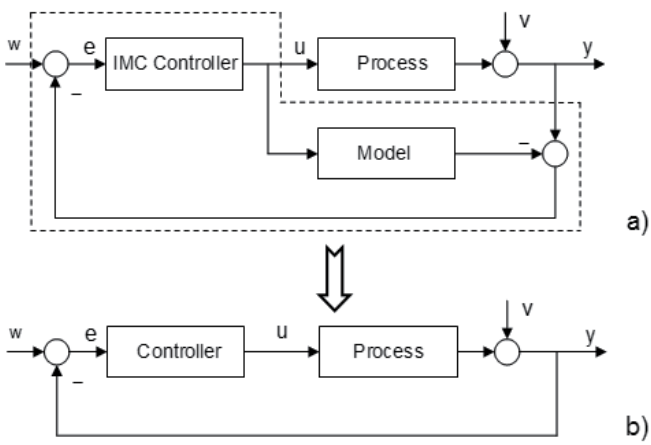


Figure 4. a) Internal Model Control structure, b) equivalent classical feedback control loop

It is well known that IMC strategy has the following properties:

1. *Dual stability*: Assume a perfect model, $(\tilde{P} = P)$ and if the controller and process are stable, then the IMC structure guarantees the closed-loop stability.
2. *Perfect control*: Assume a perfect model, $(\tilde{P} = P)$ and the closed-loop system is stable, while $Q = \tilde{P}^{-1}$, then there is no output steady-state error for set-point variance and disturbances.

The IMC structure thus provides the following benefits with respect to classical feedback: better dynamic response, system stability and robustness. One can search for Q instead of C without any loss of generality.

Structure of the distributed parameter feedback robust control system DPS with HLDS dynamics and IMC controllers is on Fig. 5.

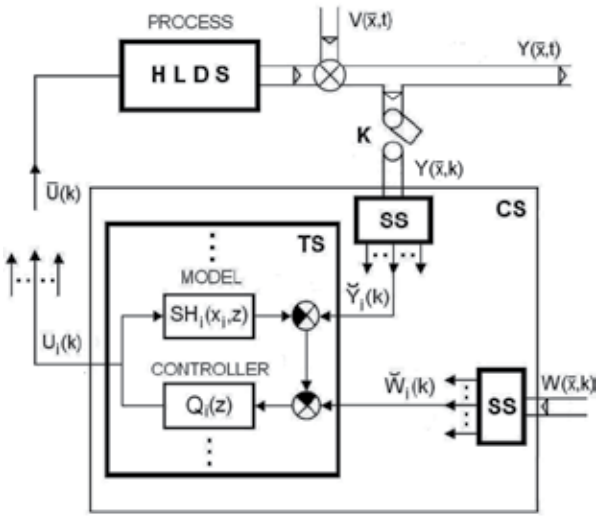


Figure 5. Distributed parameter feedback robust control system: HLDS - LDS with zero-order holds $\{H_i\}_i$ on the input, CS - control synthesis, TS - control synthesis in time domain, SS - control synthesis in space domain, K - time/space sampling, $V(\bar{x},t)$ - disturbance quantity, $Y(\bar{x},t)$ - distributed controlled quantity, $Y(\bar{x},k)$ - sampled distributed controlled quantity, $\{\breve{Y}_i(k)\}_i$ - approximation parameters of controlled quantity, $W(\bar{x},k)$ - reference quantity, $\{\breve{W}_i(k)\}_i$ - approximation parameters of reference quantity, $\{E_i(k)\}_i$ - control errors, $\{U_i(k)\}_i$ - lumped control quantities, $\{SH_i(x_i,z)\}_i$ - models of lumped controlled systems, $\{Q_i(z)\}_i$ - lumped parameter IMC controllers

H_2 optimal IMC controllers $\{Q_i(z)\}_i$ for inputs $\{\breve{W}_i(k)\}_i$ in the form unit-step function $\gamma(z)=\dfrac{z}{z-1}$ are obtained from solution of the following minimization problem:

$$\min_{Q_i(z)}\left\|e_i(z)\right\|_2=\min_{Q_i(z)}\left\|\left(1-SH_i(\bar{x}_i,z)Q_i(z)\right)\gamma_i(z)\right\|_2 \tag{16}$$

subject to the constraint $Q_i(z)$ to be stable and causal.

First, factorize the nominal stable transfer function $SH_i(x_i,z)$:

$$SH_i(\bar{x}_i,z)=SH_{Ni}(\bar{x}_i,z)\,SH_{Mi}(\bar{x}_i,z) \tag{17}$$

where $SH_{Ni}(\bar{x}_i,z)$ includes positive zeros or time-delays of the transfer function $SH_i(x_i,z)$. After this, optimal IMC controller $Q_i(z)$ is given by:

$$Q_i(z)=SH_{Mi}(\bar{x}_i,z)^{-1} \tag{18}$$

Finally, controller $Q_i(z)$ is augmented by low-pass filter $F_i(z)$ with parameter $0<\alpha_i<1$:

$$F_i(z) = \frac{1-\alpha_i}{z-\alpha_i} \tag{19}$$

Resulting IMC controller with filter $Q_{Fi}(z)$ is in following form:

$$Q_{Fi}(z) = Q_i(z)F_i(z) = Q_i(z)\frac{(1-\alpha_i)}{z-\alpha_i} \tag{20}$$

Parameter of the filter α_i is the only one tuning parameter to be selected by the user to achieve the appropriate compromise between performance and robustness and to keep the action of the manipulated variable within bounds. It must be chosen with respect to both, robust stability and robust performance condition:

$$\left|F_i\left(e^{j\omega T}\right)\right| < \left[\left|S_i(\bar{x}_i, j\omega)Q_i\left(e^{j\omega T}\right)\right|\bar{L}_{mi}(\omega)\right]^{-1}, \quad 0 \le \omega \le \frac{\pi}{T} \tag{21}$$

$$\left|\, Q_i(j\omega) \,\right|\bar{L}_{ai}(\omega) + \left|1 - S_i(\bar{x}_i, j\omega)Q_i(j\omega)\right|G_{wi}(\omega) < 1, \quad 0 \le \omega \le \frac{\pi}{T} \tag{22}$$

where $G_{wi}(\omega)$ is weighting function

For $SH_i(x_i, z) = SH_{Mi}(\bar{x}_i, z)$ and low-pass filter (19), robust controller $C_i(z)$ in equivalent classical feedback control loop takes form:

$$C_i(z) = \frac{SH_{Mi}(\bar{x}_i, z)^{-1}F_i(z)}{1 - SH_i(\bar{x}_i, z)\,SH_{Mi}(\bar{x}_i, z)^{-1}F_i(z)} = \frac{1}{SH_{Mi}(\bar{x}_i, z)} \cdot \frac{F_i(z)}{1 - F_i(z)} \tag{23}$$

4. Benchmark casting plant

The casting mould is one of the key components of a casting. It is well known, that the quality of the castings is affected strongly by the surface quality and the distribution of temperature in the mould, which has both time and space dependence. For study of the physical phenomena occurring during the casting solidification, from a DPS control point of view, control system development as well as mathematical model validation, a benchmark of the casting processes was designed (Belavý et al., 2009). At the study of casting processes and design of experimental plant, simulation studies in virtual software environments ProCAST and COMSOL Multiphysics were used.

4.1. Construction of the benchmark casting plant

Scheme of the benchmark casting plant is depicted in Fig. 6. The core item is the two-part steel mould of a complex-shape mounted in the frame of the ejector mechanism, Fig. 7. This mechanism is hinge-mounted to the main frame, to enable tilting of the mould for optimal filling and metal flow. Further, a hydraulic cooling circuit, which consists of an array of

induction motor driven roller vane pumps, a bunch of hoses, a flow divider, a collector with built-in check valves, a plate heat exchanger and an expansion tank. The main cooling circuit is divided into five independent circuits thus enabling the control of heat extraction from the casting via the chills. The coolant flow is controlled by means of frequency converters, since volumetric pumps are used. The main heating circuit is also divided into five independent circuits.

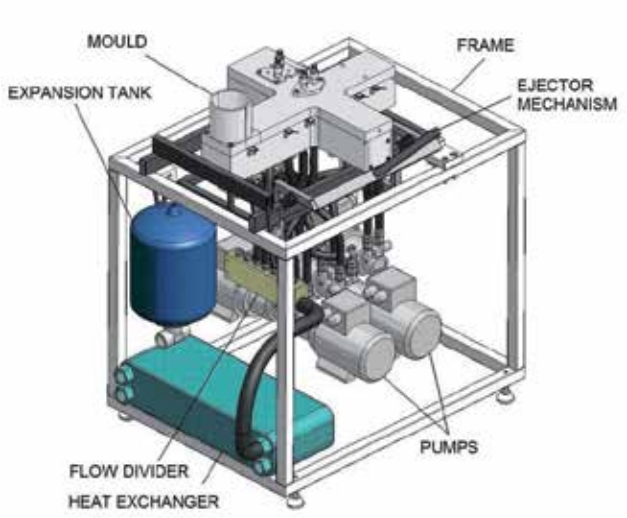


Figure 6. Scheme of the benchmark casting plant

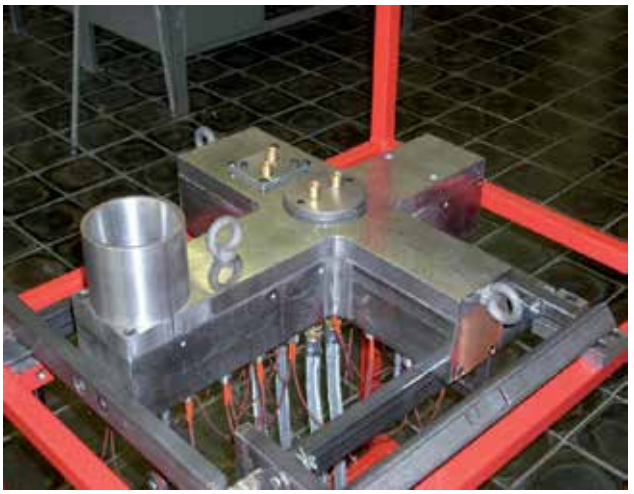


Figure 7. Two-part steel mould in detail

Inside of the casting mould are built-in 26 electric heating elements, each with maximal heating power 400 W. Heating elements are grouped to 5 zones and their heating power is actuated by the input voltage range of (0 – 10) V. In the body of the mould is also placed 7 water-cooled copper chills and 11 thermocouples, Fig. 8., Fig 9. Coordinates of measuring

points of thermocouples in *x*-*y* plane are given in Tab. 1 and *z* - coordinate is -0.05 m. Location of built-in elements has been carefully designed based on simulation studies in software environments ProCAST and COMSOL Multiphysics, in order to have the possibility of preheating the mould in 5 zones achieving desired temperature profile as well as directional solidification of the casting by means of active heat removal. The temperature field in the mould-casting system is possible to estimate through interpolation of data, measured by thermocouples.

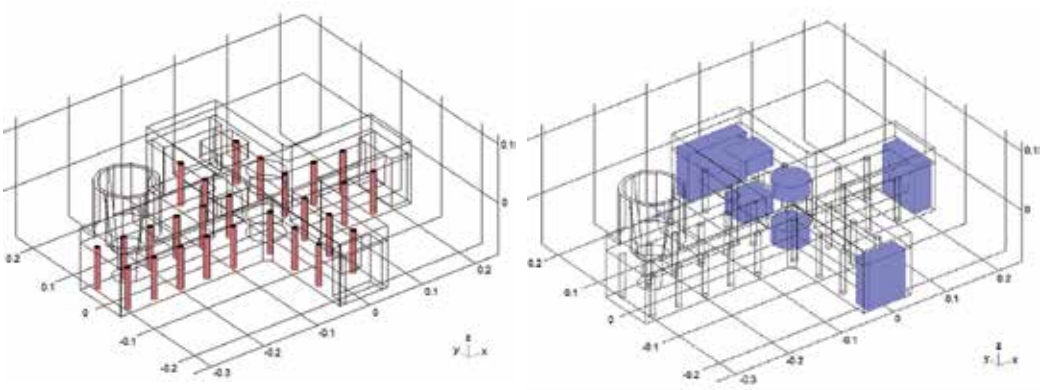


Figure 8. Location of heating elements (red) and copper chills (blue) in the mould

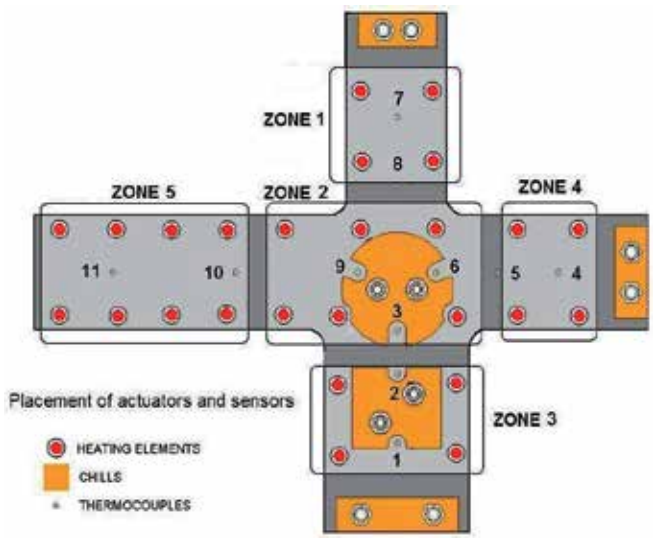


Figure 9. Bottom side of the steel casting mould

Position	1	2	3	4	5	6	7	8	9	10	11
x (m)	0	0	0	0.145	0.090	0.035	0	0	-0.035	-0.145	-0.255
y (m)	0.1585	0.0985	0.050	0	0	0	-0.135	-0.075	0	0	0

Table 1. Coordinates of thermocouples in *x*-*y* plane

4.2. Measurement and control scheme in MATLAB & Simulink

The measurement and control task of temperature field in the permanent casting mould was performed in the MATLAB & Simulink environment, where a *mould_exp_robust.mdl* scheme was setup, Fig. 10.

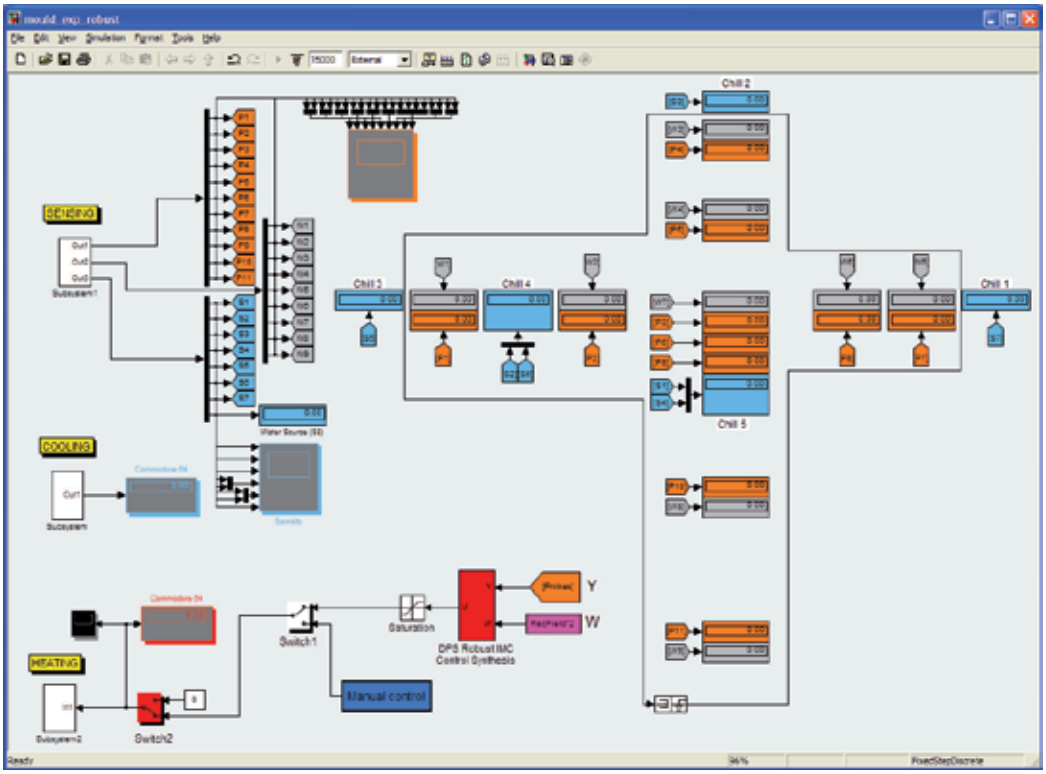


Figure 10. Measurement and control scheme in the MATLAB & Simulink

The scheme is composed of three main subsystems, namely: SENSING, HEATING and COOLING. Utilizing these subsystems and communication interface between process and computer, it is possible to measure dynamical characteristics of temperature field at zone heating. It is also possible to execute the experiment of controlled preheating of permanent mould before casting operation and controlled cooling during casting solidification. The above mentioned communication interface consists of data acquisition cards Advantech PCI-1710 [Ah], PCI-1710 [Ch] as analog input and Humusoft AD622 [Eh] as digital input.

Communication interface is performed by means of Simulink Real-Time Windows Target Toolbox and the communication is triggered by *Connect to Target* icon.

The SENSING subsystem enables the temperature measurement by thermocouples No. 1 to 11, which are permanently located in the bottom part of the steel mould, Fig. 9. These are marked on the scheme as P1 to P11. The subsystem also enables temperature measurements during casting solidification by temporal thermocouples W1 to W9, located in the casting

domain. Thermocouple sensing junctions are located in the middle of the arm cross section, right above permanent thermocouples "P", except the node of the casting, where one temporal thermocouple points to the center of the node. Sensors S1 to S7 measure the temperature of cooling water in embedded chills. *Display* blocks of the sensors "P" and "W" in the scheme correspond to the real positions of sensors. Measured temperatures are saved to the data file and continuously displayed on the *Scope* block.

The HEATING subsystem has two basic operational regimes, which are activated by *Switch2* block. In the manual control regime, it is possible to set heating performance in range (0÷10) Volt and measure the temperature transient characteristics with P1 to P11 thermocouples in given locations of the mould. The second regime activated by *Switch2* block enables to perform controlled preheating of the casting mould to desired temperature profile W defined in 11 points, where thermocouples P1 to P11 are located. The control task is performed by *DPS Robust IMC Control Synthesis* block.

5. Distributed Parameter Systems Blockset for MATLAB & Simulink

For the MATLAB & Simulink based software support of modelling, control and design of Distributed Parameter Systems given on complex 3D domains of definition, the programming environment **Distributed Parameter Systems Blockset for MATLAB & Simulink** (**DPS Blockset**) as Third-Party MathWorks Product has been developed by the Institute of Automation, Measurement and Applied Informatics, Faculty of Mechanical Engineering, Slovak University of Technology in Bratislava, within the program CONNECTIONS of The MathWorks Corporation, Fig. 11. , (Hulkó et al., 2003-2010).

The library of DPS Blockset shows Fig. 12. Blocks **HLDS** and **RHLDS** serve for modelling of distributed parameter systems as lumped-input/distributed-output systems with zero-order hold units. The block **DPS Control Synthesis** provides feedback to distributed parameter controlled systems in control loops with blocks for discrete-time **PID**, **Algebraic**, **State-Space** and **Robust Control**. The block **DPS Input** generates distributed quantities which can be used as distributed control quantities or distributed disturbances, etc. **DPS Display** presents distributed quantities with many options including export to AVI files. The block **DPS Space Synthesis** performs space synthesis as an approximation problem.

The block **DPS Wizard** in step-by-step operation, by means of several model examples with default parameters on 1D-3D definition domains, gives an automatized guide for arrangement and setting distributed parameter control loops. The block **Demos** contains examples oriented to methodology of modelling and control synthesis. The block **Show** contains motivation examples such as: *Control of temperature field of 3D metal body* (the controlled system was modelled in the virtual software environment COMSOL Multiphysics)**;** *Control of 3D beam of „smart" structure* (the controlled system was modelled in the virtual software environment ANSYS)*; Adaptive control of glass furnace* (the controlled system was modelled by Partial Differential Equations Toolbox of the MATLAB), and *Groundwater remediation control* (the controlled system was modelled in the virtual software environment MODFLOW)**.** The block **Tutorial** presents methodological framework both for formulation and solution of control tasks for distributed parameter systems.

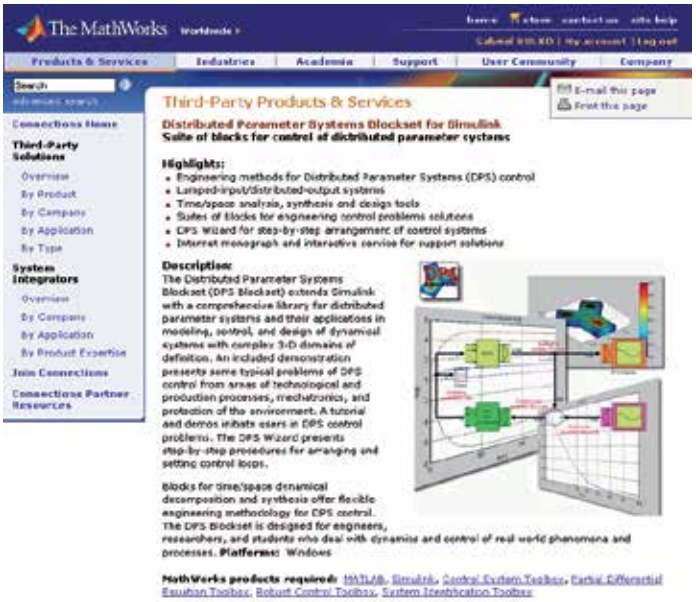


Figure 11. DPS Blockset on the web portal of The MathWorks Corporation

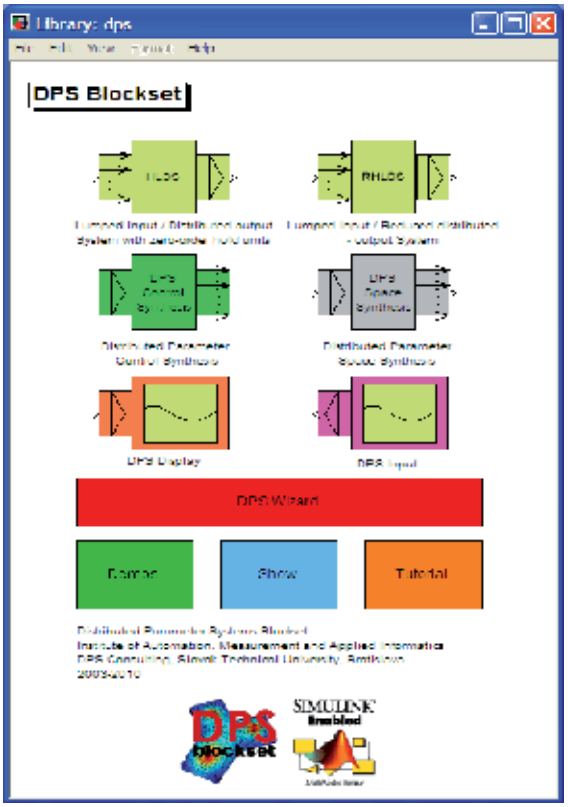


Figure 12. The library of DPS Blockset for MATLAB & Simulink

6. Dynamics of the temperature field in the casting mould

For control synthesis purpose, LDS/HLDS models of temperature fields in the casting mould have been created by means of evaluation of real-time experiments. The measurement of temperatures fields in the casting mould was performed by MATLAB & Simulink scheme *mould_exp_robust.mdl*, Fig. 10.

6.1. Experimental identification of transfer functions

In the casting mould lumped inputs $\{U_i\}_{i=1,5}$ are heating elements which act on sub-domains $\{\Omega_i\}_{i=1,5}$, (Zone 1 - 5). Distributed output is the temperature field of the casting mould.

Temperatures in the casting mould were measured by 11 thermocouples as a time-response to the step change of heating power, which was activated by the input voltage step from 0 to 2,5 V, for heating elements separately in each zone. Results of measurements for Zone #1, #3 and #5 are depicted in Fig. 13, where are time and *x*-*y* space dependences of temperatures and Fig. 14 presents temperature profiles in steady-states.

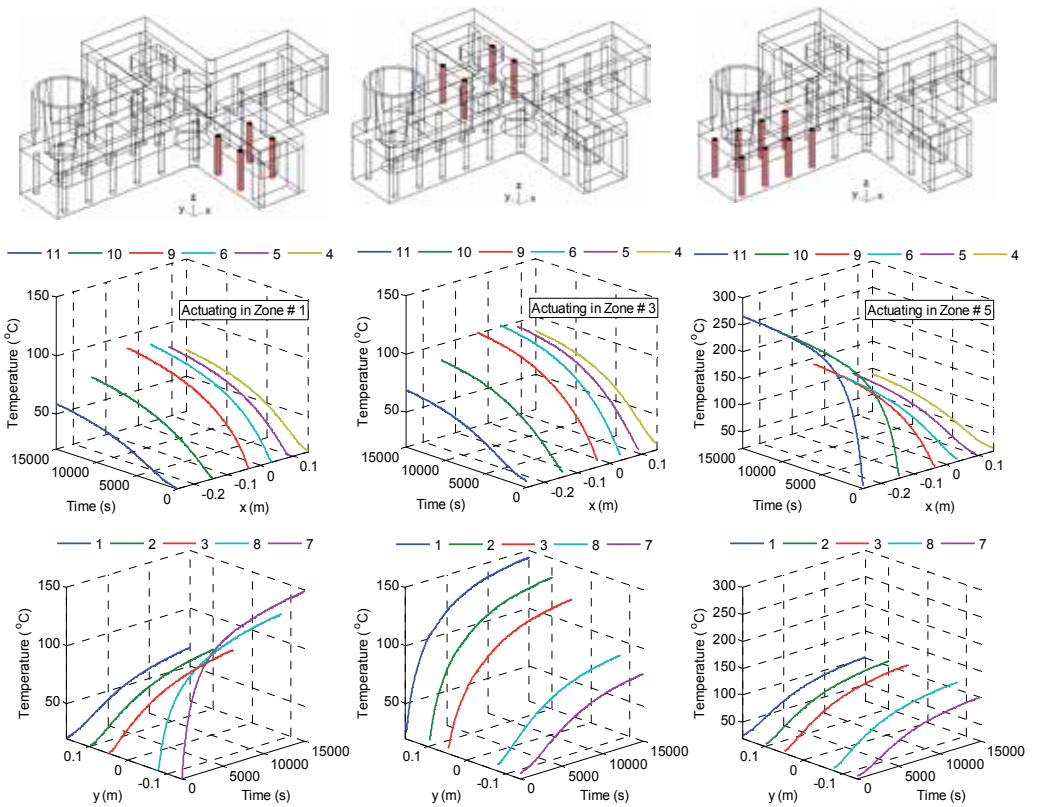


Figure 13. Time course of temperatures measured by thermocouples No. 1 - 11 after step change of the heating power separately in Zone #1, #3 and #5

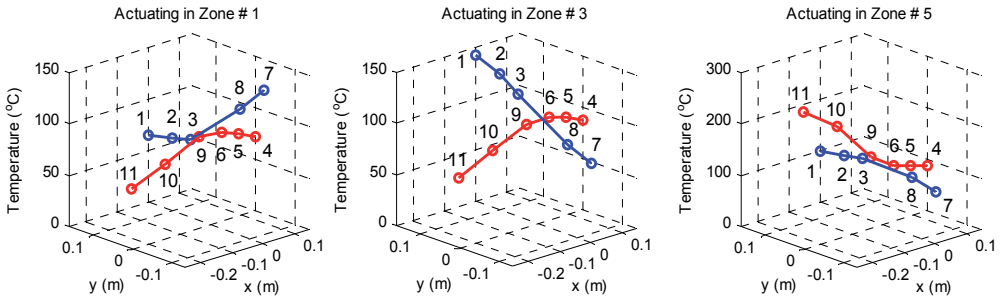


Figure 14. Temperature profiles in the steady-state measured by thermocouples No. 1 - 11 after step change of the heating power separately in Zone #1, #3 and #5

For identification of i-th transfer functions $\{S_i(\bar{x}_i,s)\}_{i=1,5}$, were determined points $\{\bar{x}_i=(x_i,y_i)\}_{i=1,5}$, located in each zone closely of lumped input quantities $\{U_i\}_{i=1,5}$, where temperatures attain maximal amplitudes by actuating the heating power in each zone separately. These points for actuating of the heating power in each zone are represented by positions of thermocouples given in Table 2.

Zone No.	1	2	3	4	5
Thermocouple No.	7	9	1	4	11

Table 2. Position of thermocouples for identification of measured temperatures

Identification of measured dynamical characteristics of temperatures was performed in the MATLAB software environment, where graphical user interface (GUI) *ident* from System Identification Toolbox was activated. There after importing the time domain input/output data, from the pop-up menu *Process models* transfer function in the form (24) for identification has been chosen, see Fig. 15., where are also results of identification from actuating in zone #1. Comparison of measured and identified model output in zone #1 is presented in Fig. 16. Identified parameters of transfer functions $\{S_i(\bar{x}_i,s)\}_{i=1,5}$ are in Table 3. Continuous transfer functions with structure (24) are converted by means of function *zpk* to zero-pole-gain format (ZPK). Then, for control synthesis purposes, they are transformed to discrete transfer functions $\{SH_i(\bar{x}_i,z)\}_{i=1,5}$ with sample time T= 10 s.

$$\frac{K(Tz\,s+1)}{(Tp_1\,s+1)(Tp_2\,s+1)} \tag{24}$$

Zone No.	1	2	3	4	5
K	53.40	58.16	49.36	62.56	97.87
Tp_1	5881.68	4945.00	5489.58	5340.97	4578.71
Tp_2	939.66	1958.20	1398.20	973.90	1214.73
Tz	3616.81	3301.80	3201.10	3195.90	3410.50

Table 3. Identified parameters of transfer functions $\{S_i(\bar{x}_i,s)\}_{i=1,5}$ for actuating in each zone

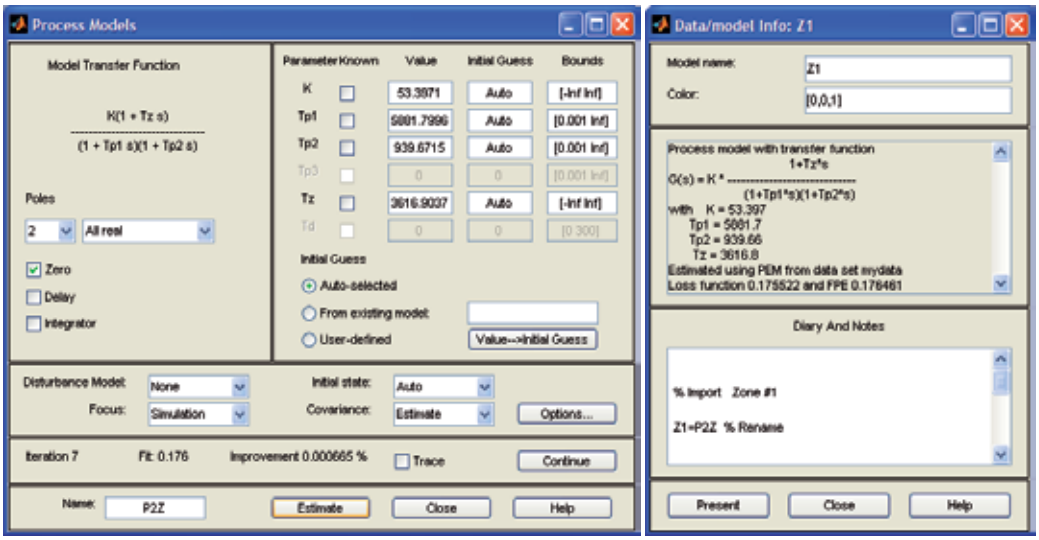


Figure 15. GUI Process Models menu and results of identification $S_i(\bar{x}_i, s)$ in Zone #1

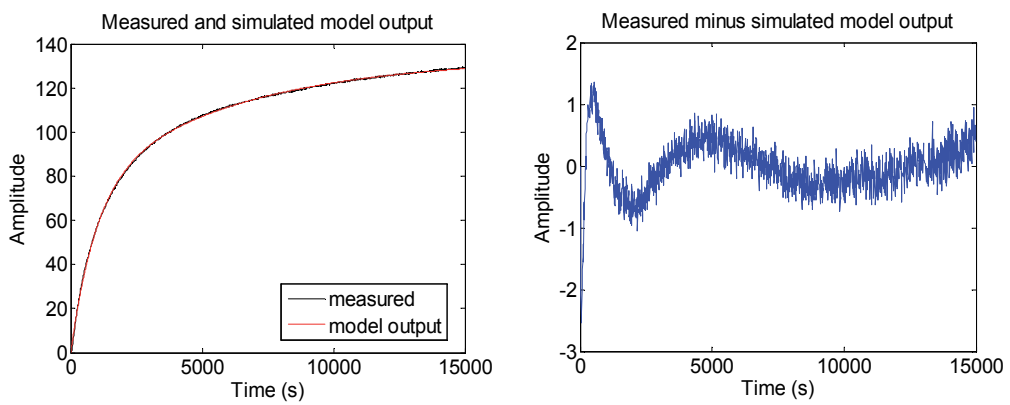


Figure 16. Comparison of measured and identified model output in Zone #1

6.2. Uncertainty analysis in the space and time domain

The proposed structure of the DPS control systems, Fig. 2, Fig. 5., are significant by decomposition of the control synthesis into the space and time subtasks. In the SS blocks, the approximation both of distributed controlled quantity $Y(\bar{x},k)$ and reference quantity $W(\bar{x},\infty)$, formulated as (8), (9) is solved. As the best approximation of controlled quantity $Y(\bar{x},k)$ vector of optimal approximation parameters $\{\breve{Y}_i(k)\}_i$ is obtained. Dynamics of these lumped quantities is different in compare with lumped quantities, $\{Y_i(\bar{x}_i,k)\}_i$, given by transfer functions $\{SH_i(\bar{x}_i,z)\}_i$ thus is created an uncertainty region in the time domain.

For uncertainty analysis, which takes place during the approximation problem solution, scheme from both, DPS Blockset blocks and Simulink blocks was arranged, see Fig. 17. There were obtained both, step responses $\{Y_i(\bar{x}_i,k)\}_i$ from each lumped input, $\{U_i\}_{i=1,5}$, to the corresponding output and approximated quantities corresponding to lumped output $\{\bar{Y}_i(k)\}_i$ from the block SS. Characteristics with uncertainty regions for actuating in each zone are depicted in Fig. 18.

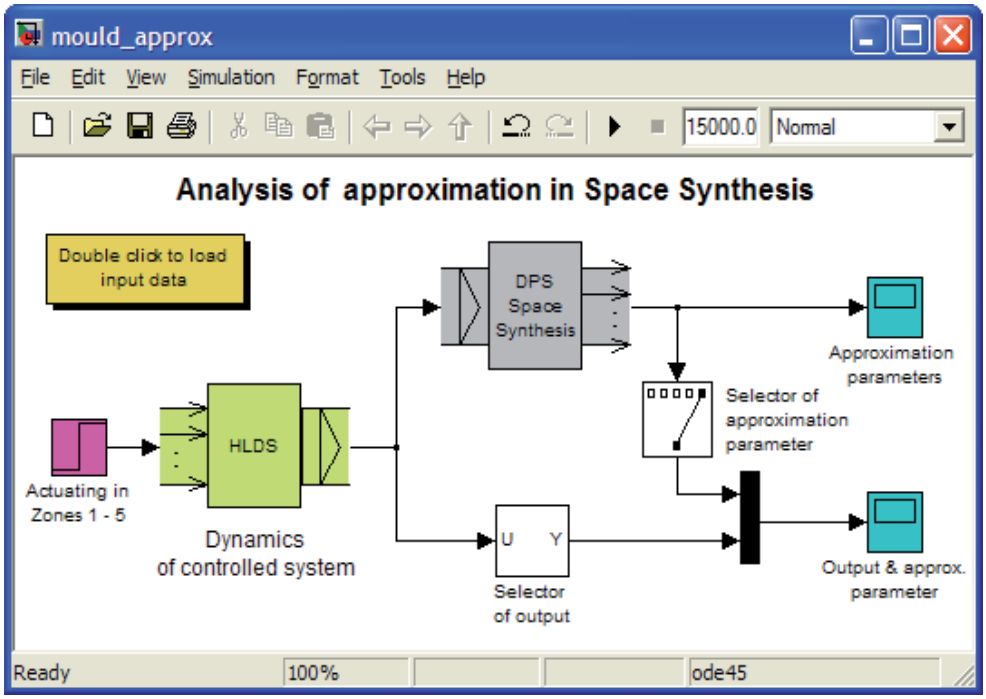


Figure 17. Block scheme in the MATLAB & Simulink for the analysis of approximation in the space synthesis and step responses in Zones 1 – 5

Identified transfer function $\{S_i(\bar{x}_i,s)\}_{i=1,5}$ in the structure (24) were also analysed in terms of uncertainties of their parameters. Analysis was performed in MATLAB environment, where functions from Robust Control Toolbox were used. Using functions *ureal* and *gridureal* were generated families of step responses with defined percentage variability of parameters in the nominal transfer functions $\{S_i(\bar{x}_i,s)\}_{i=1,5}$. Results for zone 1 are depicted in Fig. 19.

The gain variability K strongly affects to the value of the transient response in steady-state. The variability of the time constants Tp1 and Tp2 influences dynamics of transient response. Uncertainty region is caused by solution of the approximation task in the space synthesis. It would be appropriate to cover it by variability of nominal transfer function parameters. Cover of the uncertainty region in the time domain by variability of nominal parameters of the transfer function $S_i(\bar{x}_i,s)$ in Zone #1, #3 and #5 is presented in Fig. 20.

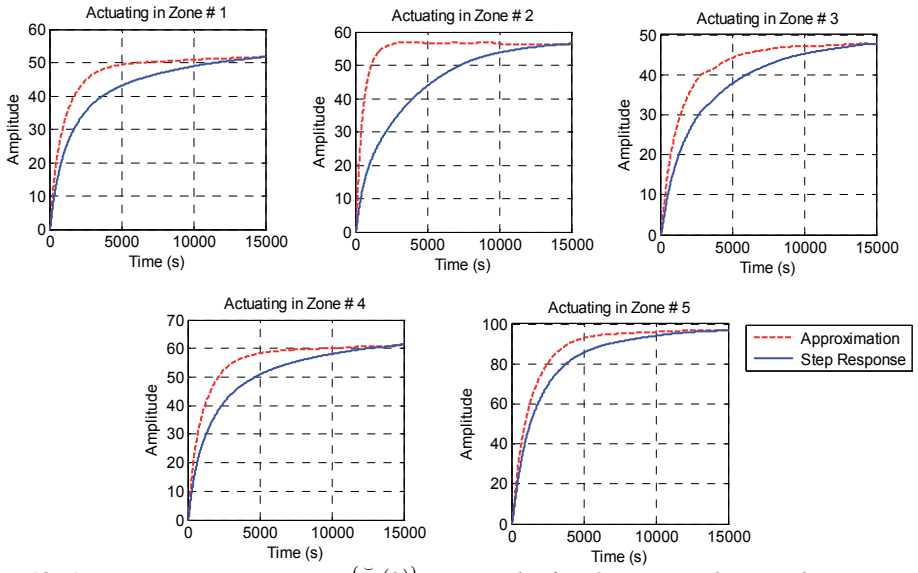


Figure 18. Approximation parameters $\{\breve{Y}_i(k)\}_i$ as a result of DPS space synthesis and step responses $\{\mathcal{H}H_i(\bar{x}_i,k)\}_i$ for actuating in Zones 1 – 5

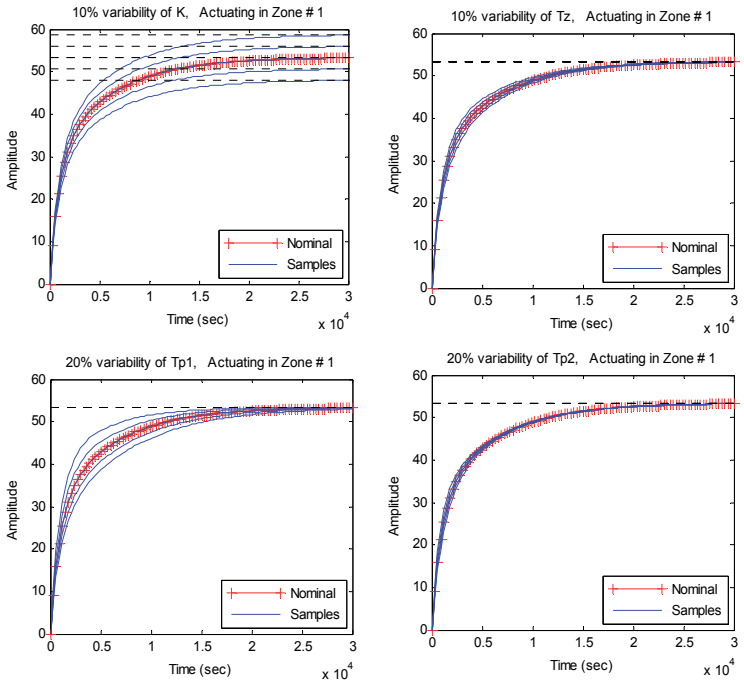


Figure 19. Family (Samples) of step responses for the defined percentage variability of parameters in the nominal transfer function $S_1(\bar{x}_i,s)$

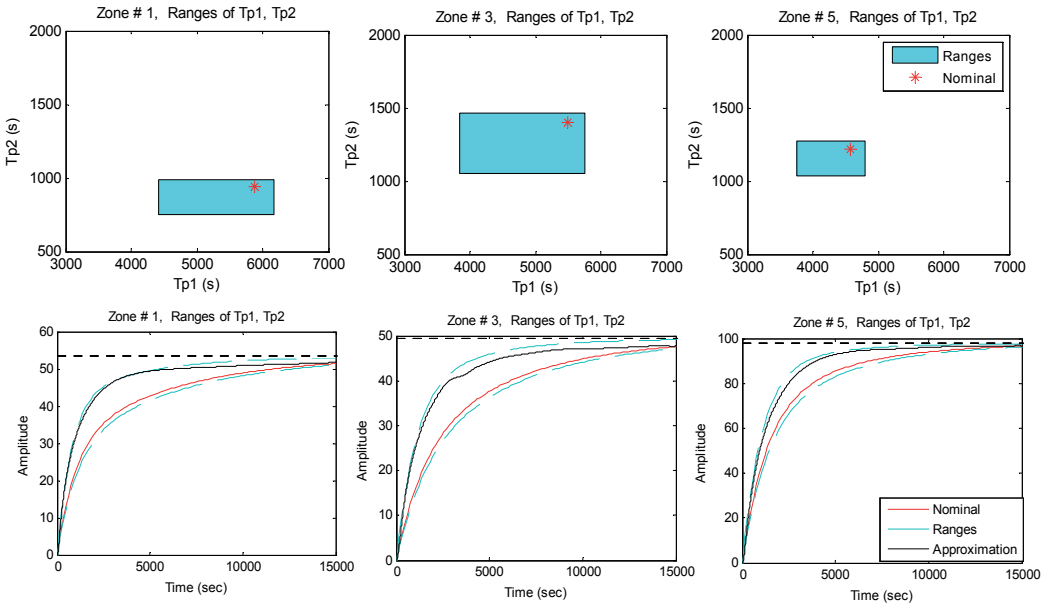


Figure 20. Cover of the uncertainty region in the time domain by variability of the nominal parameters Tp1 and Tp2 of the transfer function $S_i(\bar{x}_i, s)$ in Zone #1, #3 and #5

7. Robust control process

Robust control of temperature fields of the casting mould as distributed parameter system was performed with MATLAB & Simulink and DPS Blockset software support. Robust controllers designed by the IMC control strategy were first optimised through their tuning parameters $\{\alpha_i\}_i$ and then implemented to the control scheme for the real-time control of temperature fields of the casting mould in the benchmark casting plant.

7.1. Optimization of tuning parameters

In the MATLAB & Simulink environment, by means of the DPS Blockset, distributed parameter system of robust control, *mould_robust_DZPK.mdl* was arranged, see Fig. 21. It is DPS feedback control loop, where the *DPS Robust Control Synthesis* block includes both, time and spatial part of the control synthesis, see Fig. 22 a). In this case, the control system consists of five single parameter control loops, each for one zone of the mould, where discrete robust controllers based on IMC structure are used.

DPS Robust Control Synthesis contains two blocks named *DPS Space Synthesis*, where approximation of distributed controlled quantity $Y(\bar{x}, k)$ and reference quantity, $W(\bar{x}, \infty)$ is executed. The time control synthesis is performed by *Robust controllers based on IMC* block with five discrete controllers given by ZPK transfer function and filters with parameters $\{\alpha_i\}_{i=1,5}$, see Fig. 22 b).

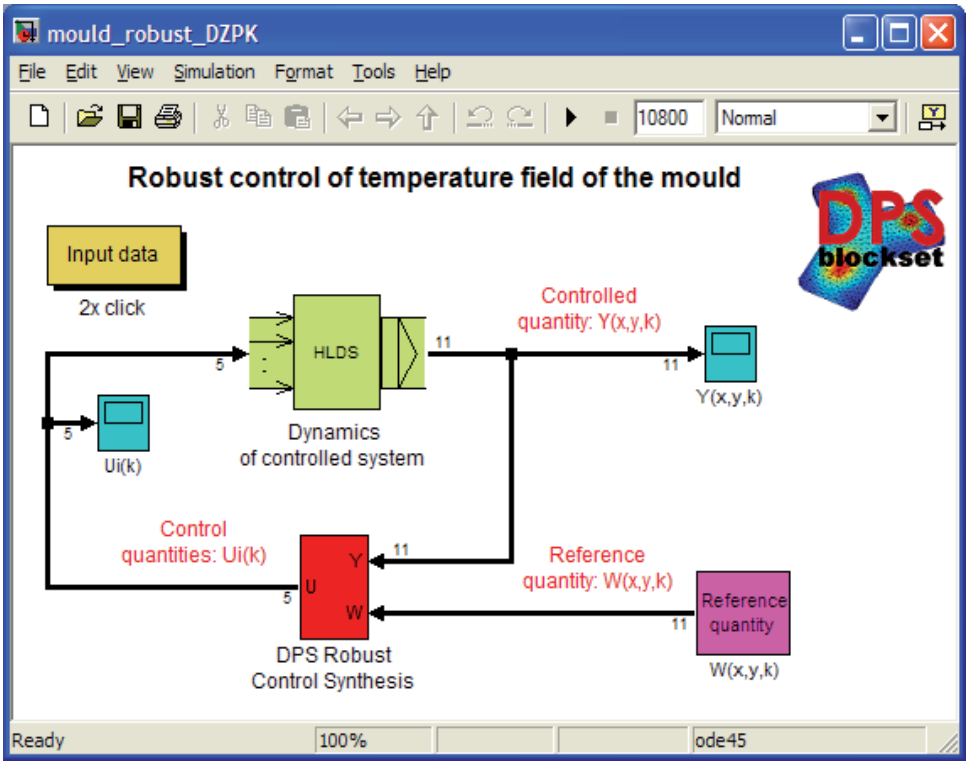


Figure 21. DPS feedback robust control system in the DPS Blockset for simulation of robust control of the temperature fields in the casting mould

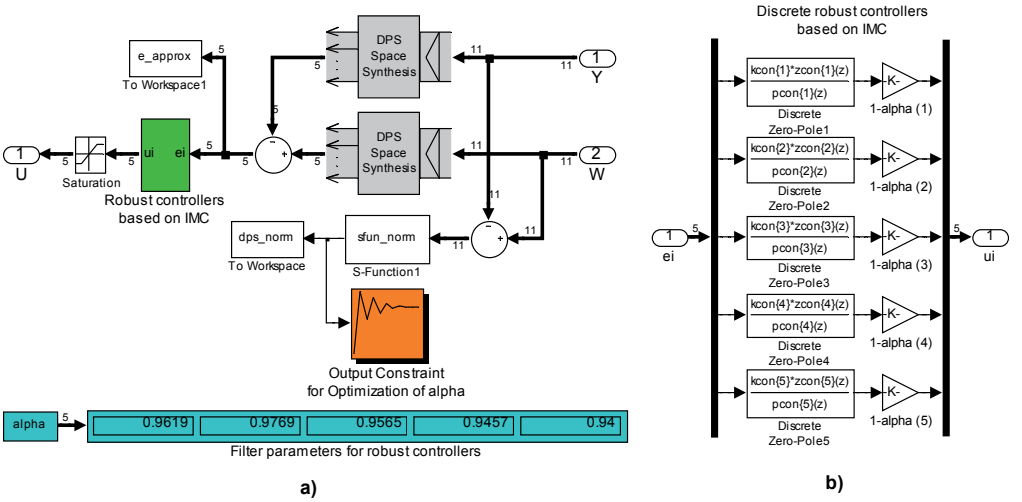


Figure 22. Structure of blocks in the scheme mould_robust_DZPK.mdl: a) DPS Robust Control Synthesis, b) Robust controllers based on IMC

The block *DPS Robust Control Synthesis* also contains block *Output Constraint for optimization of alpha*, where optimization of parameters $\{\alpha_i\}_{i=1,5}$ according to criterion function (25) is performed.

$$J = \min_{\alpha_i} \sum_{k=0}^{N} \left\| W(\bar{x},k) - Y(\bar{x},k) \right\| \tag{25}$$

Parameters of filters were optimized in the presence of constraints of the criterion function in order to assure nearly aperiodic course of the quadratic norm of the distributed control error with respect to the robust stability and robust performance conditions, see Fig. 23 and optimization progress presents Fig. 24. Control process was simulated for the reference quantities - temperatures on given 11 positions, where thermocouples are embedded. Robust stability was tested for ranges around the nominal parameters Tp1 and Tp2, of the transfer function $S_i(\bar{x}_i,s)$, see Fig. 20, with MATLAB function *robuststab*, from the Robust Control Toolbox, e.g. in the following is the printout of the robust stability testing for the control loop in zone 1.

Robust stability testing in Zone #1
StabilityMargin = UpperBound: 1.3333
LowerBound: 1.3296
DestabilizingFrequency: 8.0745e-004
DestbUnc = Tp1: 1.7210
Tp2: 438.5898
STABreport = Uncertain System is robustly stable to modeled uncertainty.
- It can tolerate up to 133% of the modeled uncertainty.

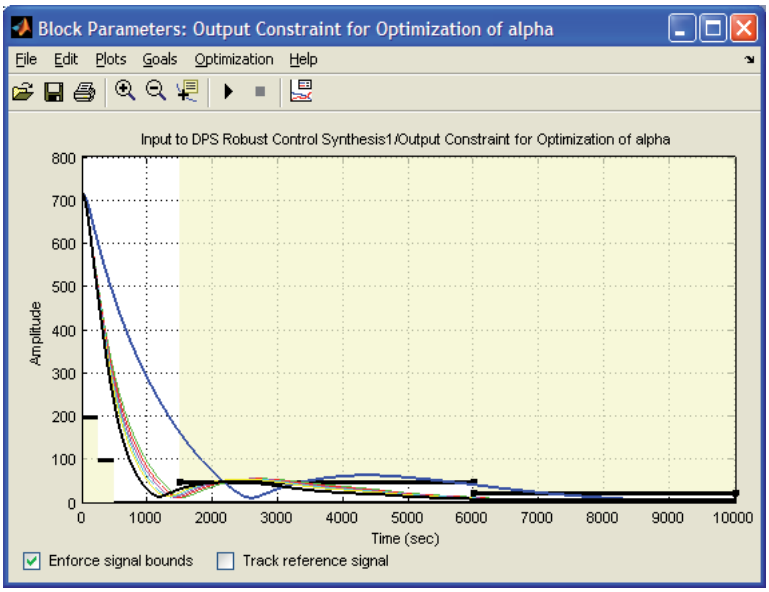
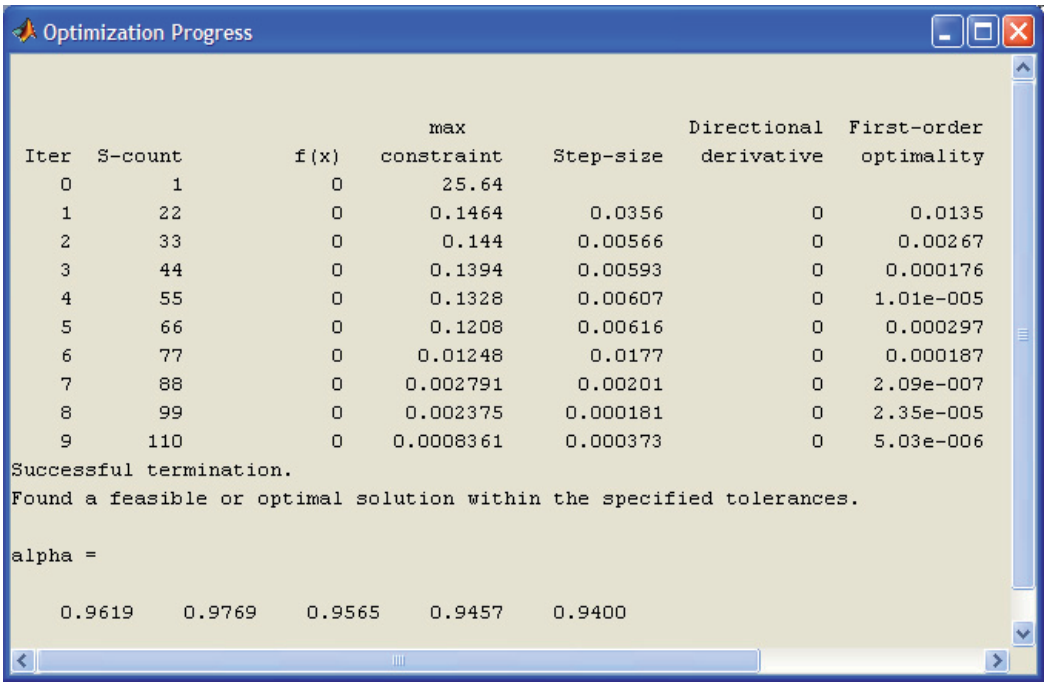


Figure 23. Minimization of the criterion function in presence of output constraints



Optimization Progress

Iter	S-count	f(x)	max constraint	Step-size	Directional derivative	First-order optimality
0	1	0	25.64			
1	22	0	0.1464	0.0356	0	0.0135
2	33	0	0.144	0.00566	0	0.00267
3	44	0	0.1394	0.00593	0	0.000176
4	55	0	0.1328	0.00607	0	1.01e-005
5	66	0	0.1208	0.00616	0	0.000297
6	77	0	0.01248	0.0177	0	0.000187
7	88	0	0.002791	0.00201	0	2.09e-007
8	99	0	0.002375	0.000181	0	2.35e-005
9	110	0	0.0008361	0.000373	0	5.03e-006

Successful termination.
Found a feasible or optimal solution within the specified tolerances.

alpha =

0.9619 0.9769 0.9565 0.9457 0.9400

Figure 24. Optimization progress for tuning parameters $\{\alpha_i\}_{i=1,5}$ of robust controllers

- A destabilizing combination of 133% of the modeled uncertainty exists, causing an instability at 0.000807 rad/s.
- Sensitivity with respect to uncertain element ...
  'Tp1' is 100%. Increasing 'Tp1' by 25% leads to a 25% decrease in the margin.
  'Tp2' is 54%. Increasing 'Tp2' by 25% leads to a 14% decrease in the margin.

7.2. Real-time robust control of preheating in the casting mould

Real-time robust control of the temperature fields of the casting mould in the benchmark casting plant was performed by means of Simulink designed block scheme *mould_exp_robust.mdl*. The structure of the block *DPS Robust IMC Control Synthesis* is depicted in Fig. 25, where are blocks *Space Synthesis Y* and *Space Synthesis W*, and block *Robust controllers based on IMC* with the same structure as in Fig. 22 b).

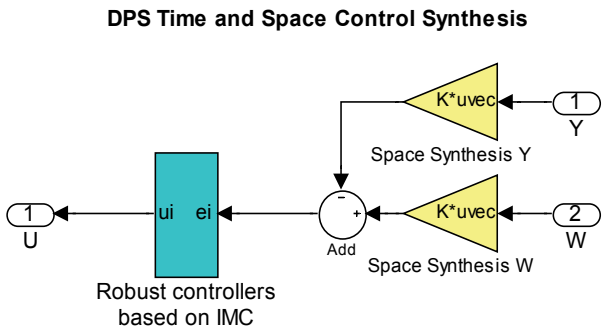


Figure 25. Structure of the block DPS Robust IMC Control Synthesis in the scheme mould_exp_robust.mdl for real-time control

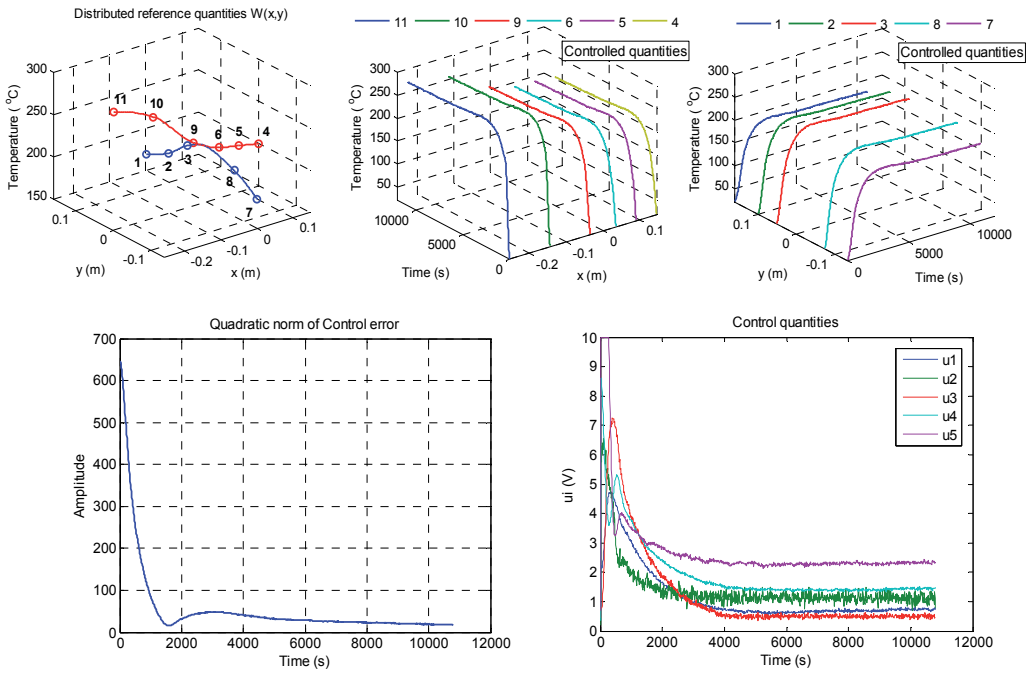


Figure 26. Robust control of the temperature field of the casting mould: distributed reference quantities, controlled quantities at given position of the mould, quadratic norm of distributed control error $\left\|E(\bar{x},k)\right\|$ and control quantities $\left\{U_i(k)\right\}_{i=1,5}$ in Zones 1-5

Control process was performed for the reference variable - temperature profile given in 11 positions, where thermocouples are embedded. Results of the real-time robust control process are on Fig. 26. The quality of control both in the time and space domain is given by the quadratic norm of the distributed control error.

8. Conclusion

The aim of this chapter was to present the engineering approach for the robust control of DPS, which opens a wide space for novel applications of the toolboxes and blocksets of the MATLAB & Simulink software environment. This approach is based on the general decomposition of controlled DPS dynamics, represented by transient and impulse characteristics, into time and space components. Starting out from this dynamics decomposition a methodical framework was presented for the decomposition of control synthesis into the space and time subtasks. In the space domain an approximation problems were solved, while in the time domain the control synthesis was performed by lumped parameter SISO control loops, where various well-known methods for design of controller is possible to utilize. The advantage of this approach is the relatively simple LDS model of DPS, which is directly suitable for control purposes and can be easily identified from input-output data by means of classical techniques.

Currently, it is interesting to formulate and solve tasks of control in various engineering branches, including the casting technology, by means of methods and tools of distributed parameter systems. Methodical approach presented in this chapter demonstrates simple possibilities, how to exploit the distributed dynamical characteristics on complex definition domains, obtained by evaluation of measured data for robust IMC control synthesis of DPS with respect to uncertainty of models and the real-time control according to technological requirements.

Author details

Cyril Belavý, Gabriel Hulkó and Karol Ondrejkovič
Institute of Automation, Measurement and Applied Informatics,
Faculty of Mechanical Engineering, Center for Control of Distributed Parameter Systems,
Slovak University of Technology in Bratislava, Slovak Republic

Acknowledgement

This work was supported by the Slovak Scientific Grant Agency VEGA under the contract No. 1/0138/11 and the Slovak Research and Development Agency under contracts No. APVV-0131-10 and No. APVV-0090-10 and also by the European Union with the co-financing of the European Social Fund, grant TAMOP-4.2.1.B-11/2/KMR-2011-0001.

9. References

Belavý, C. et al. (2009). Uncertainty Analysis and Robust Control of Temperature Fields of Casting Mould as Distributed Parameter Systems, *INCOM ´09: Preprints of the 13th IFAC symposium on Information control problems in manufacturing*, Moscow, June 3-5, 2009

Butkovskij, A. G. (1965). *Optimal Control of Distributed Parameter Systems,* Nauka, Moscow

Curtain, R. & Morris K. (2009). Transfer Functions of Distributed Parameter Systems: A tutorial. *Automatica,* Vol. 45, (2009), pp. 1101-1116

Hulkó, G. et al. (1981). On Adaptive Control of Distributed Parameter Systems, *Proceedings of 8-th World Congress of IFAC*, Kyoto, 1981

Hulkó, G. et al. (1987). Control of Distributed Parameter Systems by means of Multi-Input and Multi-Distributed-Output Systems, *Proceedings of 10-th World Congress of IFAC*, Munich, 1987

Hulkó, G. et al. (1998). *Modeling, Control and Design of Distributed Parameter Systems with Demonstrations in MATLAB,* Publishing House STU, ISBN 80-227-1083-0, Bratislava

Hulkó, G. et al (2003-2007). Distributed Parameter Systems. Web portal [Online]. Available: www.dpscontrol.sk

Hulkó, G. et al. (2009a). Engineering Methods and Software Support for Modelling and Design of Discrete-time Control of Distributed Parameter Systems. *European Journal of Control,* Vol. 15, No. Iss. 3-4, *Fundamental Issues in Control,* (May-August 2009), pp. 407-417, ISSN 0947-3580

Hulkó, G. et al. (2009b). Engineering Methods and Software Support for Control of Distributed Parameter Systems, *ASC 2009: Proceedings of the 7th Asian Control Conference,* Honk Kong, China, August 27-29, 2009

Hulkó, G. et al. (2003-2010). *Distributed Parameter Systems Blockset for MATLAB & Simulink*, www.mathworks.com/products/connections/ - Third-Party Product of The MathWorks, Bratislava-Natick, Available from: www.dpscontrol.sk

Morari, M. & Zafirou, E. (1989). *Robust Process Control,* Prentice Hall, Englewood Cliffs

Lasiecka, I. & Triggiani, R. (2000). *Control Theory for Partial Differential Equations: Continuous and Approximation Theories I. Abstract Parabolic Systems,* Cambridge University Press, ISBN 0-521-43408-4, Cambridge, UK

Li, H. X. & Qi, Ch. (2010). Modeling of Distributed Parameter Systems for Application – A Synthesized Review from Time-Space Separation. *Journal of Process Control*, No 20, (2010), pp. 891-901

Lions, J. L. (1971). *Optimal Control of Systems Governed by Partial Differential Equations,* Springer - Verlag, Berlin - Heidelberg - New York

Wang, P. K. C. (1964). *Control of Distributed Parameter Systems* (Advances in Control Systems: Theory and Applications, 1.), Academic Press, New York

Chapter 3

Fouling in Heat Exchangers

Hassan Al-Haj Ibrahim

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46462

1. Introduction

Fouling is generally defined as the deposition and accumulation of unwanted materials such as scale, algae, suspended solids and insoluble salts on the internal or external surfaces of processing equipment including boilers and heat exchangers (Fig 1). Heat exchangers are process equipment in which heat is continuously or semi-continuously transferred from a hot to a cold fluid directly or indirectly through a heat transfer surface that separates the two fluids. Heat exchangers consist primarily of bundles of pipes, tubes or plate coils.

Figure 1. Fouling of heat exchangers.

Fouling on process equipment surfaces can have a significant, negative impact on the operational efficiency of the unit. On most industries today, a major economic drain may be caused by fouling. The total fouling related costs for major industrialised nations is estimated to exceed US$4.4 milliard annually. One estimate puts the losses due to fouling of heat exchangers in industrialised nations to be about 0.25% to 30% of their GDP [1, 2]. According to Pritchard and Thackery (Harwell Laboratories), about 15% of the maintenance costs of a process plant can be attributed to heat exchangers and boilers, and of this, half is probably caused by fouling. Costs associated with heat exchanger fouling include production losses due to efficiency deterioration and to loss of production during planned or unplanned shutdowns due to fouling, and maintenance costs resulting from the removal of fouling deposits with chemicals and/or mechanical antifouling devices or the replacement of corroded or plugged equipment. Typically, cleaning costs are in the range of $40,000 to $50,000 per heat exchanger per cleaning.

Fouling in heat exchangers is not a new problem. In fact, fouling has been recognised for a long time, and research on heat exchanger fouling was conducted as early as 1910 and the first practical application of this research was implemented in the 1920's. Technological progress in prevention, mitigation and removal techniques in industrial fouling was investigated in a study conducted at the Battelle Pacific Northwest Laboratories for the U.S. Department of Energy. Two hundred and thirty one patents relevant to fouling were analysed [3]. Furthermore, great technical advance in the design and manufacture of heat exchangers has in the meantime been achieved. Nonetheless, heat exchanger fouling remains today one of the major unresolved problems in Thermal Science, and prevention or mitigation of the fouling problem is still an ongoing process. Further research on the problem of fouling in heat exchangers and practical methods for predicting the fouling factor, making use in particular of modern digital techniques, are still called for. One significant and clear indication of the relevance and urgency of the problem may be seen in the current international patent activity on fouling (Table 1).

Country	No. of Patents	% of Patents
U.S.A.	147	63.6
Germany	22	9.5
Japan	21	9.1
Sweden	9	3.9
Switzerland	8	3.5
Other	24	10.4
Total	231	100.0

Table 1. International Patent Activity [4]

Major detrimental effects of fouling include loss of heat transfer as indicated by charge outlet temperature decrease and pressure drop increase. Other detrimental effects of fouling may also include blocked process pipes, under-deposit corrosion and pollution. Where the

heat flux is high, as in steam generators, fouling can lead to local hot spots resulting ultimately in mechanical failure of the heat transfer surface. Such effects lead in most cases to production losses and increased maintenance costs.

Loss of heat transfer and subsequent charge outlet temperature decrease is a result of the low thermal conductivity of the fouling layer or layers which is generally lower than the thermal conductivity of the fluids or conduction wall. As a result of this lower thermal conductivity, the overall thermal resistance to heat transfer is increased and the effectiveness and thermal efficiency of heat exchangers are reduced. A simple way to monitor a heat transfer system is to plot the outlet temperature versus time. In one unit at an oil refinery, in Homs, Syria, fouling led to a feed temperature decrease from 210°C to 170°C. In order to bring the feed to the required temperature, the heat duty of the furnace may have to be increased with additional fuel required and resulting increased fuel cost. Alternatively, the heat exchanger surface area may have to be increased with consequent additional installation and maintenance costs. The required excess surface area may vary between 10-50%, with an average around 35%, and the additional extra costs involved may add up to a staggering 2.5 to 3.0 times the initial purchase price of the heat exchangers.

With the onset of fouling and the consequent build up of fouling layer or layers, the cross sectional area of tubes or flow channels is reduced. In addition, increased surface roughness due to fouling will increase frictional resistance to flow. Such effects inevitably lead to an increase in the pressure drop across the heat exchanger, which is required to maintain the flow rate through the exchanger, and may even lead to flow blocks. Experience with pressure drop monitoring has shown, however, that it is not usually as sensitive an indicator of the early onset of fouling when compared to heat transfer data; thus pressure drop is not commonly used for crude preheat monitoring. In situations where significant swings in flow rates are experienced, flow correction can be applied to both pressure drop and to heat transfer calculations to normalise the data to a standard flow.

Different fouling deposit structures can lead to under-deposit corrosion of the substrate material such as localised fouling, deposit tubercles and sludge piles. The factors that are most likely to influence the probability of under-deposit corrosion include deposit composition and its porosity and permeability. Even minor components of the deposits can sometimes cause severe corrosion of the underlying metal such as the hot corrosion caused by vanadium in the deposits of fired boilers [5].

Fouling is responsible for the emission of many millions of tonnes of carbon dioxide as well as the use and disposal of hazardous cleaning chemicals. Data from oil refineries suggest that crude oil fouling accounts for about 10% of the total CO_2 emission of these plants. Wastes generated from the cleaning of heat exchangers may contain hazardous wastes such as lead and chromium, although some refineries which do not produce leaded gasoline and which use non-chrome corrosion inhibitors typically do not generate sludge that contains these constituents. Oily wastewater is also generated during heat exchanger cleaning.

The factors that govern fouling in heat exchangers are many and varied. Of such factors some may be related to the feed properties such as its chemical nature, density, viscosity, diffusivity, pour and cloud points, interfacial properties and colloidal stability factors. The chemical nature of the feed in particular can be an important factor affecting to a large degree the rate and extent of fouling. This includes the chemical composition of the feed and the stability of its components and their compatibility with one another and with heat exchanger surfaces as well as the presence in the feed of unsaturated and unstable compounds, inorganic salts and trace elements such as sulphur, nitrogen and oxygen. The feed storage conditions and its exposure to oxygen on storage in particular can in most cases also affect materially the rate and nature of fouling.

Other factors of equal importance to the feed properties may be related to operating conditions and equipment design, such as feed temperature, bulk fluid velocity or flow rate, heat exchanger geometry, nature of alloy used and wettability of surfaces where fouling occurs. The rate of fouling is feed temperature dependent with different rates of fouling between the feed inlet and outlet sides of the heat exchanger. In a shell and tube heat exchanger, the conventional segment baffle geometry is largely responsible for higher fouling rates. Uneven velocity profiles, back-flows and eddies generated on the shell side of a segmentally-baffled heat exchanger results in higher fouling and shorter run lengths between periodic cleaning and maintenance of tube bundles.

All these and other factors that may affect fouling need to be considered and taken into account in order to be able to prevent fouling if possible or to predict the rate of fouling or fouling factor prior to taking the necessary steps for fouling mitigation, control and removal.

2. Fouling mechanisms and stages

Fouling can be divided into a number of distinctively different mechanisms. Generally speaking, several of these fouling mechanisms occur at the same time and each requires a different prevention technique. Of these different mechanisms some represent different stages in the process of fouling. The chief fouling mechanisms or stages include:

1. Initiation or delay period. This is the clean surface period before dirt accumulation. The accumulation of relatively small amounts of deposit can even lead to improved heat transfer, relative to clean surface, and give an appearance of "negative" fouling rate and negative total fouling amount.
2. Particulate fouling and particle formation, aggregation and flocculation.
3. Mass transport and migration of foulants to the fouling sites.
4. Phase separation and deposition involving nucleation or initiation of fouling sites and attachment leading to deposit formation.
5. Growth, aging and hardening and the increase of deposits strength or auto-retardation, erosion and removal.

Detailed analysis of deposits from the heat exchanger may provide an excellent clue to fouling mechanisms. It can be used to identify and provide valuable information about such mechanisms. The deposits consist primarily of organic material that is predominantly asphaltenic in nature, with some inorganic deposits, mainly iron salts such as iron sulphide. The inorganic content of the deposits is relatively consistent in most cases at 22-26% [6].

Deposit analysis is performed by taking a sample and extracting any degraded hydrocarbon oil by using a solvent, such as methyl chloride, that is effective at removing hydrocarbon oils and low molecular weight polymers that may have been trapped in the deposit.

The remaining material from this extraction will consist of any organic polymers, coke, and inorganic components. The basic analysis of the non-extractable material involves ashing in which organic and volatile inorganic compounds are lost. By this means, volatile inorganics such as chlorides and sulphur compounds which are lost on ashing, may be determined. The detection of iron sulphide or other volatile inorganic materials determines the cause of inorganic fouling. These values can be compared throughout the exchanger train [6]. The non-volatile material or ash will include all oxidised metallic salt–type materials or corrosion products. The presence of iron in the ash may indicate corrosion in tankage in an upstream unit or in the exchanger train itself. This basic analysis indicates if the deposits are primarily organic or inorganic.

Special techniques and tools such as the use of optical microscopy and solubility in solvents may be used for the analysis of the non-extractable material. Infrared analysis can identify various functional groups present in the deposit which may include nitrogen, carbonyls, and unsaturated paraffinic or aromatic compounds which are polymerisation precursors, identified in feed stream characterisation [6]. The carbon and hydrogen content of the non-extractable deposit can be determined by elemental analysis. If the carbon to hydrogen ratio is very high, it may indicate that the majority of the organic portion of the deposit is coke. The coke may have been particles entrained in the stream or material which has been thermally dehydrogenated in the heat exchangers. The carbon to hydrogen ratio also indicates whether the deposit is more paraffinic or aromatic. This information helps identify the polymers formed [6].

In Table 2 analytical results are shown from deposits obtained from the four chain feed/effluent heat exchangers in which the hot product effluent is used for pre-heating the cold naphtha feedstock for a naphtha hydrotreater plant at the Homs Oil Refinery [7]. This plant is one of the most important units at the Homs Refinery, with an annual capacity of 480,000 tons/yr. It is used to remove impurities such as sulphur, nitrogen, oxygen, halides and trace metal impurities that may deactivate reforming catalysts. Furthermore, the quality of the naphtha fractions is also upgraded by reducing potential gum formation as a result of the conversion of olefins and diolefins into paraffins. The process utilises a catalyst (Hydrobon) in the presence of substantial amounts of hydrogen under high pressures (50 bars) and temperatures (320°C) (Fig. 2). A major fouling problem was encountered early on in the heat exchangers, indicated by an increased pressure drop, decreased flow rate and lower temperatures at the heat exchangers outlet.

Particulate fouling

Particulate fouling, which is the most common form of fouling, can be defined as the process in which particles in the process stream deposit onto heat exchanger surfaces. These particles include particles originally carried by the feed stream before entering the heat exchanger and particles formed in the heat exchanger itself as a result of various reactions, aggregation and flocculation. Particulate fouling increases with particle concentration, and typically particles greater than 1 ppm lead to significant fouling problems.

Heat exchanger	A	B1	B2	C	D
Loss at 105°C (wt %)	1.17	1.03	1.05	1.15	1.14
Loss at 550°C (wt %)	79.70	95.10	90.17	94.42	57.17
Loss at 840°C (wt %)	80.00	95.29	90.19	94.48	57.99
Ash (wt %)	20.00	4.71	9.81	5.52	42.01
Chloride (wt %)	170	435	0	664	508
Sulphur (wt %)	17.00	13.50	13.80	10.20	13.00
Ammonium (ppm)	42	1184	43	134	4969
Iron (wt % of ash)	19.30	2.83	1.70	2.80	15.58
Sodium (ppm of ash)	1473	1047	825	3301	914
Calcium (ppm of ash)	459	179	78	377	1431
Magnesium(ppm of ash)	90	41	19	102	1341
Chromium (ppm of ash)	231	107	1166	196	1096
Copper (ppm of ash)	511	319	74	443	126
Nickel (ppm of ash)	378	129	63	90	52

Table 2. Analysis of deposits on heat exchanger surfaces [7].

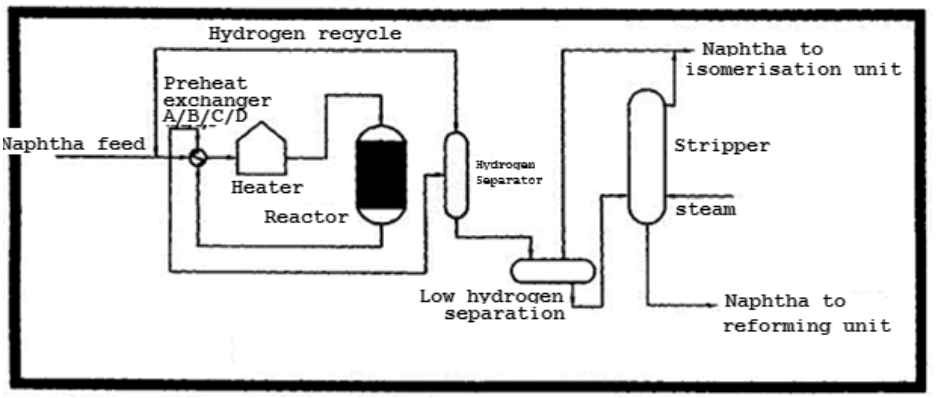


Figure 2. Naphtha hydrotreating unit

2.1. Particles in the feed stream

Particles in the fluid feed stream are solid particles which are entrained or contained in the feed stream before entering the heat exchanger and which can settle out upon the heat

exchanger surfaces. These solid particles are for the most part insoluble inorganic particles such as corrosion products (iron sulphide and rust), catalyst particles or fines, dirt, silt and sand particles, and other inorganic salts such as sodium chloride, calcium chloride and magnesium chloride. The feed streams may also contain some organic particles that may have been formed during their storage or transport.

Many streams including cooling water and other product streams from different units or plants may contain solid particles. In particular, streams from such oil refinery units as vacuum units, visbreakers, and cokers may have more particulates and metals than straight-run products due to the heavier nature of the feeds processed. Streams can also be purchased from other refiners. Due to the increased transit time and exposure to oxygen before being fed to the unit these feeds may have higher particulate levels as a result of polymerisation reactions and corrosion [6].

Particles in the fluid stream, regardless of whether they are organic or inorganic in nature, fall in general into tow classes: basic sediment and filterable solids.

Typically, particles in the fluid stream greater than 1 ptb (pounds per thousand barrels) lead to significant fouling problems in the unit. Their effect on fouling can be avoided however if these particles are removed by solid-liquid filtration, sedimentation, centrifugation or by any of various fluid cleaning devices. The only particles that need to be considered in this regard are those that are not filterable or those particles that are left to proceed to the heat exchanger.

The amount of filterable solids in the stream, reported in ptb or wt% (weight percent), may be determined by filtration of the unit feed. Filterable solids analysis can evaluate a stream deposition potential by indicating the type of materials that could contribute to fouling if allowed to pass through to the heat exchanger.

Table 3 shows the analysis of filterable solids in the naphtha feed stream to the heat exchangers of the hydrotreater unit at the Homs oil refinery. The feedstock for this unit is a blend of light and heavy straight-run naphtha fractions from four different topping units. The resulting blend is left in a blending tank for a sufficient period of time to allow for equilibrium conditions to be established [8]. To evaluate the quantity of particulate solids which are entrained with the naphtha stream before entering the heat exchangers, a number of samples of the naphtha feed were filtered and the amount of entrained particles determined. Two samples of the filterable solids were taken, one sample was taken from the feed entering a macrofilter on the unit boundary and the other from a second macrofilter on the feed pump suction. The nature of the materials entrained was then determined by ashing and analysing these two samples (Table 3). The size distribution of the filterable solid particles was also determined (Table 4).

Examination of the deposit analysis for heat exchanger D (Table 2), where the deposits are a mixture of inorganic (42%) and organic (58%) deposits, indicate particulate and polymerisation fouling. The nature of particulate fouling in D is confirmed by the variation

of fouling factor with time, with no induction time or delay period indicated (Fig. 3). The fouling factor curve is linear with saw-tooth shape, where both the fouling factor and the deposition rate increase with time. This means continuous build up of the fouling layer followed by break off periods [9].

2.2. Particle formation

Chemical particle formation is the basic mechanism of particle formation in heat exchangers fluid streams, although organic material growth and biological particle formation, or biofouling, may occur in sea water systems and in types of waste treatment systems. Biofouling may be of two kinds: microbial fouling, due to microorganisms (bacteria, algae, and fungi) and their products, and macrobial fouling, due to the growth of macroorganisms such as barnacles, sponges, seaweeds or mussels. On contact with heat-transfer surfaces, these organisms can attach and breed, reducing thereby both flow and heat transfer to an absolute minimum and sometimes completely clogging the fluid passages. Such organisms may also entrap silt or other suspended solids and give rise to deposit corrosion. Corrosion due to biological attachment to heat transfer surfaces is known as microbiologically influenced corrosion. For open recirculating systems, bacteria concentrations of the order of 1×10^5 cells/ml and fungi of 1×10^3 cells/ml may be regarded as limiting values [10].

	Feed filter	Pump filter
Loss at 105°C (wt. %)	10.0	0.1
Loss at 550°C (wt. %)	28.3	25.3
Loss at 840°C (wt. %)	30.4	26.7
Ash (wt. %)	69.5	73.2
Carbon (wt. %)	2.6	6.4
Sulphur (wt. %)	36.9	19.7
Sulphates (wt. %)	55.8	50.7
Chloride (ppm)	-	281
Ammonium (ppm)	-	52
Iron (wt. % of ash)	45	58
Sodium (ppm of ash)	-	9
Calcium (ppm of ash)	-	161

Table 3. Analysis of two samples of the filterable solids.

Mesh size (μm)	< 90	90	125	355
Particle distribution (%)	24	8	36	32

Table 4. Size distribution of the filterable solid particles

Chemical particle formation can be the result of either corrosion or decomposition and polymerisation reactions. Trace contaminants present in the fluid stream can have a

significant effect on the fouling encountered in certain chemical processes. Such contaminants may include oxygen, nitrogen, NH_3, H_2S, CN, HCN, Hg, unsaturates, organic sulphides and chlorides, and heavy hydrocarbon compounds such as paraffin wax, resins, asphaltenes, and organometallic compounds. Individual metals, which may exist as metal salts in the feed stream, can catalyse different polymerisation reactions. The concentrations of such metals are typically very low, not exceeding few ppms. However, small concentrations of certain metals can have a significant effect on catalysing different fouling-related polymerisation reactions. Metal detectors on unit feed samples can detect individual metals in the stream at less than 1 ppm.

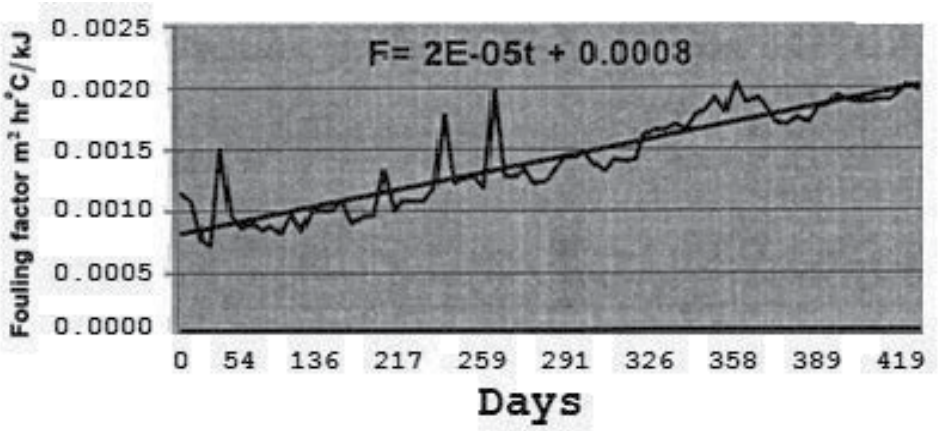


Figure 3. Variation of fouling factor in exchanger D in 2001.

Corrosion fouling is fouling deposit formation as a result of the corrosion of the substrate metal of heat transfer surfaces. This type of corrosion should not be confused, however, with the under-deposit corrosion, referred to earlier, which is one of the aftereffects of fouling.

Corrosion fouling is a mechanism which is dependent on several factors such as thermal resistance, surface roughness and composition of the substrate and fluid stream. In particular, impurities present in the fluid stream can greatly contribute to the onset of corrosion. Such impurities include hydrogen sulphide, ammonia and hydrogen chloride. In crude oil, for example, sulphur and nitrogen compounds are two very common contaminants which are mostly decomposed in certain situations to hydrogen sulphide and ammonia respectively. Chlorides which may be found in oil streams are converted to hydrogen chloride by the following reaction.

$$R\text{-}Cl + H_2 \rightarrow HCl + R$$

The chlorides may enter the refinery as salt with the crude. Chlorides in the oil stream may also be derived from various chemicals used in the oil industry which can contain high levels of chloride. Such chemicals include tertiary oil recovery enhancement chemicals and solvents used to clean tankers, barges, trucks and pipelines. As the crude oil is processed,

some of these chemicals and solvents, which are thermally stable and not soluble in water, pass overhead in the main tower of the atmospheric distillation unit along with the naphtha.

In the hydrotreater feed stream, chloride levels as high as 50 wt. ppm have been reported. High levels of chloride were detected with the filterable solids in the naphtha feed stream to the heat exchangers of the hydrotreater unit at the Homs refinery (Table 3) and in the deposits obtained from the heat exchangers (Table 2). Furthermore, the makeup hydrogen from the platforming unit will always contain trace quantities of hydrogen chloride. In order to maintain catalyst performance, modern platforming catalysts require a small, but continuous dosage of chloride, some of which is always stripped and leaves the platforming unit in the net gas stream that supplies the hydrotreater with makeup hydrogen.

In a hydrogen sulphide environment the sulphur reacts with the exposed iron to form iron sulphide compounds. This happens in both the hot and cooler sections of the unit. The sulphur effectively corrodes the plant. However, once reacted, the iron sulphide forms a complex protective scale or lattice on the base metal, which inhibits further corrosion. The sulphide lattice would remain in equilibrium with its surroundings and the corrosion rate would be minimal if no other impurities were present in the system. The presence of other impurities, however, can accelerate corrosion as these impurities interact with the sulphide lattice.

Of the impurities that contribute to corrosion and fouling, hydrogen chloride may be the most important. By itself hydrogen chloride does not cause a problem. It will not foul equipment or corrode the carbon steel in the unit. Chloride corrosion and fouling, however, take place when hydrogen chloride, ammonia, and water all interact in the colder sections of the unit to defeat the protective sulphide lattice. The extent of the damage depends on their concentration and is directly dependent on pH, with the corrosion rate increasing rapidly with pH decrease.

Hydrogen chloride will become corrosive when it comes in contact with free water, i.e. water that is not in the vapour phase or is not saturated in the liquid hydrocarbon. Oil products are almost always saturated with water, and entrained water, even if it is less of a problem, does occur in most cases. Furthermore, continuous water wash at key locations is recommended as part of the solution to minimise the effects of chloride corrosion and fouling and this further contributes to the total water in the system.

Hydrogen chloride is highly soluble in water, and in a free water environment, any hydrogen chloride present in the vapour or hydrocarbon liquid will be quickly absorbed by the water, thus driving the pH down to approximately 1.

If the iron sulphide lattice is intact this chloride competes with the bisulphate ion (SH^-) for the iron ions in the lattice:

$$\text{S-Fe-S-Fe-SH} + Cl^- \rightleftarrows \text{Fe-S-S-Fe-Cl} + SH^-$$

With a high concentration of hydrogen chloride present the reaction shifts to the right. As more and more bisulphate is released from the sulphide lattice, it eventually dissolves

leaving the base metal exposed. The reaction rate is then only limited by the chloride ion concentration in the solution at low pH. Loss of wall metal takes place very rapidly.

In water the chloride ions react directly with any exposed iron to form $FeCl_2$.

$$Fe^{++} + 2Cl \rightarrow FeCl_2$$

$$2e^- + 2H^+ \rightarrow H_2$$

As the chloride concentration in water is reduced by removing the source, diluting with additional water or neutralising with a base, the pH will increase. Hydrogen sulphide will begin to react with the exposed iron and start building a new protective layer. This sulphide lattice gets stronger as the pH increases to 6 and above. The corrosion rate falls off to a minimum.

Hydrogen chloride will also cause serious fouling problems if ammonia is present in the system. The ammonia reacts with hydrogen chloride to form ammonium chloride which may cause fouling and plugging problems. In the cooler parts of the unit, the ammonium chloride will condense from the vapour phase and solidify and deposit directly and accumulate on the walls. The salt can also break away from the walls and be carried downstream to eventually deposit somewhere else. If free water is present, ammonium chloride will be absorbed directly from the vapour phase into the water and no solid salts will form on the equipment. Another problem associated with ammonium chloride salt deposits is under deposit pitting corrosion as the hygroscopic nature of the salt will result in a wet environment at the wall under the deposit. The chloride ions will react with the iron to form iron chloride causing serious localised corrosion, the reaction rate accelerating in the presence of hydrogen sulphide. The sulphide ion as part of an ammonium sulphide salt will react with the iron chloride to form iron sulphide, thus releasing the chloride ion to start over.

Any excess ammonia available may react with the disulphide ions present in the solution to form ammonium sulphide salts, but only after most of the chloride has been neutralised. While hydrogen sulphide is only slightly soluble in the water, its salt is highly soluble. Therefore, as the pH is raised to 6 and higher the free ammonia present reacts with the small quantity of hydrogen sulphide in solution, making more ammonium sulphide salts. The rich hydrogen sulphide vapour above the water will continuously replace the consumed hydrogen sulphide. The overall sulphide concentration in the water increases making it difficult to raise the pH much further.

The sharp increase in corrosion rate in the 6.8 to 7.3 pH range is related to the concentration of the ammonium chloride and sulphide salts present. In large quantities these salts can become aggressive, especially the sulphide salts. If the pH is raised further the corrosion rate again falls off to a very low value. This is because the sulphide lattice has formed into a very strong hard film that cannot easily be broken.

The iron content in the deposits obtained from the heat exchangers may be an indication of fouling by corrosion. Although polymerisation may account for about 80% of the total

fouling associated with the "A" heat exchanger, in the Homs hydrotreating plant, fouling by corrosion is not negligible, with about 19% of the total fouling may be due to corrosion, as is clearly indicated by the iron content of the deposits obtained from this exchanger (Table 2).

Coking and Polymerisation are major causes of fouling in heat exchangers. Decomposition of organic products can lead to the formation of very viscous tar or solid coke particles at high temperatures and polymerisation involves the formation of undesirable organic sediments or polymers. The coke particles and polymers formed in the heat exchanger may grow to such a large size that they drop out of solution and deposit on the process equipment. Such deposits can be extremely tenacious and may require burning off the deposit to return the heat exchanger to satisfactory operation.

There are two major polymerisation mechanisms which can occur in the feed stream: free radical and non-free radical polymerisation.

Free radical polymerisation occurs when a free radical is formed and continues to react with other molecules. The free radicals continue to propagate in the feed stream producing longer chain polymers which will continue to be produced as long as free radicals are being formed. Free radical polymerisation is easily initiated in the presence of light and heat and its rate for polymer formation increases exponentially with temperature. A general rule is that for every 10ºC increase in temperature the rate of polymer formation doubles. Free radical polymerisation readily takes place in heat exchanger tubes and storage tanks [6].

The formation of free radicals has been investigated extensively and it is known that numerous types of free radicals can be formed in a feed stream. These include alkyl radicals produced by the breaking of double or unsaturated bonds as well as other types of precursors such as nitrogen and sulphur radicals which arc easily formed at the temperatures found in the heat exchanger train. Organic sulphur, nitrogen and oxygen compounds increase the potential for various polymerisation reactions, depending on the form in which they exist. Acidic compounds can promote free radical polymerisation by initiating free radicals through the formation of a positive ion or cation. Additional polymerisation precursors include carbonyls, mercaptans, and pyrrole nitrogen.

Oxygen may also react with hydrocarbons to form peroxide free radicals, a step that could occur in the storage tank. When the temperature is increased in the heat exchangers, the peroxides start fast polymerisation reactions leading to the formation of polymers which increase in chain length as more hydrocarbons are attached. The oxygen source is typically from air in non-blanketed storage tanks or oxygenated compounds in the feed stream, which become more reactive as the feed stream is heated [6, 11].

At lower temperatures, free radicals may be formed when a ligand is broken from a metal complex or salt. The unshared electrons resulting from this break react with an unsaturated hydrocarbon or oxygen to form a free radical [5]. There are numerous transition metals which, in very low concentrations, can act as a catalyst and initiate polymerisation reactions.

Some of these catalytically reactive metals are iron, copper, nickel, vanadium, chromium, calcium, and magnesium [6]

In non-free radical polymerisation, polymer formation results from the reaction of two different molecules under the right conditions. One of the reactive molecules may be a radical, or a compound from a free radical-initiated polymerisation step. Basic compounds can react with other compounds or with themselves to form polymers by several different polymerisation mechanisms. In condensation polymerisation, two large radicals or compounds react together to form an even larger compound, but in their reaction also generate a smaller compound, such as water. This new larger compound can continue to react with other reactive species in the feed stream to make higher molecular weight polymers. At some point, the polymer will either get so large in size that it is no longer able to stay entrained or soluble in the fluid stream and deposit, or all the different compounds that can react with it are consumed, and no further polymer is formed [6].

Various laboratory tests can provide an indication of a stream's polymerisation potential. These include laboratory simulations and analytical characterisation, to identify specific compounds in the feed which are known to contribute to polymerisation mechanisms. Such polymerisation precursors may include unsaturated hydrocarbons, acidic compounds, amines, carbonyls, mercaptans and pyrrole nitrogen.

The presence of unsaturated components in the feed stream contributes significantly to polymerisation reactions, particularly at high temperatures. The bromine number is a method of measuring the degree of unsaturation in a feed stream. The unsaturated bonds react with bromine, and the amount of bromine reacted is an indication of the degree of unsaturation [12].

The neutralisation, or acid, number measures the acidity of the fluid as it is titrated with a base. This number can be an indication of fouling tendency, where the more acidic the feed stream, the greater is its tendency to foul. This is most likely due to the fact that acidic compounds, as mentioned above, can promote free radical polymerisation.

The basic nitrogen test determines the amount of basic compounds in a sample, assumed to be mostly amines, by titrating with a mixture of organic acids. This method can, however, overestimate the basic nitrogen content.

A method of determining a sample's oxidative polymerisation potential is to run a potential gums test. This test is a method of determining a sample's oxidative polymerisation potential. In this test, the fluid is subjected to 100% oxygen for four hours, at 100ºC, in a pressurised sample bomb. The measured gum content, as compared to an initial gum value, will indicate the impact of oxygen on the stream's polymerisation potential.

Detailed deposit analysis, as mentioned earlier, can also indicate the occurrence of polymerisation. It is apparent from examination of the deposit analysis results shown in Table 2 that most deposits are organic in nature, as the loss reported on heating the deposit

samples to 840°C was greater than 80% in both the "B" and "C" heat exchangers, where working temperatures are rather high. Since organic deposits result mainly from polymerisation reactions, the high organic content observed in the deposit analysis could be taken as an indication that the fouling in these two heat exchangers is due mainly to polymerisation, which could take place in the heat exchangers themselves or it could occur prior to the heat exchangers either during storage or in transport. Analysis for metals in the deposits indicates the presence of individual metals in the stream. Although, some of these metals are only found in very low concentrations, this may be sufficient for catalysing different polymerisation mechanisms [7].

2.3. Aggregation and flocculation

Some of the heavy organics, especially asphaltenes, will separate from the oil phase into large particles or aggregates. These aggregates may then remain in the oil by some peptising agents, like resins, which will be adsorbed on their surface and keep them afloat, but the stability of such steric colloids is a function of concentration of the peptising agent in the solution. When this concentration drops to a point at which its adsorbed amount is not high enough to cover the entire surface of heavy organic particles, these particles coalesce together, grow in size and flocculate. Flocculation of asphaltene in paraffinic crude oils is known to be irreversible. Due to their large size and their adsorption affinity to solid surfaces flocculated asphaltenes can cause irreversible deposition. Segments of the separated particles which contain S, N and/or H bonds could also start to flocculate and as a result produce the irreversible heavy organic deposits which may be insoluble in solvents.

Inorganic particles may also act as nuclei on which agglomeration of organic particles proceed until the particles become eventually large enough to drop out.

2.4. Transport and migration to the fouling sites

Starting with submicron particles, three transport mechanisms progressively predominate in turbulent flow as the particle size increases. After Gudmunsson [13], the corresponding regimes are designated simply as diffusion, inertia and impaction, respectively.

2.4.1. Diffusion

In the diffusion regime, suspended colloidal particles i.e., particles smaller than about 1 μm in at least one dimension, move with the fluid and are carried to the wall by the Brownian motion of the fluid molecules and through the viscous sublayer in the case of a turbulent flow. The submicron particles can then be treated like large molecules, so that the transport coefficient becomes equivalent to the conventional mass transfer coefficient, which can be obtained from the relevant empirical correlations or theoretical equations for forced convection mass transfer in the literature [14].

In the diffusion regime, the smaller the particle size, the greater is its propensity to be deposited. Thus, it is precisely the very fine submicron particles that are most difficult to remove by filtration or other means which have the greatest propensity to foul a surface.

2.4.2. Inertia

The transition from diffusional to inertial control of transport occurs at particle diameter in the order of 1–2 μm. In the inertia regime the particles are sufficiently large that turbulent eddies give some of them a transverse (Free flight) velocity which is not completely dissipated in the viscous sublayer. These particles then possess sufficient momentum to reach the wall. Some of the particles also experience a more gradual movement towards the wall by migration down the turbulence intensity gradient, i.e. by "turbophoresis" [15]. Much work has been done by a large number of investigators on predicting the results of this free flight excursion or inertial coasting in a turbulent field [16].

In the inertial regime a more desirable situation prevails. Here the larger particles, which are relatively easy to remove, are those which have the greater propensity to be deposited.

2.4.3. Impaction

In this regime, which starts at particle diameter $d_p \cong 10$–20 μm, the particle velocity towards the wall approaches the friction velocity and the particle stopping distance becomes of the same order as the pipe diameter. The response of such large particles to turbulent fluctuations becomes limited and the transport coefficient therefore levels off. As the particles get still larger they get even more sluggish in their response to turbulent eddies and the transport coefficient actually starts to fall gradually [14].

In the impaction regime, transport-controlled deposition would be virtually independent of particle size.

There is considerable experimental evidence to indicate that the effect of surface roughness is usually to enhance the transport of particles to the surface. The enhancement occurs because of the decrease of viscous sublayer thickness and corresponding increase in turbulence level above the roughness elements, because of the smaller stopping distance required for the particles to arrive at the outer asperities of the roughness elements, and because of the additional mechanism of particle interception by those elements along flow lines parallel to the macrosurface [15].

On the other hand, turbulent particle transport may be retarded as a result of deposition of very fine particles which tends to smooth initially rough surfaces. Transport-retardation is, however, far less common than transport-enhancement by surface roughness. The importance of clean and, where feasible also, polished surfaces for mitigating particle deposition under transport-controlled conditions is thus apparent [15].

2.5. Phase separation

Separation of solid particles from fluid stream and their eventual deposition onto heat exchanger surfaces may be a result of many physical processes including condensation from gas phase, gravitational settling, crystallisation and electro-kinetic effect.

Suspended particles such as sand, silt, clay, and non-oxides may become too large to remain entrained in the flowing fluid stream. If the particles are sufficiently large and/or heavy that gravity controls the deposition process, we then have what is known as sedimentation fouling, which can often be prevented with relative ease by pre-filtration or pre-sedimentation of the offending particles. Sedimentation fouling is strongly affected by fluid velocity, and suspended particles in the process fluids will deposit in low-velocity regions, particularly where the velocity changes quickly, as in heat exchanger water boxes and on the shell side [17]. Wall temperature, on the other hand, may have less effect in general on sedimentation fouling, although a hot wall may cause a deposit to "bake on" and become very hard to remove.

Dissolved inorganic salts in a polydisperse fluid may become supersaturated if any change in temperature, pressure, composition (such as solvent evaporation or degasification or addition of a miscible solvent) or other factors destabilises the fluid. The heavy and/or polar fractions may then separate from the fluid into steric colloids, micelles (= charged groups of molecules), another liquid phase or into a solid precipitate.

The dependence of salt solubility on temperature is often the driving force for precipitation fouling. This temperature dependence may be different for different salts, with salt solubility increasing or decreasing with increasing temperature so that different salts may foul the cooling or heating surfaces depending on their solubility temperature dependence. While for most salts the solubility gets higher with increasing temperatures, there are salts such as calcium sulphate which have retrograde solubility dependence and are therefore less soluble in warm streams. Such salts will crystallise on heat transfer surfaces if the streams encounter a surface at a temperature higher than the saturation temperature of these salts. The calcium sulphate scale is hard and adherent and usually requires vigorous mechanical or chemical treatment to remove it. Other typical scaling problems are calcium and magnesium carbonates and silica deposits.

Crystallisation normally begins at specially-active nucleation sites such as scratches and pits, whereas a scratch-free or a smooth surface can flush salt crystals. Subsequently particle deposit will start and continue to build up as long as the surface in contact with the fluid has a temperature above or below saturation. High fluid velocity, by increasing the attrition, can however reduce the rate of particle deposition and fouling.

The solubility of certain heavy hydrocarbons with high melting points such as paraffin wax and diamondoids depends strongly on temperature. If the temperature is decreased, the heavy hydrocarbons may precipitate in the form of solid crystals. Deposition of paraffin wax in cooled heat exchanger tubes showed an asymptotic behaviour due to decreasing heat flux

and increasing shear stress [18]. When various heavy organic compounds are present in a petroleum fluid, their interactive effects largely determine their collective deposition especially when one of the interacting heavy organic compounds is asphaltene.

Changes in the nature of oil fluids may lead to the precipitation of some heavy hydrocarbons, mainly asphaltenes, exceeding their solubility limits. Aspaltenes precipitation, which may be a major cause of crude unit fouling, is affected by many factors including variations of temperature, pressure, composition, flow regime, and wall and electrokinetic effect.

The deposition of heavy hydrocarbons is an example of what is known as solidification fouling, another example of which is the solidification of molten ash carried in a furnace exhaust gas onto a heat exchanger surface.

Precipitation fouling can also occur as a result of pressure changes, where the solubility of salts such as calcium sulphate decreases with decreasing pressure. Laboratory tests have further indicated that variations of pressure exerted on a petroleum fluid can cause the deposition of some of its heavy organic contents.

Motion of charged particles in a conduit may lead to the development of electrical potential differences along the conduit. This electrical potential difference could then cause a change in charges of the colloidal particles further down in the conduit, the ultimate result of which is their untimely deposition. The factors influencing this effect are the electrical and thermal characteristics of the conduit, flow regime, flowing oil properties, characteristics of the polar heavy organics and colloidal particles.

2.6. Particle deposition

Deposition and attachment of solid particles on heat exchanger surfaces is a function of several different operating variables which include particle size and concentration, bulk fluid density and bulk fluid velocity through the heat exchanger [4, 19]. Furthermore, the stickiness and attractive or repulsive forces between particles can significantly contribute to the deposition of particles [3]. Organic deposits may also be the result of heavy hydrocarbon particles bound to the metal surfaces by inorganic deposition. Attachment is also a function of the interfacial properties of the fouling material and the roughness and wettability of the surface where the fouling is going to occur. Whereas smooth and nonwetting surfaces may delay fouling, rough surfaces provide “nucleation sites” that encourage the laying down of the initial fouling deposits. Most initially smooth walls would tend to roughen as particle deposition occurred, so that roughness would then have to be taken into account. On the other hand, deposition of very fine particles onto initially rough surfaces can conceivably result in filling the roughness cavities, thereby smoothing the surface [15].

Recent studies have shown that particle size and concentration have great impact on every type of particle deposition. The average diameter of particles entrained in the fluid stream may vary widely, between a maximum of over 350 μm and a minimum of less than 90 μm (Table 4). Solid particles which foul heat exchangers range in size from submicron to several

hundred microns. Investigation revealed that shell and tube exchangers are generally plugged by particles above 20 microns. On the other hand, plate fin exchangers, having much narrower slots, can be plugged by particles as small as 2 microns [3]. The deposition mechanism for the smaller particles is Brownian diffusion while for the larger particles (10-100 μm) it is mainly gravitational settling. At areas of minimum flow velocities, the larger particles in the stream deposit first followed by the smaller particles and the fouling layer starts to build up as a consequence.

2.7. Deposit growth, aging and hardening

Following particle deposition, deposit growth and consolidation or alternatively auto-retardation and erosion, re-entrainment or removal may take place.

The rate of deposition growth and the accumulation of particles on heat exchanger surfaces is a function of the nature of the fouling material, the composition of the fluid stream and other variables such as temperature and flow rate.

With time, the surface deposit strength may increase and the deposit hardens through various processes collectively known as aging such as, for example, polymerisation, recrystallisation and dehydration. Some types of particles can bake on the surface and will become more difficult to remove over time. The toughness of the deposits may be further affected by the presence of asphaltenes, which are highly polar compounds, and which could act as glue and mortar in hardening the deposits. Biological deposits, on the other hand, may weaken with time due to contamination of organisms.

2.8. Auto-retardation and erosion or removal

The decline of particle deposition rate is commonly referred to as auto-retardation. This is a desirable but spontaneous process that is in the main not under the control of the designer or operator. Several mechanisms may account for auto-retardation and the progressive decrease in adherence of particles to the surface including the already referred to possibility of slowing down particle transport in cases where very fine particles fill the roughness cavities of a surface.

Depending on the strength of the deposit, erosion occurs immediately after the first deposit has been laid down. In saw-tooth fouling part of the deposit is detached after a critical residence time or once a critical deposit thickness has been reached. The fouling layer then builds up and breaks off again. Sometimes impurities such as sand or other suspended particles in fluid streams may have a scouring action, which will reduce or remove deposits [13].

3. Fouling mitigation, control and removal

In order to prevent or mitigate the impact of fouling problems, various steps can be taken during plant design and construction and also during plant operation and maintenance.

However, fouling mitigation and control is a very complex process and anticipating the likely extent of fouling problems to be encountered with different flow streams is a major difficulty faced alike by designers and operators of heat exchangers. In most cases, optimisation of the design and operational conditions is not possible or at least would not be realistic without a comprehensive modelling of the process backed up by practical observations. Modelling, however, is not an easy process, and the different models available in the literature are generally of limited value and application.

The use of multiple regression analysis (MRA), which is an extension of simple least squares regression analysis on a set of data, is an excellent means of modelling heat exchanger fouling. A dependent variable, such as the heat exchanger outlet temperature, is regressed against a set of independent variables, temperatures, pressures, and flows, which directly impact the dependent variable. Regression analysis results in a model equation of independent variables that combine to yield the dependent response. Variability in data and the interaction between independent variables is taken into account in the model equation which can be used to predict future performance. The impact of a change, such as antifoulant addition, can then be compared to the predicted response from the model to determine how effective the treatment programme is.

3.1. Plant design and construction

Fouling mitigation and control require scientific considerations in design and construction. In general, high turbulence, absence of stagnant areas, uniform fluid flow and smooth surfaces reduce fouling and the need for frequent cleaning. In addition, designers of heat exchangers must consider the effects of fouling upon heat exchanger performance during the desired operational lifetime of the heat exchangers. The factors that need to be considered in the designs include the extra surface required to ensure that the heat exchangers will meet process specifications up to shutdown for cleaning, the additional pressure drop expected due to fouling, and the choice of appropriate construction materials. The designers must also consider the mechanical arrangements that may be necessary for fouling inspection or fouling removal and cleaning.

Fouling resistances are different in different designs of heat exchangers. More than 35-40% of heat exchangers employed in global heat transfer processes are of the shell and tube type of heat exchangers. In process industries, more than 90% of heat exchangers used are of the shell and tube type [20]. This is primarily due to the robust construction geometry as well as ease of maintenance and upgrades possible with the shell and tube heat exchangers [1]. Well established procedures for their design and manufacture from a wide variety of materials, as well as availability of codes and standards for design and fabrication and many years of satisfactory service make them first choice in most process industries. However, fouling resistance in the shell and tube heat exchangers are usually much greater than in other types of heat exchangers (Table 5). In the shell side in particular lower fluid flow velocities and low-velocity or stagnant regions, for example in the vicinity of baffles, encourage the

accumulation of foulants. Furthermore, segmental baffles have the tendency for poor flow distribution if spacing or baffle cut ratio is not in the correct proportions. Fouling resistance in plate heat exchangers, on the other hand, can be much smaller. This may be due to the high degree of turbulence even at low velocities which keeps solids in suspension. Also, in plate exchangers there are no dead spaces where fluids can stagnate and solids deposit. Furthermore, heat transfer surfaces are generally smooth and plates are built with higher-quality materials with no corrosion products to which fouling may adhere. Finally, cleaning of plate heat exchangers is a very simple operation and the interval between cleanings is usually smaller [21]. Hence, the fouling factors required in plate heat exchangers are normally 20-25% of those used in shell and tube exchangers [22]. In certain applications, spiral plate exchangers may be chosen for fouling services, where the scrubbing action of the fluids on the curved surfaces minimises fouling. On the other hand, fouling is one of the major problems in compact heat exchangers, particularly with various fin geometries and fine flow passages that cannot be cleaned mechanically [23].

Type	Shell and tube	Plate	Spiral plate	Air cooled	Lamella	Plate fin	Coiled tube	Double pipe	Graphite	Scraped surface
Fouling risk	Very poor	Very good	Good	Poor	Fair	Poor	Fair	Fair	Fair	Very good
Fouling effect	Poor	Good	Good	Very poor	Poor	Very poor	Poor	Fair	Poor	Good

Table 5. Fouling risk and effects for different types of heat exchangers [24, 25]

Over the years, there has been much advance in the design and manufacture of shell and tube heat exchangers with resultant improvements in their fouling behaviour in operation. A striking example of a new design is the Helixchanger heat exchanger (Fig. 4) where the conventional segmental baffle plates are replaced by quadrant shaped baffles arranged at an angle to the tube axis creating a uniform velocity helical flow through the tube bundle. Near plug flow conditions are achieved in a Helixchanger heat exchanger with little back-flow and eddies, often responsible for fouling and corrosion. Low fouling characteristics are provided offering much longer exchanger run lengths between scheduled cleaning of tube bundles. Such run lengths are increased by 2 to 3 times those achieved using the conventionally baffled shell and tube heat exchangers. Heat exchanger performance is maintained at a higher level for longer periods of time with consequent savings in total life cycle costs of owning and operating Helixchanger heat exchanger banks [1].

If fouling is expected on the tube side, some engineers recommend using larger diameter tubes (a minimum of 25 mm OD) [26]. The use of corrugated tubes has been shown to be beneficial in minimising the effects of at least two of the common types of fouling mechanisms, *viz. deposition fouling* because of an enhanced level of turbulence generated at lower velocities, and *chemical fouling* because the enhanced heat transfer coefficients

produced by the corrugated tube result in tube wall temperatures closer to the bulk fluid temperature of the working fluids.

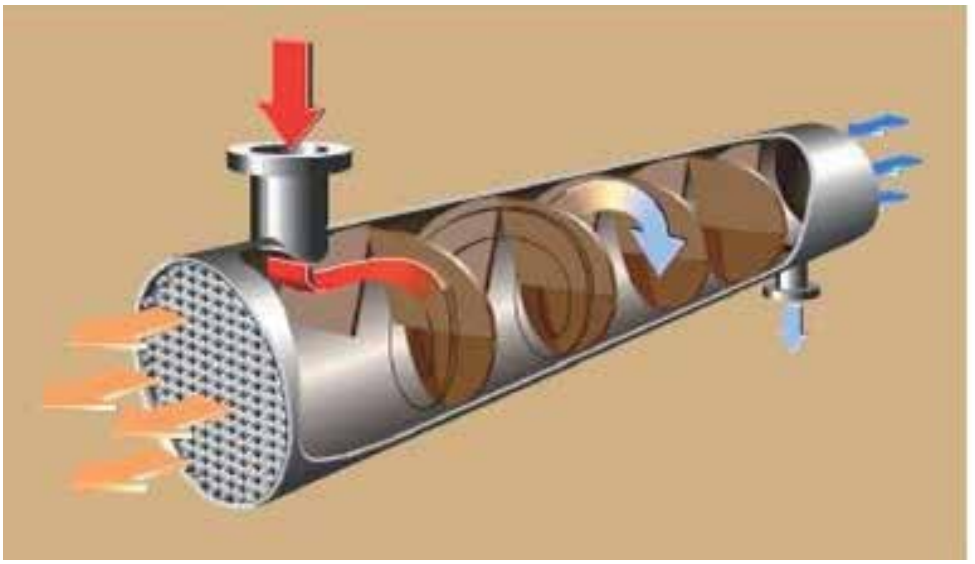


Figure 4. Helixchanger heat exchanger.

Mounting the heat exchanger vertically can minimise the effect of deposition fouling as gravity would tend to pull the particles out of the heat exchanger away from the heat transfer surface even at low velocity levels. Appropriate orientation of heat exchangers may also make cleaning easier [27]. In fluid allocation, it is usually preferred to allocate the most fouling fluid to the tube side as it is easier to clean the tube interiors than the exteriors and the probability of low-velocity or stagnant regions is less on the tube side. Placing the fouling fluid in the tube side tends also to minimise fouling by allowing better velocity control. The use of concurrent flow instead of counterflow is a strategy that may be resorted to sometimes in order to control solidification fouling [23].

Appropriate choice of construction materials for heat transfer surfaces may be necessary to alleviate fouling problems. For example, the use of low-fouling surfaces such as surfaces implanted with ions, very smooth surfaces or surfaces of low surface energy may be an option for some applications. Surface coatings and treatment, ultraviolet, acoustic, electric and radiation treatment, may further help to alleviate fouling problems. Surface treatment by plastics, vitreous enamel, glass, and some polymers can also minimise the accumulation of deposits [13]. Similarly, if biofouling is expected or encountered, the use of non-ferrous high copper alloys, which are poisonous to some organisms, can discourage the settling of these organisms on the heat transfer surfaces. Alloys containing copper in quantities greater than 70% are effective in preventing or minimising biological fouling, and generally 70% to 90% copper and 30% to 10% nickel are used for this purpose. Copper alloys are however prohibited in high-pressure steam power plant heat exchangers, since

the corrosion deposits of copper alloys are transported and deposited in high-pressure steam generators and subsequently block the turbine blades. Environmental protection also limits the use of copper in river, lake, and ocean waters, since copper is poisonous to aquatic life [23].

Corrosion-type fouling can also be minimised by the choice of a construction material which does not readily corrode or produce voluminous deposits of corrosion products. A wide range of corrosion resistant materials based on stainless steel is now available to the heat exchanger manufacturer. Noncorrosive but expensive materials such as titanium and nickel based alloys may be used sometimes to prevent corrosion. If one of the fluids is more corrosive, it may be convenient to send it through the tube side because the shell can then be built with a lower-quality and cheaper material.

The construction material selected must also be resistant to attack by the cleaning solutions in situations where chemical removal of the fouling deposit is planned. For fluid allocation, it is usually preferred to allocate the most fouling fluid to the tube side as it is easier to clean the tube interiors than the exteriors.

3.2. Plant operation and maintenance

In many cases, even the right design of a heat exchanger will not prevent fouling problems that may not be predictable at the design stage. For the control and mitigation of fouling it is generally necessary to take into account the different plant operational conditions such as temperature range, fluid flow rate and chemical composition, and, where possible, make such changes as are required by the severity and type of the fouling problems. For example, some types of fouling can be minimised by using high flow velocities, with due consideration of the possibility of metal erosion as it may be necessary to restrict the velocity to values consistent with satisfactory tube life.

Several techniques may be used for the control of fouling as part of plant maintenance. Some of these techniques are designed to prevent or mitigate fouling. These include avoidance of feed contact with air or oxygen by nitrogen blanketing, elimination or reduction of unsaturates, prior treatment of feed, the use of anti-foulants and application of mechanical on-line mitigation strategies. Cathodic protection and surface treatment such as passivation of stainless steel will minimise corrosion fouling [23].

Prior treatment of feed includes caustic scrubbing, desalting, filtration or sedimentation of feed. Caustic scrubbing removes sulphur compounds and desalting reduces trace metal contamination, both of which reduce polymerisation fouling [10]. Depending on system parameters, including fluid temperature, viscosity, pressure, solid concentration, particle size distribution, and fluid compatibility with the filter media, a filter can be designed to remove solid particles from the fluid. Filtration, however, can only remove the larger-sized particles leaving the smaller-sized particles in the feed stream. Filters used on the feed line require also regular maintenance. At the filter design stage, the

most important question to be answered is, whether the cost of filtration is higher than the fouling cost [3].

Antifoulants or chemical fouling inhibitors may be used to reduce fouling in many systems mainly by preventing reactions causing fouling, and minimising or interfering with the different steps of the fouling process such as crystallisation, agglomeration of small insoluble polymeric or coke-like particles, sticking or attachment of particles to tube walls and deposit consolidation [28]. Such antifoulants include antioxidation additives used to inhibit polymerisation reactions, metal coordinators which react with the trace elements and prevent them from functioning as fouling catalysts, corrosion inhibitors and dispersion agents. Other antifoulants may be used to control crystallisation such as distortion and dispersion agents, sequestering agents and threshold chemicals [28].

Various strategies and devices for the continuous mitigation and reduction of fouling have been proposed such as periodical reversal of flow direction for the removal of weakly adherent deposits, intermittent air injection and/or increasing wall shear stress by raising flow velocity or by increasing turbulence level. In order to enhance the removal of the fouling deposits, velocities in tubes should in general be above 2 m/s and about 1 m/s on the shell side [10]. In several patents, tubular heat exchangers equipped with fouling reduction devices mounted inside the exchanger tubes are described [29]. Such fouling reducers may comprise a mobile turbulence generating element that consists of a metallic winding in the form of a solenoid. The solenoid can be held in position by a hanging system in such a manner that the turbulence generating element can be driven in rotation by the liquid that circulates in the exchanger. The mobile components can be made of spring steel to make them unstretchable. Alternatively, an elastic solenoid may be used that extends over the entire length of the tubes and is agitated by the liquid that circulates in the exchanger [19]. In some heat-transfer applications, mechanical mitigation with dynamic scraped surface heat exchangers is an option. In self-cleaning fluidised-bed exchangers, a fluidised bed of particles is used to control fouling on the outside or inside of tubular exchangers. The self-cleaning exchanger consists of a large number of parallel vertical tubes, in which small solid particles are kept in a fluidised condition by the liquid velocities. The particles have a slightly abrasive effect on the tube walls, so that they remove the deposits [30].

Finally, different mechanical strategies for continuous on-stream cleaning of the interior surfaces of the tubes have been proposed including such strategies as circulation of cleaning balls such as sponge rubber or grit coated balls and pushing of brushes through tubes. In the sponge rubber ball cleaning system, the balls used for normal operation should have the right surface roughness to gently clean the tubes without scoring the tube surface. To remove heavy deposits, special abrasive balls that have a coating of carborundum are available [31]. In the Amertap System, slightly oversized sponge rubber balls are continuously recirculated through the tubes in order to remove the accumulation of scale or corrosion products. The M.A.N. System provides for on-stream cleaning by passage of brushes through the tubes.

Notwithstanding the various control and maintenance techniques which can minimise fouling problems and reduce their severity, fouling may still occur and fouling removal and process equipment cleaning may be necessary. A review of the patent activities related to fouling indicates in fact that most of the work deals with fouling removal and process equipment cleaning techniques. This could also mean that many process equipment manufacturers face problems after they appear rather than proactively prevent them from occurring [3].

There are several different techniques that can be employed for the removal of fouling. All such techniques require, however, costly system shutdown after a longer period of low efficiency heat transfer. The chief techniques normally utilised are either chemical or mechanical cleaning, but other procedures may sometimes be employed for some specific applications such as ultrasonic cleaning, which is a more recent procedure, and abrasive cleaning.

Mechanical cleaning is generally preferred over chemical cleaning because it can be a more environmentally-friendly alternative, whereas chemical cleaning causes environmental problems through the handling, application, storage and disposal of chemicals. However, mechanical cleaning may damage the equipment, particularly tubes, and it does not produce a chemically clean surface. Furthermore, chemical cleaning may be the only alternative if uniform or complete cleaning is required and for cleaning inaccessible areas. The shell side in particular can only be chemically cleaned. The tubes on the other hand can be mechanically cleaned provided that the tube pattern and pitch provide sufficient space and access to the inside of the bundle, and if mechanical cleaning is required for one of the fluids, the usual practice is to put that fluid in the tube side.

For the chemical removal of fouling material, weak acids and special solvents or detergents are normally used. Chlorination may be used for the removal of carbonate deposits. Mechanical techniques for the removal of fouling include scraping and air bumping. Air bumping is a technique that involves the creation of slugs of air, thereby creating localised turbulence as slugs pass through the equipment.

For tightly plugged tubes drilling, generally known as bulleting, may be employed and for lightly plugged tubes roding is employed. Particularly weakly adherent deposits may be mechanically removed by applying high velocity water jets or a mixture of sand and water. Jet cleaning can be used mostly on external surfaces where there is an easy accessibility for passing the high pressure jet [23].

In cases where biofouling occurs it may be removed by either chemical treatment or mechanical brushing processes. In chemical cleaning techniques biocides are employed such as chlorine, chlorine dioxide, bromine, ozone and surfactants. A more usual practice, however, is by continuous or intermittent "shock" chlorination which kills off the responsible organisms. Other cleaning techniques that can be effective in controlling biological fouling include thermal shock treatment by application of heat or deslugging with steam or hot water, and some less well-known techniques like ultraviolet radiation [23].

4. Rate of fouling

The rate of fouling is normally defined as the average deposit surface loading per unit of surface area in a unit of time. Deposit thickness (μm) and porosity (%) are also often used for description of the amount of fouling.

Depending on the fouling mechanism and conditions, the rate of fouling may be linear, falling, accelerating, asymptotic or saw-tooth as the case may be.

1. Linear fouling is the type of fouling where the fouling rate can be steady with time with increasing fouling resistance and deposit thickness. This is perhaps the most common type of fouling. It occurs in general where the temperature of the deposit in contact with the flowing fluid remains constant.

Ebert and Panchal [32] have presented a fouling model that expressed the average (linear) fouling rate under given conditions as a result of two competing terms, namely, a deposition term and a mitigation term.

Fouling Rate = (deposition term) - (anti-deposition term)

$$\frac{dR_f}{dt} = \alpha \mathrm{Re}^{\beta} \mathrm{Pr}^{\delta} \exp\left(\frac{-E}{RT_{film}}\right) - \gamma \tau_w \tag{1}$$

where α, β, γ and δ are parameters determined by regression, τ_w is the shear stress at the tube wall and T_{film} is the fluid film temperature (average of the local bulk fluid and local wall temperatures). The relationship in Eq. (1) points to the possibility of identifying combinations of temperature and velocity below which the fouling rates will be negligible. Ebert and Panchal [32] present this as the "threshold condition". The model in Eq. (1) suggests that the heat exchanger geometry which affects the surface and film temperatures, velocities and shear stresses can be effectively applied to maintain the conditions below the "threshold conditions" in a given heat exchanger.

2. Falling fouling is the type of fouling where the fouling rate decreases with time, and the deposit thickness does not achieve a constant value, although the fouling rate never drops below a certain minimum value. Falling fouling in general is due to an increase of removal rate with time. Its progress can often be described by two numbers: the initial fouling rate and the fouling rate after a long period of time.
3. Accelerating fouling is the type of fouling where the fouling rate increases with time. It is the result of hard and adherent deposit where removal and aging can be ignored. It can develop when fouling increases the surface roughness, or when the deposit surface exhibits higher chemical propensity to fouling than the pure underlying metal.
4. Asymptotic fouling rate is where rate decreases with time until it becomes negligible after a period of time when the deposition rate becomes equal to the deposit removal rate and the deposit thickness remains constant. In general, this type of fouling occurs

where the tube surface temperature remains constant while the temperature of the flowing fluid drops as a result of increased resistance of fouling material to heat transfer. Asymptotic fouling may also be the result of soft or poorly adherent suspended solid deposits upon heat transfer surfaces in areas of fast flow where they do not adhere strongly to the surface with the result that the thicker the deposit becomes, the more likely it is to wash off in patches and thus attain some average asymptotic value over a period of time.

The asymptotic fouling resistance increases with increasing particle concentration and decreasing fluid bulk temperature, flow velocity, and particle diameter. The asymptotic fouling model was first described by Kern and Seaton [33]. In this model, the competing fouling mechanisms lead to an asymptotic fouling resistance beyond which no further increase in fouling occurs. The Tubular Heat Exchanger Manufacturers Association (TEMA) standards suggest fouling factors for several fluids based upon the asymptotic values. This approach, however, does not address all fouling phenomena as it does not, for example, address fouling at the "hot" end of a crude oil preheat train, since fouling there does not exhibit the asymptotic behaviour.

5. Saw-tooth fouling occurs where part of the deposit is detached after a critical residence time or once a critical deposit thickness has been reached. The fouling layer then builds up and breaks off again. This periodic variation could be due to pressure pulses, spalling, trapping of air inside the surface deposits during shutdowns or other reasons. It often corresponds to the moments of system shutdowns, startups or other transients in operation.

5. Prediction of fouling factor

The effect of fouling, as has been noted above, is to form an essentially solid deposit of low thermal conductivity upon the heat transfer surface, through which heat must be transferred by conduction. But since the thermal conductivity of the fouling layer and its thickness are not generally known, the only possible solution to the heat transfer problem is by the introduction of a fouling factor in order to take into account the additional resistance to heat transfer and make possible the calculation of the overall heat transfer coefficient. A fouling coefficient also is sometimes specified, which is the reciprocal value of the fouling factor.

In carrying out heat transfer calculations, caution needs to be exerted in selecting fouling factors, particularly where fouling resistances completely dominate the thermal design. The influence of uncertainties inherent in fouling factors is generally greater than that of uncertainties in other design parameters such as fluid properties, flow rates and temperatures [34]. A large fouling factor is sometimes adopted as a safety margin to cover uncertainties in fluid properties and even in process knowledge but the use of an excessively large fouling factor will result in an oversized heat exchanger with two or three times more area than is really necessary. Although there are many experience-based

tabulations available that provide typical fouling factors such as TEMA Table RGP-T-2.4 [35], acceptable evaluation of the effects of fouling needs to be judged and evaluated for each particular application. Such tabulations, however, can be used as a guide in the absence of more specific information.

A number of methods empirical or otherwise have been proposed over the years for the prediction of the rate of fouling in heat exchangers or for estimating a fouling factor to be used in heat transfer calculations. With the advent and development of digital computers with their ability to provide rapid means of performing calculations, new and accurate methods became possible. Matlab being a programming language widely used in all scientific fields and in engineering sciences in particular may be used in conjunction with the artificial neural network (ANN) approach to provide an accurate and reliable method for predicting the fouling rate and rate of heat transfer in heat exchangers. The development of high speed digital computers has provided a rapid means of performing the many calculations involved in the ANN method, and has had a stimulating effect on the current expansion of the ANN method which is progressing at an impressive rate.

In recent years, the ANN method has been applied in many disciplines of engineering and has produced promising results. The main feature of this method is its ability to learn and generalise the relationships in a data set and to provide quick and satisfactory estimations, which make it attractive for many different applications.

The artificial neural network method is a computational structure inspired by a biological neural system. An ANN consists of very simple and highly interconnected processors called neurons. The neurons are connected to each other by weighted links over which signals can pass. Each neuron receives multiple inputs from other neurons in proportion to their connection weights and generates a single output, which may be propagated to several other neurons [36]. The inputs (X) into a neuron are multiplied by their corresponding connection weights (W) and summed together, a threshold (θ), acting at a bias, is added also to the sum. This sum is transformed through a transfer function (f) to produce a single output (Y), which may be passed on to other neurons. The function of a neuron can be mathematically expressed as

$$Y = f\left(\sum wx - \theta\right) \tag{2}$$

Where the transfer function (f) of the neuron is the linear activation function, being in the present work given as:

$$f(x) = \text{purelin}\ (x) \tag{3}$$

for the output layer and the tansing function

$$f(x) = \text{tansig}\ (x) \tag{4}$$

for the hidden layers

Among the various existing kinds of ANNS, the back propagation (BP) learning algorithm has become the most popular in engineering applications [37]. The BP algorithm is designed to solve the problem of determining weight values for a multi-layer ANN with feed forward connections from the input layer to the hidden layers and then to the output layer. The algorithm is an iterative gradient algorithm, designed to minimise the mean square error between the predicted output and the desired output. Fig. 5 shows a scheme of a simple example for BP algorithm, and Fig. 6 is a flow chart for the back propagation (BP) learning algorithm.

Results of the application of the ANN approach show that it can be used to develop the best configuration in the training period. In general, this approach may however be time consuming; it is nonetheless feasible due to its ability to learn and generalise the complex data set with a wide range of experimental conditions. For exhaustive and fundamental treatment of the ANN technique, the reader is referred to any of the textbooks [38, 39].

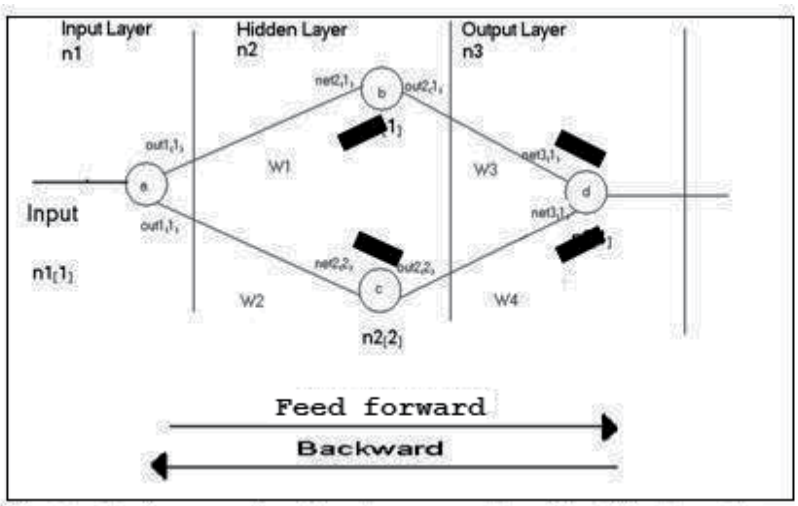


Figure 5. Scheme of a simple example of a BP algorithm.

Some of the recent applications for energy systems include modelling of appliances, light and space cooling energy consumption [40], solar radiation estimation [41], modelling gasoline consumption [42], modelling of heat pumps [43], performance prediction of solar domestic water heating systems [44], prediction of energy consumption of a passive solar building [45], prediction of temperature profiles in producing oil wells [46] and developing heating, ventilating and air conditioning systems for automobiles [47]. In addition, various applications of artificial neural networks in energy problems have been presented thematically [48—50].

In the field of heat exchangers, the Ann approach has been widely used, particularly in the design and control of heat exchangers [51, 52], and in the simulation of heat exchanger performance [14] and heat transfer analysis for different systems such as air—water spray cooling [53].

The ANN approach has also been used as an alternative and practical technique to evaluate the rate of heat exchange and heat transfer coefficient for tubular heat exchangers [34], fin tube heat exchangers [37], fluid—particle systems [54], heat rate predictions in humid air—water heat exchangers [55], and in other applications [22] including in particular the prediction of the rate of fouling and fouling factor in a shell-and-tube heat exchanger [56].

For predicting the rate of fouling and fouling factor in heat exchangers an ANN model can be developed and the available data set used for training the network and verifying its generalisation capability. The rates of heat transfer are then calculated and the input—output pairs presented to the network, and the weights adjusted to minimise the error between the network output and the actual value. Once training is complete, predictions from a new set of data may be done using the already trained network. The proposed algorithm is solved by a Matlab computer programme. After the rate of heat transfer is calculated, it can be used to estimate the fouling factor.

In order to test the applicability of the ANN model using the back propagation learning algorithm to predict the rate of heat transfer and fouling factor for heat exchangers, it was applied on a tube-shell heat exchanger used as a preheat device for the naphtha feed to the reactor in the naphtha hydrotreating unit at the Homs oil refinery.

Operation data of the heat exchanger are collected for this purpose [8]. A total of 73 readings tabulated in Table 6 are used for the ANN method. Empirical correlations for the rate of heat transfer (Q) is determined [57] as given below by Eq.5:

$$Q = m\ Cp\ (t1d\text{-}t2d)\ kJ/hr \tag{5}$$

$Q = q1+q2$
$q1 = m{\cdot}1\times Cp1 \times (t1d\text{-}t2d)$ (liquid)
$q2 = m{\cdot}2\times Cp2\times (t1d\text{-}t2d)$ (gas)
$Cp1 = 0.0045\times t+2.0687$
$Cp2 = Cp\ (hydrogen) + Cp\ (hydrocarbons)$
$Cp\ (hydrogen) = 4.19\ (6.8+0.0006\times t)\ y$
$Cp\ (hydrocarbons) = (0.00428\times t+1.5606)\ (1\text{-}y)$
$y = 2.016\times N/(100\times M)$

In developing an ANN model, the available data set is used for training the network and the same data are used to verify the generalisation capability of the network [58]. The input parameters were liquid naphtha feed quantity (m·1), gas reaction quantity (m·2), inlet temperature (t1d), outlet temperature (t2d), hydrogen purity (N), gas molecular weight (M), naphtha specific heat (Cp1), gas specific heat (Cp2) and the output parameter is heat transfer rate (Q).

The rates of heat transfer (Q) for the same data are calculated using Excel programme (actual values). Input—output pairs are presented to the network, and the weights are

adjusted to minimise the error between the network output and the actual value (this is done at the training step on a number of data (n = 73)). Once training is complete, predictions from a new set of data may be done using the already trained network. The proposed algorithm in this study was solved by a computer programme developed using the Matlab programming language, and all computations were performed with a personal computer. After the rate of heat transfer is calculated, it can be used to estimate the fouling factor.

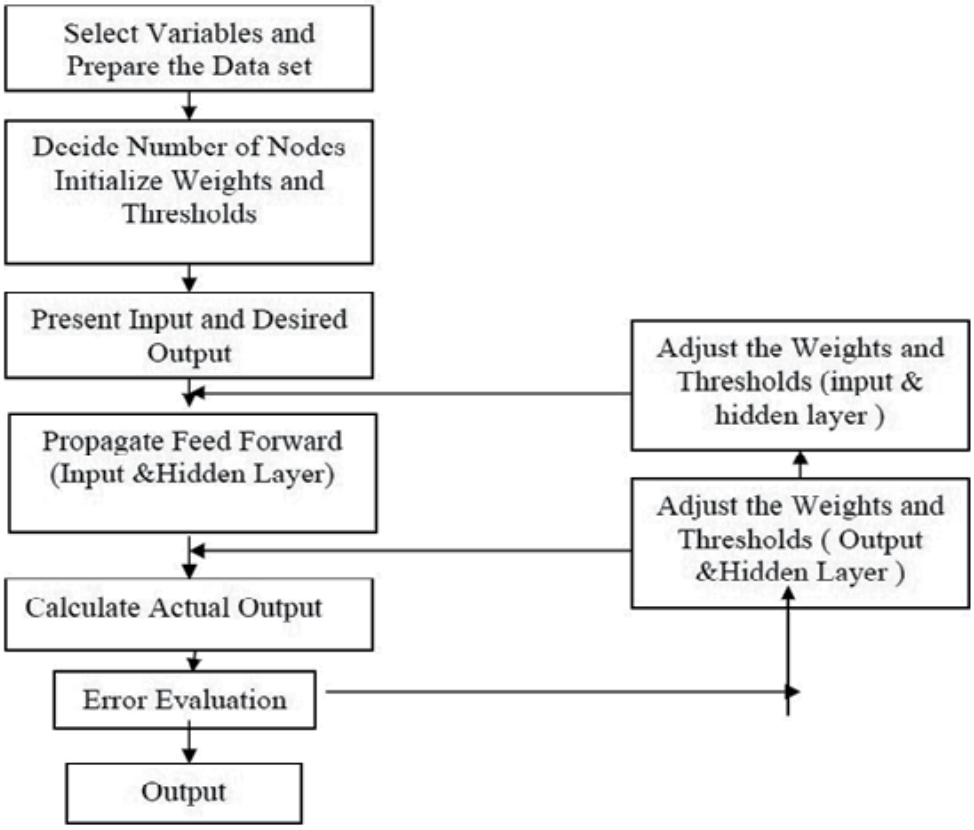


Figure 6. Flow chart for the back propagation (BP) learning algorithm.

The same steps used to calculate the rate of heat transfer by neural network (design, training) are used to calculate fouling factor for different data (Input-output pairs). The empirical correlations for the fouling factor are determined [57] as given below in Eq. 6:

$$f = Uc\text{-}Ud/Uc\times Ud\ m2hr°C/kJ \tag{6}$$

Uc = ho×hi/ho+hi
Ud = Q/A×LMTD
ho = Jh×k/De×pr0.33
hi = Jh×k/D×pr0.33
LMTD = (T1-t2)–(T2-t1)/ln(T1-t2)/(T2-t1)

Gs = m·/as where as = ID×C×B/Pt
Res = Gs×De/μ (shell)
Prs = Cp× μ /K (shell)
Gt= m·/at where at = Nt×at'/z
Ret = D×Gt/ μ (tube)
Prt = Cp× μ /k (tube)

A total of 73 values of input date tabulated in Table 7 are used for the ANN method, which were the rate of heat transfer (Q) obtained from the neural network, log mean temperature difference (LMTD), shell volumetric flow Gs, tube volumetric flow Gt, shell Reynolds Res and Prandtl Prs numbers, tube Reynolds Ret and Prandtl Prt numbers, and the output is the fouling factor (f).

The Fouling factor (f) for the same data is calculated using Excel programme (actual value). Input—output pairs are presented to the network, and the weights are adjusted to minimise the error between the network output and the actual value (this is done at the training step on a number of data (n=73). Once training is complete, predictions from a new set of data may be done using the already trained network. The proposed algorithm in this study was solved by a computer programme developed using Matlab programming language; also all computations were performed with a personal computer.

The purpose of using the ANN model with the considered BP learning algorithm as a practical approach is to test the ability to predict the rate of heat transfer and fouling factor of heat exchanger. For the heat transfer function a network is designed with eight input parameters, namely liquid naphtha feed quantity (m·1), gas reaction quantity (m·2), inlet temperature (t1d), outlet temperature (t2d), hydrogen purity (N), gas molecular weight (M), naphtha specific heat (Cp1) and gas specific heat (Cp2), and one output parameter, the rate of heat exchange (Q). Model sensitivity was examined for different numbers of hidden layer nodes in the range of (1-5). There is no general rule for selecting the number of a hidden layer. The choice of hidden layer size is a specific problem and, to some extent, depends on the number and the quality of training patterns. The number of neurons in a neural network must be sufficient for correct modelling of the problem, and also, it should be low to ensure generalisation.

The number of neurons in a hidden layer drastically affects the outcome of the network training. If too few neurons are included, then the network may not be able to learn properly. On the other band, if too many neurons were included, the network would encourage over fitting. Some researchers mentioned that the upper bound for the required number of neurons in the hidden layer should be greater than twice the number of input units. This rule does not guarantee generalisation of the network [59]. Every stage of an ANN problem requires a little trial and error to establish a suitable and stable network for the problem; so many networks are built by changing their parameters in order to reach a suitable result in these statements:

```
net_q=newff(minmax(input),[8,10,1],{'tansig','tansig','purelin'},'trainlm');
net_q.trainParam.show = 5;
```

```
net_q.trainParam.lr = 0.05;
net_q.trainParam.epochs = 1000;
net_q.trainParam.goal = 1e-13;
[net_q,tr]=train(net_q,input,q);
```

Q	Cp_2	Cp_1	M	N	t_2d	t_1d	m'_2	m'_1
7420754	5.62	2.43	5.9	88.1	113	50	30	1097
7897551	4.93	2.43	7.0	85.3	112	50	37	1173
10427147	4.50	2.42	7.9	82.2	111	48	53	1529
10937005	5.48	2.43	6.2	89.7	113	49	65	1548
6782209	4.47	2.43	7.9	81.6	112	50	69	945
9735495	4.71	2.42	7.3	82.5	111	48	46	1430
10815713	4.50	2.43	7.9	82.2	111	49	80	1564
10467559	5.07	2.42	6.7	85.4	111	47	41	1525
10544419	5.49	2.44	6.2	89.7	114	53	85	1512
9775821	6.32	2.453	5.1	90.2	116	55	75	1380
10508073	5.79	2.443	5.7	88.9	113	53	91	1491
9889096	5.87	2.454	5.6	88.9	115	56	94	1409
10594878	6.12	2.447	5.3	89.9	114	54	88	1507
10333419	6.19	2.436	5.2	89.5	112	51	85	1450
10032838	5.58	2.446	6.0	87.9	114	54	90	1449
9756511	6.05	2.452	5.4	89.6	114	56	83	1435
93056465	5.43	2.433	6.2	88	111	51	86	1342
10170994	5.29	2.433	6.5	88.6	112	50	79	1444
9552569	5.56	2.440	5.9	87	113	52	78	1368
10403199	6.52	2.427	4.9	90,9	110	49	86	1443
7971085	6.03	2.425	5.3	88.6	110	48	64	1105
10522022	5.17	2.418	6.5	85.7	109	46	73	1492
10463870	5.65	2.412	5.8	88.1	107	45	78	1486
10277598	5.18	2.407	6.5	86	106	44	88	1453
10288698	5.56	2.412	5.9	87	107	45	82	1452
9849226	5.36	2.408	6.2	86.5	107	44	77	1394
6762596	4.83	2.415	7.1	84.4	109	45	80	890
9817752	4.36	2.412	8.3	82.6	107	45	73	1488
9993798	5.10	2.400	6.6	85.5	105	42	81	1490
9955236	5.27	2.400	6.3	85.9	105	42	76	1492
9836656	5.38	2.423	6.2	86.9	109	48	82	1497
7035618	5.28	2.392	6.34	86.8	104	40	73	1060
10432795	5.26	2.381	6.2	84.7	102	37	81	1528
9787933	5.25	2.403	6.36	86.2	106	43	76	1471
9465628	5.27	2.415	6.3	85.7	108	46	82	1467

Table 6. Operation data for heat transfer.

f	Tube Pr	Tube Re	Gt	Shell Pr	Shell Re	Gs	LMTD	Q
0.0011	1.48	21626.83	602030	1.62	21123	757392	66.59	7420754
0.0011	1.47	23438.94	646368	1.62	22744	813172	67.64	7897551
0.0008	1.48	30387.8	845085	1.62	29534	1063172	67.38	10427147
0.0007	1.47	31538.99	361645	1.62	30543	1084005	67.95	10937005
0.0015	1.45	21187.03	541667	1.61	19793	681452	73.76	6782209
0.0010	1.46	29257.62	788462	1.62	27492	991935	72.36	9735495
0.0009	1.46	33515.82	878205	1.62	31264	1104839	73.89	10815713
0.0009	1.46	31068.49	836538	1.63	28929	1052419	73.99	10467559
0.0008	1.45	32702.56	853098	1.61	31140	1073253	70.74	10544419
0.0009	1.45	29913.23	777244	1.60	28687	977823	69.61	9775821
0.0008	1.46	32350.07	845085	1.61	30950	1063172	70.04	10508073
0.0010	1.44	32120.8	802885	1.60	30004	1010081	73.84	9889096
0.0008	1.45	32907.62	852030	1.60	31298	1071909	70.96	10594878
0.0010	1.44	32575.66	819979	1.61	29646	1031586	77.83	10333419
0.0010	1.44	32759.62	822115	1.60	30239	1034274	75.68	10032838
0.0010	1.44	32013.2	810897	1.60	29970	1020161	73.48	9756511
0.0011	1.45	29774.46	762821	1.61	27603	959677	75.02	93056465
0.0009	1.46	30972.04	813568	1.61	29213	1023522	72.19	10170994
0.0010	1.45	29631.19	772436	1.61	28037	971774	71.75	9552569
0.0009	1.45	31756.42	816774	1.62	29211	1027554	76.16	10403199
0.0013	1.45	24243.94	624466	1.62	22226	785618	76.65	7971085
0.0010	1.45	32383.05	836004	1.63	29227	1051747	79.01	10522022
0.0011	1.44	32829.65	835470	1.63	29028	1051075	82.29	10463870
0.0011	1.45	32212.47	823184	1.63	28628	1035618	81.54	10277598
0.0011	1.44	32131.64	819444	1.63	28575	1030914	81.41	10288698
0.0012	1.44	32125.61	785791	1.63	27224	988575	83.99	9849226
0.0018	1.43	21261.79	518162	1.62	18711	651882	83.17	6762596
0.0012	1.44	31827.77	817842	1.64	27804	1028898	84.03	9817752
0.0012	1.44	31967.45	817842	1.64	27325	1028898	87.40	9993798
0.0013	1.44	13937.11	816239	1.64	27185	1026882	88.28	9955236
0.0012	1.44	32069.81	821047	1.63	28309	1032930	82.41	9836656
0.0020	1.44	22709.1	577991	1.65	18965	727151	90.99	7035618
0.0013	1.44	32925.24	838141	1.66	27244	1054435	92.43	10432795
0.0013	1.44	31540.68	807158	1.64	27077	1015457	86.75	9787933
0.0013	1.43	31264.22	795406	1.64	26834	1000672	86.70	9465628
0.0012	1.44	31845.35	814637	1.63	27932	1024866	83.42	9780958
0.0012	1.44	31368.23	795940	1.63	27658	1001344	82.63	9517534
0.0013	1.43	32173.46	804487	1.63	27566	1012097	87,09	9509371
0.0014	1.44	29723.24	756410	1.64	25225	951613	88.49	9116431

f	Tube Pr	Tube Re	Gt	Shell Pr	Shell Re	Gs	LMTD	Q
0.0014	1.43	30476.46	760684	1.63	26214	956989	86,47	8893684
0.0015	1.42	33656.34	831197	1.65	27457	1045699	94.73	9748671
0.0014	1.43	33818.59	840812	1.64	28607	1057796	89.17	9601441
0.0013	1.43	32548.29	815171	1.62	28482	1025538	83.92	9280739
0.0014	1.43	34152.85	848825	1.65	28277	1067876	92.50	9834141

Table 7. Operation data for fouling factor.

After 211 training cycles the goal set was achieved, the level of error was satisfactory, and further cycles had no significant effect on error reduction. The network configuration with ten nodes in the hidden layer, a learning rate of 0.05 resulted in the fastest convergence and a low level of error during the training period. To be able to design a stable ANN, it would be more appropriate to conduct a parametric study by changing the number of neurons in the hidden layer in order to test the stability of the network. Fig. 7 shows the performance of the network with the numbers of neurons in the hidden layer (training step); in Fig. 8 the experimental values (actual value) of heat transfer rate (Q) are compared with the results predicted using the best ANN configuration.

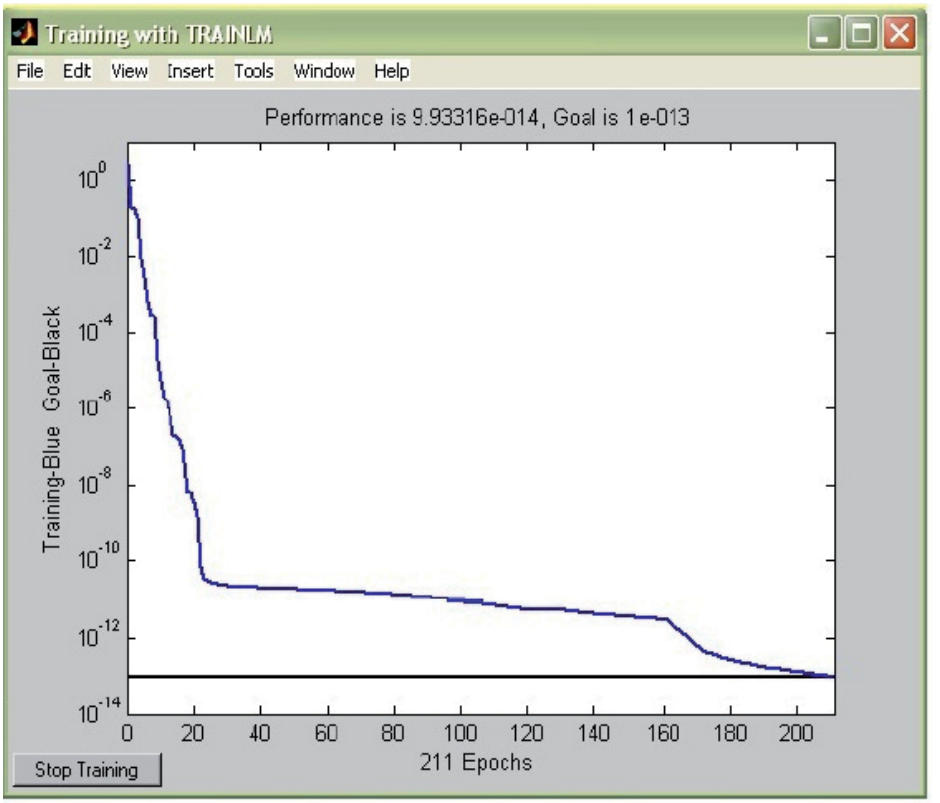


Figure 7. Training step (Heat transfer)

For fouling factor determination a network is designed with eight input parameters, the rate of heat transfer (Q) obtained from the neural network, log mean temperature difference LMTD, shell volumetric flow Gs, tube volumetric flow Gt, shell Reynolds number Res, shell Prandtl number Prs, tube Prandtl number Prt, tube Reynolds Number Ret, and one output parameter, fouling factor (f), all physical properties are estimated at average values of the inlet and outlet temperatures for both sides of the heat exchanger. As before, every stage of an ANN problem requires a little trial and error to establish a suitable and stable network for the problem; so many networks are built by changing their parameters in order to reach a suitable result in these statements:

```
net_f=newff(minmax(input),[8,10,1],{'tansig','tansig','purelin'},'trainlm');
net_f.trainParam.show = 5;
net_f.trainParam.lr = 0.05;
net_f.trainParam.epochs = 1000;
net_f.trainParam.goal = 1e-13;
[net_f,tr]=train(net_f,input,f);
```

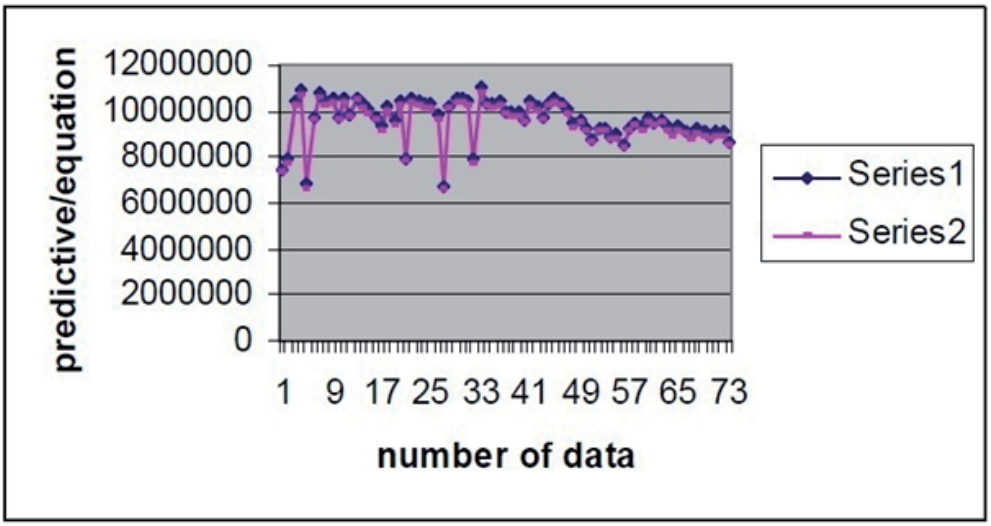


Figure 8. Comparison of experimental and predicted values of heat transfer rate (Q)

After 623 training cycles, the goal set was achieved, the level of error is satisfactory, and further cycles had no significant effect on error reduction. The network configuration with ten nodes in the hidden layer, a learning rate of 0.05 resulted in the fastest convergence and a low level of error during the training period. To be able to design a stable ANN, it would be more appropriate to conduct a parametric study by changing the number of neurons in the hidden layer in order to test the stability of the network. Fig. 9 shows the performance of the network with the numbers of neurons in the hidden layer (training step). In Fig. 10 the experimental values of fouling factor (f) are compared with the results predicted using the best ANN configuration.

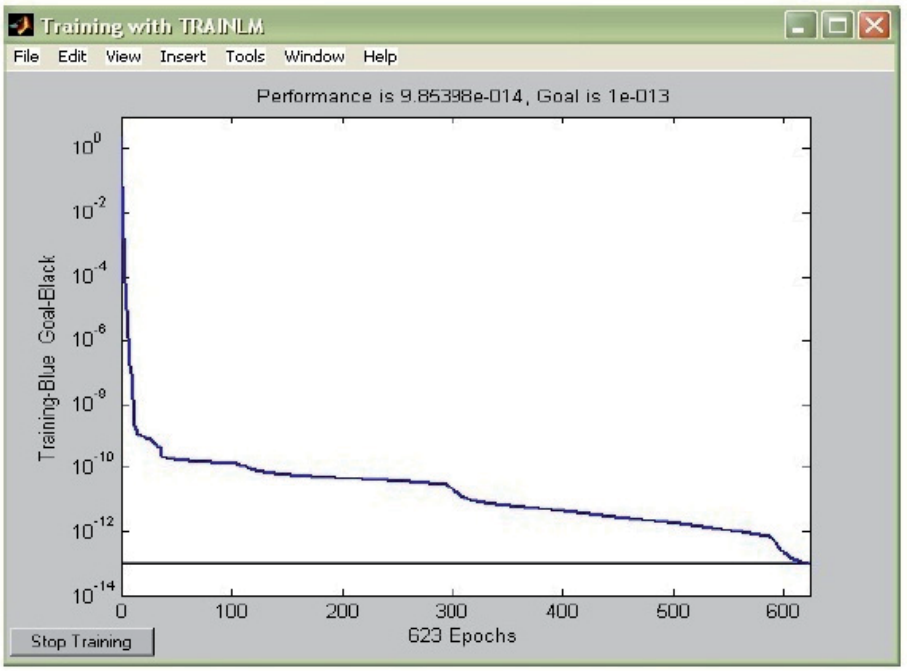


Figure 9. Training step (Fouling factor)

The results obtained and the comparisons made using both the ANN model and empirical correlations show conclusively that the proposed ANN approach could be used successfully to predict both the rate of heat transfer and fouling factor for heat exchangers. The accuracy and reliability of this approach for predicting the fouling factor can go a long way towards mitigating the detrimental effects of fouling in heat exchangers in particular and in other process equipment in which fouling may be of a major concern.

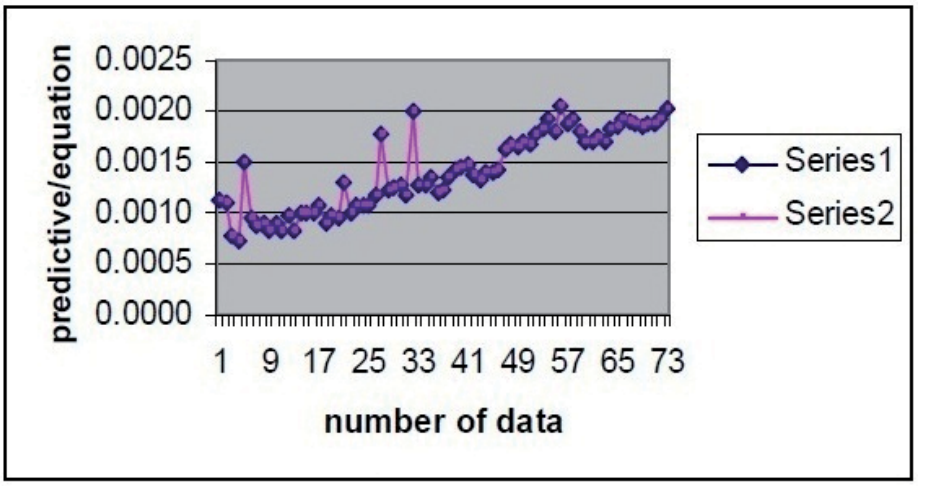


Figure 10. Comparison of experimental and predicted values of fouling factor (f)

Nomenclature

C_p	Specific heat, kJ/kg.°C
C_{p1}	Liquid specific heat, kJ/kg.°C
C_{p2}	Gas specific heat, kJ/kg.°C
D	Flow area per tube, m^2
D_e	Equivalent diameter, m
E	Activation energy, J/mol K
f	Fouling factor, hr.m^2.°C/kJ
G_s	Shell mass flow, kg/hr.m^2
G_t	Tube mass flow, kg/hr.m^2
h_i	Heat transfer coefficient inside tube, kJ/hr.m^2.°C
h_o	Heat transfer coefficient outside tube, kJ/hr.m^2.°C
k	Thermal conductivity, kJ/hr.m.°C
LMTD	Log mean temperature difference, °C
M	Gas molecular weight
m	Mass flow of gas and naphtha, kg/hr
m_1	Liquid mass flow, kg/hr
m_2	Gas mass flow, kg/hr
N	Hydrogen purity, %
n	Number of data
Pr	Prandtl number
Pr_s	Shell Prandtl number
Pr_t	Tube Prandtl number
Q	Total heat transfer, kJ/hr
q_1	Liquid heat transfer, kJ/hr
q_2	Gas heat transfer, kJ/hr
R	Gas constant, J/mol K
Re	Reynolds number
Re_s	Shell Reynolds number
Re_t	Tube Reynolds number
R_f	Fouling resistance, m^2 K/kW
T_{1d}	Tube inlet temperature, °C
T_{2d}	Tube outlet temperature, °C
T_{film}	Fluid film temperature, K
t	Time, s
t_{1d}	Shell inlet temperature, °C
t_{2d}	Shell outlet temperature, °C
U_c	Clean overall coefficient, kJ/hr.m^2.°C
U_d	Dirt overall coefficient, kJ/hr.m^2.°C
W	Weight
x	Input
y	Output
θ	Threshold

μ Viscosity, kg/m. hr
τ_w Shear stress at the tube wall

Author details

Hassan Al-Haj Ibrahim
Department of Chemical Engineering, Al-Baath University, Homs, Syria

6. References

[1] Master BI, Chunangad KS, Pushpanathan V. Fouling mitigation using helixchanger heat exchangers.
[2] Mueller-Steinhagen H, Malayeri MR, Watkinson AP. Fouling of heat exchanger-New approaches to solve old problem. Heat Transfer Engineering, 2005;26(2).
[3] Hashemi R and R. L. Brown, Jr. RL. Heat exchanger fouling causes problems in gas and liquid systems. American Filtration Society Seminar, Chicago, Illinois, May 11, 1992.
[4] Muller-Stehinhagen H, Reif F, Epstein N, Watkinson AP. Influence of operating conditions on particulate fouling. Canadian Journal of Chemical Engineering, 1988: 66, 42-50.
[5] Herro HM. Deposit-Related Corrosion in Industrial Cooling Water Systems, Presented at the National Association of Corrosion Engineers Corrosion '89 meeting, New Orleans, Louisiana, April 17–21, 1989.
[6] Bernard C and Groce PE. Controlling hydrotreater fouling problem identification is key to cost-effective solutions. Betz process Chemicals, Oil and Gas Journal, January 1996.
[7] Al-haj Ibrahim H, Safwat A, Hussamy N. Investigation of the fouling mechanisms in the heat exchangers of a hydrotreater. Engineering Journal of the University of Qatar, 2005: 18, 9-14.
[8] Process and operating manual of naphtha hydrotreating unit, Homs Oil Refinery, 1989, 15-20.
[9] Al-haj Ibrahim H, Safwat A, Hussamy N. Particulate fouling evaluation in the preheat exchangers of a hydrotreater, Yemeni Journal of science, 2006: 7(1), 15-20.
[10] Bott TR. Fouling Notebook, Institution of Chemical Engineers, London, 1990.
[11] Fouling problems on Homs refinery naphtha hydrotreater, Technical note, Betzdearborn, 2000, 1-4.
[12] Vanhove A. Fouling Control in Refinery, Hydrocarbon Engineering, July/august 1998, 50-6.
[13] Gudmundsson JS. Particulate fouling, fouling of heat transfer equipment, E.F.C. Somerscales and J.G. Knudsen (eds.), Proceedings of the International Conference on the Fouling of Heat Transfer Equipment, Hemisphere, Washington, D.C., 1981, 357-388.
[14] Din G, Sen M, Yang KT, McClain RL. Simulation of heat exchanger performance by artificial neural networks, IVAC Res 5, 1999, 195—208.
[15] Epstein N. Particle deposition and mitigation, Dept Of Chemical engineering, The University of British Columbia.
[16] Papavergos PG and Hedley AB. Particle deposition behaviour from turbulent flows, Chem. Eng. Res. Des. 1984: 62, 275-295.
[17] Puckorius PR. Contolling deposits in cooling water systems, Mater. Protect. Perform., November, 1972, 19-22.

[18] Bott TR and Gudmunson JS. Operation of paraffin wax from flowing system, Institute of petroleum, IP77-007, London, 1978.
[19] Muller-Steinhagen H and Blochl R. Particulate fouling in heat exchangers, Transcripts of Institute of Professional Engineers, New Zealand, EMC Eng-Sec., 1988: 15(3), 109-118.
[20] Chisholm D (ed.). Developments in heat exchanger technology-I, Applied Science Publishers, London, 1980.
[21] Marriot J. Where and how to use plate heat exchangers, Chem Eng, May 4, 1971, p. 127.
[22] Marriott J. Where and how to use plate heat exchangers, in Process Heat Exchange, Chemical Engineering Magazine (V. Cavaseno, ed.), McGraw-Hill, New York, 1979, 156-162.
[23] Kuppan T. Heat exchanger design handbook, Marcel Dekker, Inc., New York, 2000.
[24] Larowski A and M.A. Taylor MA. Systematic procedures for selection of heat exchangers, C58/82, Institution of Mechanical Engineers, London, 1982, 32-56.
[25] Larowski A and Taylor MA. Systematic procedures for selection of heat exchangers, Proc. Instn. Mech. Eng., 1983: 197A, 51-69.
[26] Mukherjee R. Conquer heat exchanger fouling, Hydrocarbon Processing, January, 1996, 121-127.
[27] Chenoweth JM. Final Report of the HTRVTEMA Joint Committee to Review the Fouling Section of the TEMA Standards, Heat Transfer Research, Inc., Alhambra, Calif., 1988.
[28] Cowan JC and Weintritt DJ. Water-formed scale deposits. A comprehensive study of the prevention, control, removal and use of mineral scale, Gulf Publishing Company, Houston, Texas, 1976.
[29] Fouling reduction device for a tubular heat exchanger, Patents FR 2479964 dated Apr. 8, 1980; EP 0174254 dated Nov. 9, 1986; US Patent 6782943 dated Aug. 31, 2004.
[30] Klaren DG. Fluid bed heat exchangers-A new approach in severe fouling heat transfer, Resources and Conservation, 1981, 301-314.
[31] Stegelman AF, Renfftlen R. On line mechanical cleaning of heat exchangers, Hydrocarbon Processing, 1983, 95-97.
[32] Ebert WA and Panchal CB. Analysis of Exxon crude-oil slip stream coking data, Fouling Mitigation of Industrial Heat-Exchange Equipment, Begell House, New York, 1997, 451-460.
[33] D. Q. Kern DQ and Seaton RE. A Theoretical Analysis of Thermal Fouling, Br. Chem. Eng., 1959: 4(5), 258-269.
[34] Riverol C, Napolitano V. Estimation of the overall heat transfer coefficient in a tubular heat exchanger under fouling using neural networks, Appl Flash Pasteurizer Int Comm Heat Mass Transfer, 2002: 29, 453—7.
[35] Standards of Tubular Exchanger Manufacturers Associaton, seventh edition, Tubular Exchanger Manufacturers Association, Tarrytown, NY, 1988.
[36] Sreekanth S, Ramaswamy US, Sablani SS, Prasher SO. A neural network approach for evaluation of surface heat transfer coefficient, J Food Process Reser, 1999: 23, 329—48.
[37] Pacheco-Vega A, Sen M, Yang KT, MeClain RL. Neural network analysis of fin-tube refrigerating heat exchanger with limited experimental data, Int J Heat Mass Transfer, 2001: 44, 763—70.
[38] S. Haykin. Neural networks: a comprehensive foundation, New York, McMillan College Publishing Company, 1994.
[39] Fauselt L. Fundamentals of neural networks, New York, Prentice-Hall, 1994.

[40] Aydinalp M, Ugursal VI, Fung AS. Modelling of the appliance, lighting, and space cooling to energy consumptions in the residential sector using neural networks. Appl Energy, 2002: 7l, 87-100.
[41] Dorylo ASS, J.A. Jervase JA, AI-Lawati A. Solar radiation estimation using artificial neural network. Appl Energy; 2002: 71, 307—19.
[42] Masr GE, Hadr EA, Joun C. Back propagation neural networks for modelling gasoline consumption, Energy Conver Manage, 2002: 44, 893—905.
[43] Bechtler H, Browne MW, Bansal PK, Kecman V. Neural networks—a new approach to model vapour-compression heat pumps, Int J Energy Res, 2001: 25, 591—9.
[44] Kalogirou SA. Long-term performance prediction of forced circulation solar domestic water heating systems using artificial neural networks, Appl Energy, 2000: 6663—74.
[45] Kalogirou SA, Bojic M. Artificial neural networks for the prediction of the energy consumption of a passive solar building, Energy, 2000: 1(5), 419—91.
[46] Farshad FE, Garber SD, Lorde IN. Predicting temperature profiles in producing oil wells using artificial neural networks, Eng Comput, 2000: 17, 735—54.
[47] Ueda M, Taniguchi Y, Asano A, Mochizuki M, Ikegarni T, Kawai T. An automobile heating, ventilating and air conditioning (HVAC) system with a neural network for controlling the thermal sensations felt by a passenger, ISME Ins J Series B-Fluid Thermal Eng, 1997: 40, 469—77.
[48] Kalogiron SA. Artificial neural networks in renewable energy systems applications: a review, ken Sustain Energy Rev, 2001: 5, 373—401.
[49] Kalogiron SA. Applications of artificial neural-networks for energy systems, Appl Energy, 2000: 67, 17—35.
[50] Kalogiron SA. Applications of artificial neural networks in energy systems: a review, Energy Conver Manage, 1999: 40, 1073—91.
[51] Diaz G, Sen M, Yang KT, McClain RL. Dynamic prediction and control of heat exchangers using artificial neural networks, Int Heat Mass Transfer, 2001: 44, 1671—9.
[52] Lavric D, Lavric L, Woinaroschy A, Danciu AE. Designing tin heat exchanger with a neural network, Rev Roumaine De Chim, 1995: 40, 561—5.
[53] Oliveira MSA, Sousa ACM. Neural network analysis of experimental data for air/water spray cooling, S Mater Process Technol, 2001: 113, 439—45.
[54] Sablani SS. A neural network approach for non-iterative calculation of heat transfer coefficient in fluid—particle systems, Chem Eng Process, 2001: 40, 363—9.
[55] Pacheco-Vega A, Diax G, Sen M, Yang KT, McClain RL. Heat rate predictions air-water beat exchangers using correlations and neural networks, S Heat Transfer—Trans ASME, 2001: 123, 348—54.
[56] Al-haj Ibrahim H, Safwat A, Hussamy N. Prediction of the rate of heat transfer and fouling factor by using artificial neural network approach, Bassel Al-Assad Journal for engineering sciences, 2007: 23, 9-27.
[57] Kern DQ. Process Heat Transfer, McGraw-Hill Book Co., 1950, 151-153.
[58] Aydinalp M, Ugursal VI, Fung AS. Predicting residential appliance, lighting, and space cooling energy consumption using neural networks, Proc. the Fourth International Thermal Energy Congress, Cesme, Turkey, 2001, 417-22.
[59] Rafig MY, Bugmann G, Easterbrook DI. Neural network design for engineering applications, Comput Struct, 2001: 79, 1541-52.

Chapter 4

Optimal Solution to Matrix Riccati Equation – For Kalman Filter Implementation

Bhar K. Aliyu, Charles A. Osheku, Lanre M.A. Adetoro and Aliyu A. Funmilayo

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46456

1. Introduction

Matrix Riccati Equations arise frequently in applied mathematics, science, and engineering problems. These nonlinear matrix equations are particularly significant in optimal control, filtering, and estimation problems. Essentially, solving a Riccati equation is a central issue in optimal control theory. The needs for such equations are common in the analysis and synthesis of Linear Quadratic Gaussian (LQC) control problems. In one form or the other, Riccati Equations play significant roles in optimal control of multivariable and large-scale systems, scattering theory, estimation, and detection processes. In addition, closed forms solution of Riccti Equations are intractable for two reasons namely; one, they are nonlinear and two, are in matrix forms. In the past, a number of unconventional numerical methods were employed for the solutions of time-invariant Riccati Differential Equations (RDEs). Despite their peculiar structure, no unconventional methods suitable for time-varying RDEs have been constructed, except for carefully re-designed conventional linear multistep and Runge-Kutta(RK) methods.

Implicit conventional methods that are preferred to explicit ones for stiff systems are premised on the solutions of nonlinear systems of equations with higher dimensions than the original problems via Runge-Kutta methods. Such procedural techniques do not only pose implementation difficulties but are also very expensive because they require solving robust non-linear matrix equations.

In this Chapter, we shall focus our attention on the numerical solution of Riccati Differential Equations (RDEs) for computer-aided control systems design using the numerical algorithm with an adaptive step of *Dormand-Prince*. It is a key step in many computational methods for model reduction, filtering, and controller design for linear systems. In the meantime, we shall limit our investigation to the optimality in the numeric solution to Riccati equation as it affects the design of Kalman-Bucy filter state estimator, for an LQG control of an

Expendable Launch Vehicle (ELV) in pitch plane during atmospheric ascent. Furthermore, the approach in the paper by Aliyu Kisabo Bhar et al will be fully employed from a comparative standpoint of the solution to a differential Riccati equation and an Algebraic Riccati for Kalman-Bucy filter implementation.

2. Kalman filter

Theoretically the Kalman Filter is an estimator for the linear-quadratic problem, it is an interesting technique for estimating the instantaneous 'state' of a linear dynamic system perturbed by white -noise measurements that is linearly related to the corrupted white noise state. The resulting estimator is statistically optimal with respect to any quadratic function of the estimation error.

In estimation theory, Kalman introduced stochastic notions that applied to non-stationary time-varying systems, via a recursive solution. C.F. Gauss (1777-1855) first used the Kalman filter, for the least-squares approach in planetary orbit problems. The Kalman filter is the natural extension of the Wiener filter to non-stationary stochastic systems. In contrast, the theory of Kalman has provided optimal solutions for control systems with guaranteed performance. These control analyses were computed by solving formal matrix design equations that generally had unique solutions. By a way of reference, the U.S. space program blossomed with a Kalman filter providing navigational data for the first lunar landing.

Practically, it is one of the celebrated discoveries in the history of statistical estimation theory in the twentieth century. It has enable humankind to do many things, one obvious advantage, is its indispensability as silicon integration in the makeup of electronic systems. It's most dependable application is in the control of complex dynamic systems such as continuous manufacturing processes, aircraft, ships, or spacecraft. To control a dynamic system, you must first know what it is doing. For these applications, it is not always possible or desirable to measure every variable that you want to control, and Kalman filter provides a means of inferring the missing information from indirect (and noisy) measurements. Kalman Filter is also very useful for predicting the likely future course of dynamic systems that people are not likely to control, such as the flow of rivers during flood, the trajectory of celestial bodies, or the prices of traded commodities. Kalman Filter is 'ideally noted for digital computer implementation', arising from a finite representation of the estimated problem-by a finite number of variables. Usually, these variables are assumed to be real numbers-with infinite precision. Some of the problems encountered in its uses, arose from its distinction between 'finite' and 'manageable' problem sizes. These are significant issues on the practical side of Kalman filtering that must be considered in conjunction with the theory. It is also a complete statistical characterization of an estimated problem than an estimator, because it propagates the entire probability distribution of the variables in its task to be estimated. This is a complete characterization of the current state of knowledge of the dynamic system, including influence of all past measurements. These probability distributions are also useful for statistical analysis and predictive design of sensor systems.

The applications of Kalman filtering encompass many fields, but its use as a tool, is almost exclusively for two purposes: estimation and performance analysis of estimators. Figure 1 depicts the essential subject for the foundation for Kalman filtering theory.

Despite the indication of Kalman filtering process in the apex of the pyramid, it is an integral part in the foundation of another discipline *modern control theory*, and a proper subset of statistical decision theory.

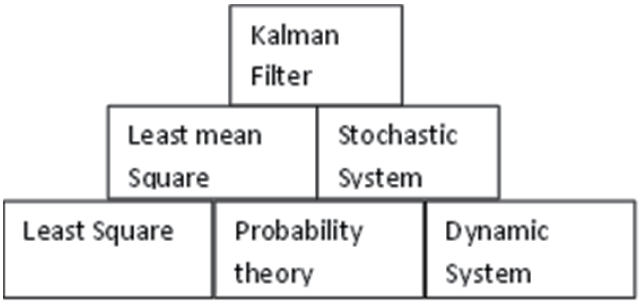


Figure 1. Foundation concept in Kalman filtering.

Kalman filter analyses a dynamic systems' behavior with external and measurement noise. In general, the output y is affected by noise measurement. In addition, the process dynamics are also affected by disturbances, such as atmospheric turbulence. Following this, we shall now present the model for the LQG control for the ELV via equation (1). The essential assumptions are viz; the dynamic system is linear, the performance cost function is quadratic, and the random processes are Gaussian.

By defining, a continuous –time process and measurement model is as follows;

$$\begin{aligned} \dot{x} &= Ax + Bu + Gw, \\ y &= Cx + v. \end{aligned} \tag{1}$$

Where, w and v are zero-mean Gaussian noise processes (uncorrelated from each other). The following process and measurement covariance matrices hold namely:

$$\left.\begin{aligned} Ev(t)v^T(s) &= R_N\delta(t-s) \\ Ev(t)w^T(s) &= 0 \\ Ew(t)w^T(s) &= Q_N\delta(t-s) \end{aligned}\right\} \quad t,s \in \Re \tag{2}$$

From the foregoing,a Kalman filter equation admits the form;

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - \hat{y}), \tag{3}$$

where L is the Kalman gain represented as

$$L = PC^T R_N^{-1}. \tag{4}$$

The covariance matrix P, in equation (4) is the solution to a Riccati Differential Equation (RDE) or an Algebraic Riccati Equation (ARE).

3. Riccati equation

In mathematics, a Riccati equation is any ordinary differential equation that is quadratic in the unknown function. In other words, it is an equation of the form

$$y'(x) = q_0(x) + q_1(x)y(x) + q_2(x)y^2(x), \tag{5}$$

where, $q_0(x) \neq 0$ and $q_2(x) \neq 0$ ($q_0(x)=0$ is Bernoulli equation and $q_2(x)=0$ is first order linear ordinary equation). It is named after Count Jacopo Francesco Riccati (1676-1754).

More generally, "Riccati equations" refer to matrix equations with analogous quadratic terms both in continuous-time and in discrete-time Linear-Quadratic-Gaussian Control. The steady-state (non-dynamic) versions of these equations are classified as algebraic Riccati equations.

3.1. Riccati differential equation (RDE)

The Riccati differential equation was first studied in the eighteen century as a nonlinear scalar differential equation, and a method was derived for transforming it to a linear matrix form. This same method works when the dependent variable of the original Riccati differential equation is a matrix.

The statistical performance of the Kalman filter estimator can be predicted a priori by solving the Riccati equations for computing the optimal feedback gain of the estimator. Also, the behaviors of their solutions can be shown analytically for the most trivial cases. These equations also provide a means for verifying the proper performance of the actual estimator when it is running.

For the LQG problem, the associated Riccati Differential Equation which provides the covariance $P(t)$ needed for the solution of Kalman gain is of the form,

$$\dot{P}(t) = A_\phi(t)P(t) + P(t)A_\phi^T(t) + G(t)Q_N(t)G(t) - P(t)C^T(t)R_N^{-1}(t)C(t)P(t) \tag{6}$$

where A_ϕ is the state transitional matrix defined as

$$\frac{d}{dt}x(t) = A(t)x(t) + G(t)w(t). \tag{7}$$

The Riccati Differential Equation in (6) can be solved by using a technique, called the Matrix Fraction Decomposition. A matrix product of the sort AB^{-1} is called a *matrix fraction*, and a representation of a matrix M in the form

$$M = AB^{-1} \tag{8}$$

A fractional decomposition of the covariance matrix results in a linear differential equation for the numerator and the denominator matrices. The numerator and denominator matrices as functions of time, such that the product $A(t)B^{-1}(t)$ satisfies the matrix Riccati equation and its boundary conditions. By taking the derivative of the matrix fraction $A(t)B^{-1}(t)$ with respect to t and using the fact that

$$\frac{d}{dt}B^{-1}(t) = -B^{-1}(t)\dot{B}(t)B^{-1}(t), \tag{9}$$

Now let us represent the covariance matrix $P(t)$ by

$$P(t) = A(t)B^{-1}(t), \tag{10}$$

and on applying equations (9-10) yields

$$\frac{dP(t)}{dt} = \dot{A}(t)B^{-1}(t) - A(t)B^{-1}(t)\dot{B}(t)B^{-1}(t) \tag{11}$$

From the Riccati equation in (6) substitution for $P(t)$ with $A(t)B^{-1}(t)$ in the right hand side of the equation, leads to the following namely

$$\frac{dP(t)}{dt} = A_{\phi}(t)A(t)B^{-1}(t) + A(t)B^{-1}(t)A_{\phi}^{T}(t) + G(t)Q_N(t)G^T(t)$$

$$-A(t)B^{-1}(t)C^T(t)R_N^{-1}(t)C(t)A(t)B^{-1}(t). \tag{12}$$

Equating (11) and (12) and multiplying through with $B(t)$ yields

$$\dot{A}(t) - A(t)B^{-1}(t)\dot{B}(t) = \left\{A_{\phi}(t)A(t) + G(t)Q_N(t)G^T(t)B(t)\right\}$$

$$-A(t)B^{-1}(t)\left\{C^T(t)R_N^{-1}(t)C(t)A(t) - A_{\phi}^{T}(t)B(t)\right\} \tag{13}$$

Therefore, if we find $A(t)$ and $B(t)$ that satisfy:

$$\dot{A}(t) = A_{\phi}(t) + G(t)Q_N(t)G^T(t)B(t), \tag{14}$$

$$\dot{B}(t) = C^T(t)R_N^{-1}(t)C(t)A(t) - A_{\phi}^{T}(t)B(t), \tag{15}$$

then $P(t)=A(t)B^{-1}(t)$ satisfies the Riccati differential equation. Note that equations (14) and (15) are the linear differential equations with respect to matrices $A(t)$ and $B(t)$. The foregoing can be arranged as follows viz;

$$\begin{pmatrix} A(t) \\ B(t) \end{pmatrix} = \begin{bmatrix} A_{\varphi}(t) & G(t)Q_N(t)G^T(t) \\ C^T(t)R_N^{-1}(t)C(t) & -A_{\varphi}(t) \end{bmatrix} \begin{pmatrix} A(t) \\ B(t) \end{pmatrix} \tag{16}$$

Such a representation is a Hamiltonian Matrix known as matrix Riccati differential equation.

$$\Psi(t) = \begin{bmatrix} A_\phi(t) & G(t)Q_N(t)G^T(t) \\ C^T(t)R_N^{-1}(t)C(t) & -A_\phi(t) \end{bmatrix} \tag{17}$$

The initial values of $A(t)$ and $B(t)$ must be constrained by the initial value of $P(t)$. This is easily satisfied by taking $P_0=I$, an identity matrix.

In the time-invariant case, the Hamiltonian matrix Ψ is also time-invariant. As a consequence, the solution for the numerator $A(t)$and denominator $B(t)$of the matrix fraction can be represented in matrix form as the product

$$\begin{bmatrix} A(t) \\ B(t) \end{bmatrix} = e^{\Psi t} \begin{bmatrix} P(0) \\ I \end{bmatrix}, \tag{18}$$

where $e^{\Psi t}$is a $2n \times 2n$ matrix.

Convergence Properties of a Scalar Time-Invariant Case. In this case, the numerator $A(t)$ and the denominator $B(t)$ of the 'matrix fraction' $A(t)B^{-1}(t)$ will be scalars, but Ψ will be a $2n \times 2n$ matrix. Considering a case: $A(t) \rightarrow a(t)$, and $B(t) \rightarrow b(t)$and the process and measurement equations becomes

$$\begin{aligned} A_\phi(t) &= A_\phi, \\ G(t) &= G, \\ Q(t) &= Q, \\ R(t) &= R, \\ C(t) &= C. \end{aligned} \tag{19}$$

The scalar time-invariant Riccati differential matrix equation and its linearized equivalent is

$$\dot{P}(t) = A_\phi P(t) + P(t)A_\phi - P(t)CR_N^{-1}CP(t) + GQG^T. \tag{20}$$

Hence, equation (16) reduces to

$$\begin{pmatrix} \dot{a} \\ \dot{b} \end{pmatrix} = \begin{bmatrix} A_\phi & GQG^T \\ CR^{-1}C & -A_\phi \end{bmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \tag{21}$$

with the following initial conditions namely; $a(0)=P_0$ and $b(0)=1$. In the meantime, the eigenvalues of the Hamilton Matrix are;

$$\lambda_1, \lambda_2 = \pm\sqrt{A_\phi^2 + \frac{Q}{R}G^2C^2}. \tag{22}$$

Using λ_1 and λ_2 , where, $q=G^2Q$, we can write the following:

$$a(t) = \frac{1}{2\lambda}\{[P_0(\lambda - A_\phi) + q]e^{\lambda t} + [P_0(\lambda - A_\phi) + q]e^{-\lambda t}\} \tag{23}$$

$$b(t) = \frac{1}{2\lambda q}\{(\lambda - A_\phi)[P_0(\lambda + A_\phi) + q]e^{\lambda t} - (\lambda + A_\phi)[P_0(\lambda - A_\phi) - q]e^{-\lambda t}\}. \tag{24}$$

Consequently, the covariance follows as;

$$P(t) = \frac{a(t)}{b(t)} = q\frac{[P_0(\lambda + A_\phi) + q] + [P_0(\lambda - A_\phi) - q]e^{-2\lambda t}}{(\lambda - A_\phi)[P_0(\lambda + A_\phi) + q] - (\lambda + A_\phi)[P_0(\lambda - A_\phi) - q]e^{-2\lambda t}}. \tag{25}$$

If the system is *observable*, i.e. *(A,C)*: Observable Pair, then the RDE has a positive-definite, symmetric solution for an arbitrary positive-definite initial value of matrix $P_0 > 0$;

$$P(t) > 0 \quad p.d., \quad P(t) = P^T(t) \in R^{n\times n}, \quad \forall t > 0. \tag{26}$$

The need to solve Riccati equation is perhaps the greatest single cause of anxiety and agony on the part of people faced with implementing Kalman filter. Because there is no general formula for solving higher order polynomials equations (i.e., beyond quartic), finding closed-form solutions to algebraic Riccati equations by purely algebraic means is very rigorous. Thus, it is necessary to employ numerical solution methods. Numbers do not always provide us as much insight into the characteristics of the solution as formulas do, but readily amenable for most problems of practical significance.

3.2. Numerical example – An expendable launch vehicle (ELV) autopilot

This problem is taken from Aliyu et al, and it is significant for modeling and simulating an ELV autopilot problem in Matlab/Simulink®. It solves the symmetrical RDE:

$$\dot{P}(t) - AP(t) + P(t)A - P(t)CR_N^{-1}CP(t) + GQ_NG^T, \quad P_0 = I. \tag{27}$$

Where,

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 14.7805 & 0 & 0.01958 \\ -100.858 & 0 & -0.1256 \end{bmatrix}, \quad Q_N = \begin{bmatrix} 1.1\times10^{-3} & 0 & 0 \\ 0 & 1.1\times10^{-3} & 0 \\ 0 & 0 & 1.1\times10^{-3} \end{bmatrix},$$

$$R_N = \begin{bmatrix} 4.0\times10^{-6} & 0 & 0 \\ 0 & 4.0\times10^{-6} & 0 \\ 0 & 0 & 4.0\times10^{-6} \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

and

$$G = \begin{bmatrix} 0 & 14.7805 & -94.8557 \end{bmatrix}^T.$$

3.3. Numerical Methods and Problem Solving Environment (PSE), for Ordinary Differential Equations

In the last decade, two distinct directions have emerged in the way Ordinary Differential Equation (ODE) solvers software is utilized. These include Large Scale Scientific Computation and PSE. Most practicing engineers and scientists, as well as engineering and science students use numerical software for problem solving, while only a few very specific research applications require large scale computing.

MATLAB® provides several powerful approaches to integrate sets of initial value, Ordinary Differential Equations. We have carried out an extensive study of the requirements. For the simulation of the autopilot problem, we have used the mathematical development environment Matlab/Simulink®. Matlab/Simulink® was chosen, as it is widely used in the field of numerical mathematics and supports solving initial value ordinary differential equations like the type we have in (27) with easy.

From the humble beginnings of Euler's method, numerical solvers started relatively simple and have evolved into the more complex higher order Taylor methods and into the efficient Runge-Kutta methods. And the search for more efficient and accurate methods has led to the more complicated variable step solvers.

3.3.1. One step solver

For solving an initial value problem

$$y' = f(x,y), \quad y(x_0) = y_0, \tag{28}$$

a numerical method is needed. One step solvers are defined by a function $\Phi(x,y,h;f)$ which gives approximate values $y_i := y(x_i)$ for the exact solution $y(x)$:

$$y_{i+1} := y_i + h\Phi(x_i, y_i, h; f), \tag{29}$$

$$x_{i+1} := x_i + h, \tag{30}$$

where, h denotes the step size. In the following let x and y be arbitrary but fixed, and $z(t)$ is the exact solution of the initial value problem

$$z'(t) = f(t, z(t)), \quad z(x) = y \tag{31}$$

with the initial values x and y. Then the function

$$\Delta(x,y,h,f) := \begin{cases} \dfrac{z(x+h)-y}{h} & h \neq 0 \\ f(x,y) & h = 0 \end{cases} \tag{32}$$

describes the differential quotient of the exact solution *z(t)* with step size *h*, whereas *Φ(x,y,h;f)* is the differential quotient of the approximated solution with step size *h*. The difference $\tau = \Delta - \Phi$ is the measure of quality of the approximation method and is denoted as local discretization error.

In the following, $F_N(a,b)$ is defined as the set of all functions f, for which exist all partial derivations of order N on the area

$$S = \left\{ x, y \mid a \le x \le b, y \in \Re^n \right\} \quad a,b \text{ finite,} \tag{33}$$

where they are continuous and limited.

One step solvers must fulfill

$$\lim_{h \to 0} \tau(x, y, h; f) = 0. \tag{34}$$

This is equivalent to

$$\lim_{h \to 0} \Phi(x, y, h; f) = f(x, y). \tag{35}$$

If this condition holds for all $x \in [a,b]$, $y \in F_1(a,b)$ then Φ and the corresponding one step method are called *consistent.* Thus, the one step method is of *order p*, if

$$\tau(x, y, h; f) = 0(h^p), \tag{36}$$

holds for all $x \in [a,b]$, $y \in R$, $f \in F_p(a,b)$. The global discretization error

$$e_n(X) := y(X) - y_n \quad X = x_n \text{ fix, } n \text{ variable} \tag{37}$$

is the difference between exact solution and the approximated solution. The one step method is denoted as *convergent*, if:

$$\lim_{n \to \infty} \left\| e_n(X) \right\| = 0. \tag{38}$$

Theorem: Methods of order $p > 0$ are convergent and it holds

$$e_n(X) = 0(h^p). \tag{39}$$

This means that the order of the global discretization error is equal to the order of the local discretization error. The crucial problem concerning one-step methods is the choice of the step size *h*. If the step size is too small, the computational effort of the method is unnecessary high, but if the step size is too large, the global discretization error increases. For initial values x_0, y_0 a step size as large as possible would be chosen, so that the global discretization error is below a boundary ε after each step. Therefore, a step size control is necessary.

3.3.2. Explicit Euler

The most elementary method of solving initial value problems is the explicit Euler. The value of y_{i+1} can be calculated the following way:

$$y_{i+1} = y_i + h \cdot f(x_i, y_i) \tag{40}$$

The explicit Euler calculates the new value y_{i+1} by following the tangent at the old value for a distance of h. The slope of the tangent is given by the value of $f(x_i, y_i)$. The explicit Euler uses no step size control; the step size h is fixed. Therefore, it is only useful in special cases, where the function to integrate is pretty flat. Nevertheless, it is very easy to implement and calculates very fast, so it can be a good choice.

3.3.3. Runge-Kutta method

The Runge-Kutta methods are a special kind of one-step solvers, which evaluate the right side in each step several times. The intermediate results are combined linearly. The general discretization schema for one-step of a Runge-Kutta method is

$$y_1 = y_0 + h(b_1 K_1 + b_2 K_2 + \cdots + b_a K_a), \tag{41}$$

with corrections

$$K_i = f(x_0 + c_i h, y_0 + h\sum_{j=1}^{i-1} a_{ij} K_j), \quad i = 1, \cdots s. \tag{42}$$

The coefficients are summarized in a tableau, the so-called Butcher-tableau, see figure 2.

$$\begin{array}{c|cccc}
c_1 & 0 & & & \\
c_2 & a_{21} & \ddots & 0 & \\
\vdots & \vdots & \ddots & \ddots & \\
c_s & a_{s1} & \dots & a_{ss-1} & 0 \\
\hline
 & b_1 & b_2 & \dots & b_s
\end{array}$$

Figure 2. Butcher-tableau.

3.4. Step size control

The Runge-Kutta methods use an equidistant grid, but this is for most applications inefficient. A better solution is to use an adaptive step size control. The grid has to be chosen so that

- a given accuracy of the numerical solution is reached
- the needed computational effort is minimized.

As the characteristics of the solution are a priori unknown, a good grid structure cannot be chosen before the numerical integration. Instead, the grid points have to be adapted during the computation of the solution. Trying to apply this to Runge-Kutta methods lead to the following technique:

To create a method of order p (for y_{i+1}), it is combined with a method of order $P+1$(for $\hat{y}_{k+1}$).

This method for y_{i+1} is called the embedded method. The idea of embedding was developed by Fehlberg and methods using this technique therefore are called *Runge-Kutta-Fehlberg* methods. This leads to a modified Butcher-tableau (see figure 3). The new step size is calculated with

$$h_{new} = h \sqrt[p+1]{\frac{\varepsilon}{\|y - \hat{y}\|}}, \tag{43}$$

where ε denotes the tolerance.

$$\begin{array}{c|c} c & A \\ \hline & b^T \\ \hline & \hat{b}^T \end{array} \quad \begin{array}{l} \longrightarrow \; y_1 = y_0 + h \sum_{i=1}^{s} b_i K_i \\ \\ \longrightarrow \; \hat{y}_1 = y_0 + h \sum_{i=1}^{\hat{s}} \hat{b}_i K_i \end{array}$$

Figure 3. Modified Butcher-tableau for embedded Runge-Kutta-methods.

3.4.1. Error control and variable step size

The main concern with numerical solvers is the error made when they approximate a solution. The second concern is the number of computations that must be performed. Both of these can be addressed by creating solvers that use a variable step size in order to keep the error within a specified tolerance. By using the largest step size allowable while keeping the error within a tolerance, the error made is reduced.

The way to keep the error under control is to determine the error made at each step. A common way to do this is to use two solvers of orders p and $p+1$, as earlier explained. Any approximations made of order p will have an error no larger than the value of the $p + 1$ term. This leads us to take the difference of the two solvers to find the value of the error term.

Since the downside of using two distinct methods is a dramatic increase in computations, another method is typically used. An example of this method is the Rung-Kutta-Fehlberg Algorithm. The Rung-Kutta-Fehlberg combines Rung-Kutta methods of order four and order five into one algorithm. Doing this reduces the number of computations made while returning the same result.

3.4.2. Dormand-Prince method

The *Dormand-Prince* method is a member of the *Runge-Kutta-Fehlberg* class with order 4(5).It means that the method has order 5 and the embedded method has order 4. This is described by the following equations:

$$
\begin{aligned}
y(x_0+h) &= y_0 + h\sum_{k=0}^{4} b_k f_k(x_0, y_0; h) \\
\hat{y}(x_0+h) &= y_0 + h\sum_{k=0}^{5} \hat{b}_k f_k(x_0, y_0; h) \\
f_k &= f(x_0 + c_k h, y_0 + h\sum_{t=0}^{k-1} a_{ki} f_i)
\end{aligned}
\tag{44}
$$

The coefficients from Dormand and Prince can be seen in figure 4.

$$
\begin{array}{c|ccccccc}
0 & & & & & & & \\
\frac{1}{5} & \frac{1}{5} & & & & & & \\
\frac{3}{10} & \frac{3}{40} & \frac{9}{40} & & & & & \\
\frac{4}{5} & \frac{44}{45} & \frac{-56}{15} & \frac{32}{9} & & & & \\
\frac{8}{9} & \frac{19372}{6561} & \frac{-25360}{2187} & \frac{64448}{6561} & \frac{-212}{729} & & & \\
1 & \frac{9017}{3168} & \frac{-355}{33} & \frac{46732}{5247} & \frac{49}{176} & \frac{-5103}{18656} & & \\
1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & \frac{-2187}{6784} & \frac{11}{84} & \\
\hline
 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & \frac{-2187}{6784} & \frac{11}{84} & 0 \\
\hline
 & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & \frac{-92097}{339200} & \frac{187}{2100} & \frac{1}{40}
\end{array}
$$

Figure 4. Butcher-tableau for Dormand-Prince-method.

For solving an initial value problem like the one we have in (27) Matlab was chosen, as it is widely used in the field of numerical mathematics and supports solving ordinary differential equations. Moreover, it is possible to visualize the simulation results of the autopilot. In our program we used the *ode45*, a standard solver included in Matlab/Simulink. The solver *ode45* implements the method of *Dormand-Prince*, which is a member of the class of *Runge-Kutta-Fehlberg* methods. More specifically, the *Dormand–Prince* method uses six function evaluations to calculate fourth- and fifth-order accurate solutions. The difference

between these solutions is then taken to be the error of the (fourth-order) solution. This error estimate is very convenient for adaptive step size integration algorithms. This adaptive step-size control algorithm monitors the estimate of the integration error, and reduces or increases the step size of the integration in order to keep the error below a specified threshold. The accuracy requested is that both the relative and absolute (maximal) errors be less than the truncation error tolerance. In MATLAB, both relative (*RelTol*) and absolute (*AbsTol*) tolerances can be specified. The default values (that were used in solving the problem) are *RelTol*= 0.001 and *AbsTol*= 10^{-6} .We intend to integrate the DRE from *t=0* up to *t=1.* The Simulink model for the DRE is as shown in Figure 5.

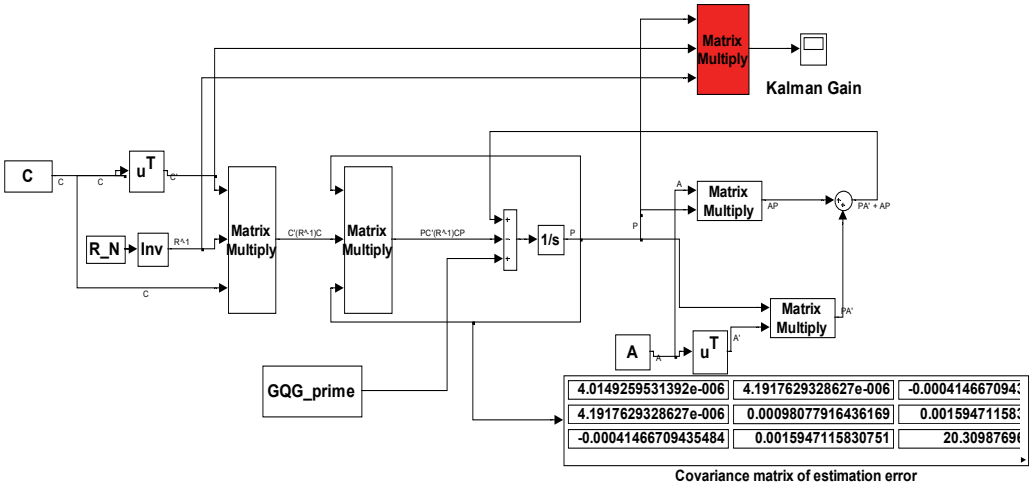


Figure 5. Simulink model for matrix Differential Riccati Equation.

After successfully simulating the above model, it was used to design an Linear Quadratic Gaussian (LQG) autopilot with the following Linear Quadratic Regulator (LQR) Characteristics;

$$Q = \begin{bmatrix} 2.5 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 173.6 \end{bmatrix}, \text{ and } R = 0.1. \tag{45}$$

For this autopilot, the Kalman-Bucy filter model was implemented as shown in Fig.6. It should be noted that if for any reason the initial condition P_0 in (27) is taking as a matrix with entries less than 1, then initial covariance matrix could be used as a tuning parameter to meet a specific *time response* characteristics as was presented in Aliyu, et al.

4. Filter performance

The dependent variable in the Riccati differential equation of the Kalman-Bucy filter is the covariance matrix of the *estimation error,* defined as the difference between the estimated

state vector *x-hat* and the true state vector x. Matrix P is a matrix of covariance of an error estimate of the state vector x . The initial value of which is chosen as

$$P_0 = E\{[x(t) - \hat{x}(t)][x(t) - \hat{x}(t)]^T\} \quad \text{At } t \to \infty. \tag{46}$$

The state error covariance matrix is $n \times n$ and symmetric, and must remain positive definite to retain filter stability. Diagonal elements of this matrix are variances of errors of the estimations for corresponding components of the state vector. These also serve as a definition of accuracy for the estimation.

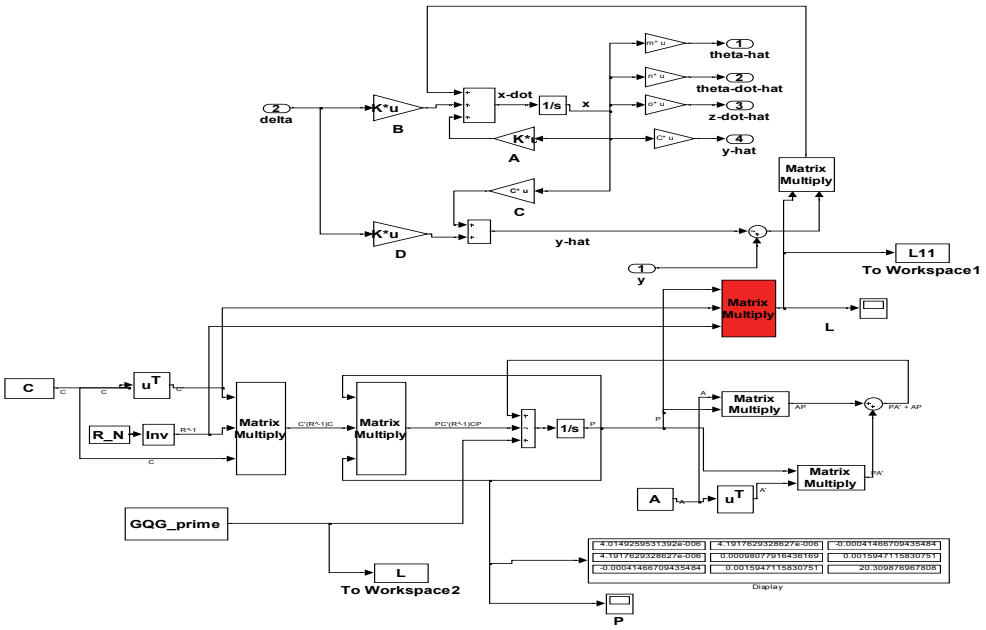


Figure 6. Simulink model for Kalman-Bucy filter with a RDE.

The solution of the matrix Riccati equation was found to provide a quantitative measure of how well the state variables can be estimated in terms of mean-squared estimation errors. Therefore, the matrix Riccati equation from the Kalman filter was soon recognized as a practical model for predicting the performance of sensor systems, and it became the standard model for designing aerospace sensor systems to meet specified performance requirements. More importantly, covariance analysis is crucial in exploring what-if scenarios with new measurement sources.

Note that in (27) we increase estimation uncertainty by adding in process noise and we decrease estimation uncertainty by the amount of information (R^{-1}) inherent in the measurement.

For an initial guess for the value of covariance, a very large value could be selected if one is using a very poor sensor for measurement. This makes the filter very conservative. Converse is the case if very good sensors are used for measurement.

The LQG autopilot simulation for tracking a pitch angle of 3 degrees (0.05rads) with the observer as designed in Fig.6 gave the result presented in Fig.7. It is interesting to note that the time response characteristics of the simulated autopilot in Fig. 7 meets all the design specifications of; *percentage overshoot* less than 10 percent; *settling time of less* than 4 seconds; *rise time* of less than 1 second and *steady state error* of less than 2 percent.

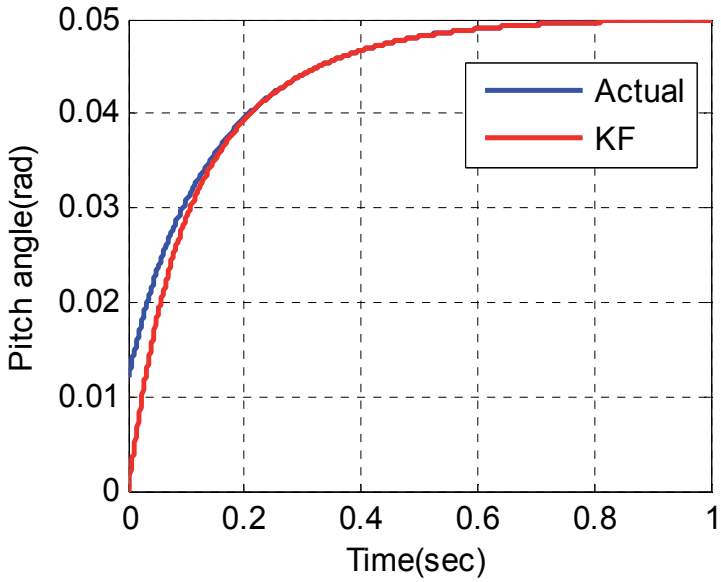


Figure 7. Set-point command tracking of LQG autopilot for a RDE solution to Kalman gain.

The numerical result at *t=1* for the covariance matrix with respect to Fig.5 is given in (47) and hence the associated Kalman gain is given in (48). The Kalman gain harnessed at this point urged us to re-design the Kalman filter model as shown in Fig. 8 for our LQG autopilot.

$$P = \begin{bmatrix} 5.228197174\times10^{-6} & 4.261615461\times10^{-6} & -0.000394471297 \\ 4.261615461\times10^{-6} & 0.0009807093782 & 0.0006722035747 \\ -0.0003944771297 & 0.0006722035747 & 8.750905161 \end{bmatrix}, \tag{47}$$

$$L = \begin{bmatrix} 1.30704929362596 & 1.06540386549228 & 0 \\ 1.06540386549228 & 245.177344558508 & 0 \\ -98.6192824463156 & 168.050893692856 & 0 \end{bmatrix}. \tag{48}$$

A further investigation was carried out-the single point value of Kalman filter gain given in (48) was used to implement the Kalman filter algorithm as popular represented in most textbooks-as a constant gain. For which, the Kalman-Bucy model in Fig. 8 was developed.

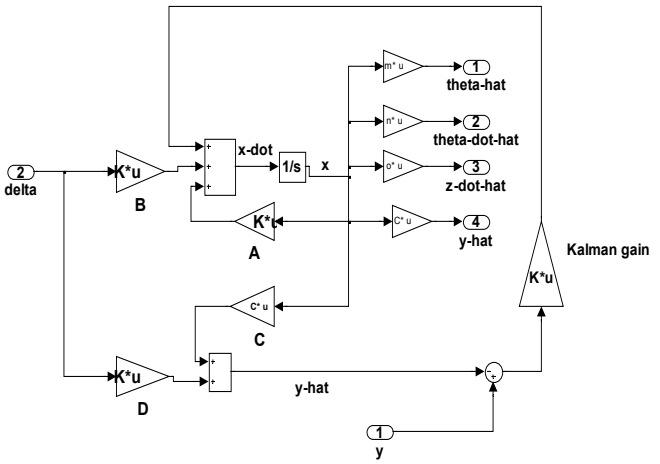


Figure 8. Simulink Model of Kalman filter model for a constant gain value of Kalman gain.

Hence, the same LQG autopilot was simulated with the Kalman filter based observer as shown in Fig.8. Simulation result for the system is as shown in Fig. 9. It is also interesting to note that all the time response characteristics as earlier mentioned were met. Though, the *LQR* controller could bring the ELV to a *settling time* at about 2 seconds.

The Matlab in-built command function *[K,P,E]=lqr(A,B,C,D,Q,R)* was used to obtain the solution for the design of the LQR controller. Where, *K* is the controller gain, *P* is the associated solution to the Algebraic Riccati Equation of the controller design and *E*, the closed-loop eigenvalues of the plant dynamics. Note, that for LQR design the pair *(A,B)* must be *controllable* then, a state feedback control law can be constructed to arbitrarily locate the closed-loop eigenvalues.

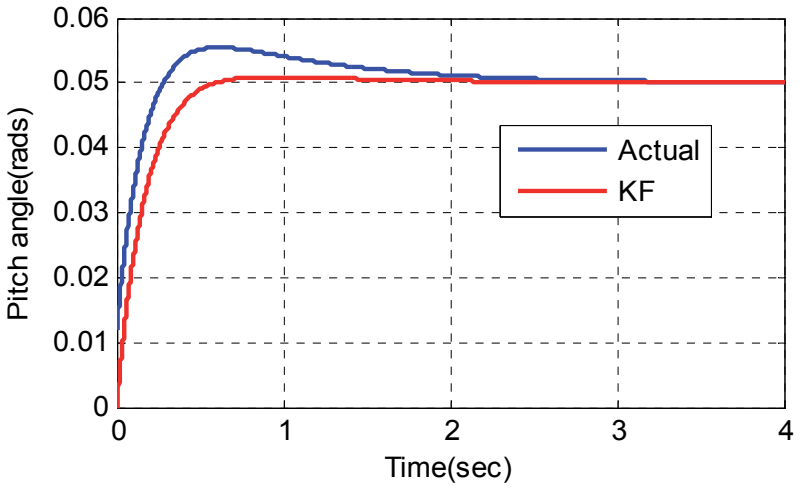


Figure 9. Set point command tracking of ELV autopilot for a Kalman gain obtained by evaluating covariance matrix to a Riccati Differential Equation at *t=1sec.*

5. Algebraic Riccati equation

Assume that the Riccati differential equation has an asymptotically stable solution for *P(t)*:

$$\lim_{t\to\infty} P(t) = P_\infty. \tag{49}$$

Then the time derivative vanishes

$$\lim_{t\to\infty} \frac{dP(t)}{dt} = 0. \tag{50}$$

Substituting this into the (6) yields

$$AP + PA^T + GQG^T - PC^T R^{-1} CP = 0. \tag{51}$$

This is called the Algebraic Riccati Equation. This is a nonlinear matrix equation, and need a numerical solver to obtain a solution for P_∞.

Consider a scalar case: $P_\infty \in R^{1\times I}$, A_ϕ ,C, Q, R ,$G \in R^{1\times I}$. The Algebraic Riccati Equation can be solved analytically

$$\frac{C^2}{R} P_\infty^2 - 2A_\phi P_\infty - G^2 Q = 0,$$

$$P_\infty = \frac{R}{C^2}\left\{ A_\phi \pm \sqrt{A_\phi^2 + \frac{Q}{R} C^2 G^2} \right\}. \tag{52}$$

From (52) there exist two solutions; one positive and the other negative, corresponding to the two values for the signum (±). There is no cause for alarm. The solution that agrees with (52) is the non-negative one. The other solution is non-positive. We are only interested in the non-negative solution, because the variance *P* of uncertainty is, by definition, non-negative. Thus, taking the positive solution

$$\lim_{t\to\infty} P(t) = P_\infty = \frac{R}{C^2}\left\{ A_\phi + \sqrt{A_\phi^2 + \frac{Q}{R} C^2 G^2} \right\} \tag{53}$$

For the numerical example at hand, the Kalman gain for this problem is easily solved with the following Matlab command *[L, P,E]=lqe(A,G,C,Q,R)*. This command solves a continuous time Algebraic Riccati Equation associated with the described model. Where, *L*, is the Kalman gain, *P*, the covariance matrix, and *E*, the closed-loop eigenvalues of the observer. The numeric values are as follows respectively:

$$L = \begin{bmatrix} 0.999999834557877 & 1.0005722536001 & 0 \\ 1.00057522536001 & 23530.07308334456 & 0 \\ -12.5039701153711 & -150868.724947992 & 0 \end{bmatrix}, \tag{54}$$

$$P = \begin{bmatrix} 3.99999938\times10^{-6} & 4.0023\times10^{-6} & -5.0016\times10^{-5} \\ 4.0023\times10^{-6} & 0.0941\times10^{-6} & -0.6035 \\ -5.0016\times10^{-6} & -0.6035 & 673.4535 \end{bmatrix}, \tag{55}$$

and

$$E = \begin{bmatrix} -23530.19862489 \\ -1.00000017144017 \\ -5.81187094955956\times10^{-5} \end{bmatrix}. \tag{56}$$

Based, on the result in (54), the LQG autopilot was simulated with the Kalman-Bucy filter state observer as modeled in Fig. 8. The result obtained from this was used in designing an LQG controller for the case of an ELV during atmospheric ascent. This seeks to track and control a pitch angle of 3 degrees (0.05rads) and the result is as shown Figure 10. Though, in this case the controller could bring the ELV to a *Settling Time* at 3 seconds. A minute further, compared to that obtained in Fig. 9. In both cases a negligible difference in *Percentage Overshoot* was observed.

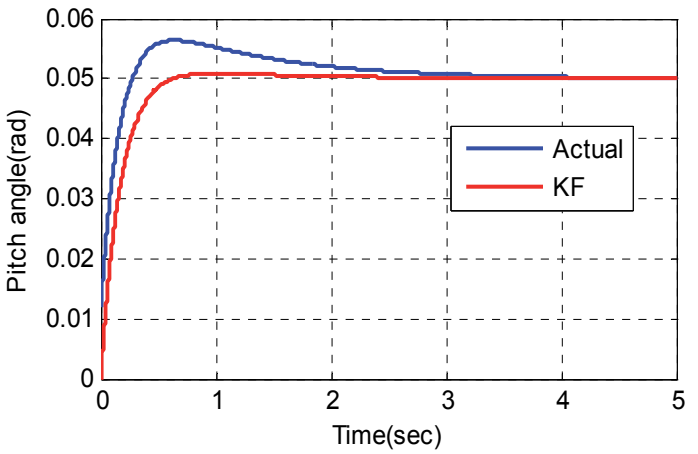


Figure 10. Set-point command tracking of LQG autopilot for an Algebraic Riccati Equation's solution to Kalman gain.

6. Comparative analysis

It can be clearly seen from figure 7 that the result of applying Kalman gain in the LQG problem of an ELV is most suitable by solving the associated Riccati Equation in its differential form (RDE). All time response characteristics were met within a second and with a zero *percent overshoot*! Though, issues might arise when hardware implementation is to be carried out. This basically will be due to the computational demand that will be placed on the selected micro-controller. In view of that, if we choose to design the

Kalman filter in the traditional manner but still not solving the associated Riccati equation as an algebraic one but as a differential one as done earlier and then harvesting the value of the Kalman gain at some specific point (in this research, we chose 1 second). Then, applying the Kalman gain, as a constant gain throughout the regime of simulation (RDE@1sec). It is obvious that for this example, this result still out performs that obtained from solving an Algebraic Riccati Equation (ARE). Actually, it could be clearly seen in Fig. 9 that the *settling time* is 2 second and in Fig. 10 is 3 seconds as the case applies. Figure 11 tries to give a holistic view of the three cases considered in this research for perusal.

Though one might be tempted to look at the difference in *settling time* of 1 second between the case of harvesting the Kalman gain value after solving a RDE at *t=1sec.*, With that of ARE as negligible or insignificant. On the contrary, considering an aerospace vehicle like the ELV, moving with a speed rage of *Mach* number *0.8-1.2* (transonic). One will be force to rethink. Let alone, compared to a *settling time* of 0.7sec obtained when Kalman gain is obtained from solving a RDE.

It is of paramount interest to add here that the plant and observer dynamics for all cases explored in this research gave a dynamic system with stable poles (*separation principle*). Contrary to that obtained in the paper Aliyu et al and in the book Aliyu Bhar Kisabo.

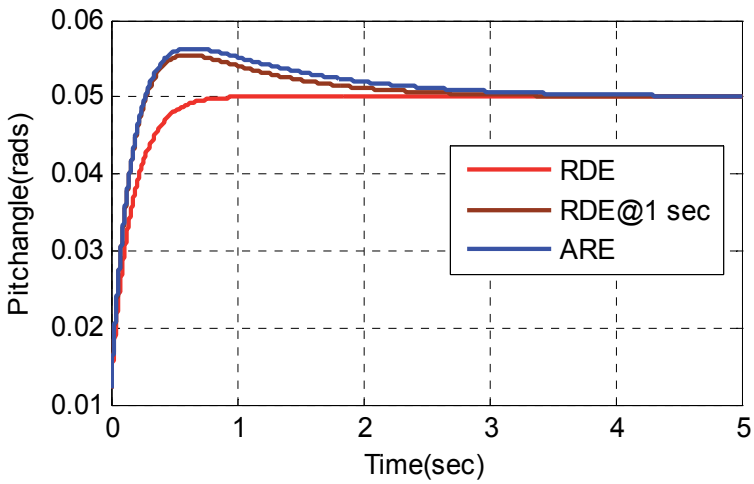


Figure 11. Compared results of the three approaches to the solution of Kalman gain as applied to an autopilot.

7. Conclusion

It can be clearly seen in Figure 6, that the synthesized LQG autopilot, with Kalman gain obtained by solving an Algebraic Riccati Equation (ARE) has 6 *percent overshoot* and a *settling time* of 3seconds. While, that in Fig. 3, is most preferred in all *time-domain* characteristics-zero *percentage overshoot* and *settling time* of 0.7second. This is the result of implementing

Kalman gain as a solution to a Riccati Differential Equation (RDE). Thus, the solution to *Riccati Differential Equation* for the implementation of Kalman filter in LQG controller design is the most optimal for pitch plane control of an ELV in the boast phase.

It is required that after designing Kalman filter, the accuracy of estimation is also assessed from the covariance matrix. It could be seen that both cases gave a very good estimation (very small covariance). Though, that of ARE gave a much smaller value. This has less significance to our research since we are majorly interested in the time response characteristic of the controlled plant. MATLAB 2010a was used for all the simulations in this paper.

Author details

Bhar K. Aliyu, Charles A. Osheku, Lanre M.A. Adetoro and Aliyu A. Funmilayo
Federal Ministry of Science and Technology (FMST), National Space Research & Development Agency (NASRDA), Center For Space Transport & Propulsion (CSTP) Epe, Lagos, Nigeria

Acknowledgement

The authors will like to specially appreciate the Honourable *Minister* of Science and Technology, Prof. Ita Okon Bassey Ewe, for his special intereste in the Research and Development activities at National Space Research and Development Agency (NASRDA). His support has been invaluable. Also, in appreciation is the *Director-General* of NASRDA, Dr. S. O Mohammed for making CSTP a priority agency among others Centres of NASRDA. We also thank Dr. Femi A. Agboola, *Director*, Engineering and Space Systems (ESS) at NASRDA, for his meaningful suggestions and discussions.

8. References

Aliyu Bhar Kisabo. (2011). *Expendable Launch Vehicle Flight Control; Design & Simulation With Matlab/Simulink*, ISBN 973-3-8443-2729-8, Germany.

L. F. Shampine, I. Gladwell, S. Thompson (2003). *Solving ODEs with MATLAB*, ISBN 978-0-511-07707-4, USA.

Robert H. Bishop. (2002). *The mechatronics Handbook* , Second Edition, ISBN 0-8493-0066-5, Boca Raton, London, New York, Washington D.C

Mohinder S Grewal.; & Angus P. Anderson. (2001). *Applications of kalman Filtering; Theory & practice Using Matlab,* Second Edition, ISBN 0-471-26638-8, New York, Chichester, Weinhem, Brisban,Sigapore,Toronto

Aliyu Bhar Kisabo et al (2011). Autopilot Design for a Generic Based Expendable Launch Vehicle, Using Linear Quadratic gaussian (LQG) Control. *European Journal of Scientific Research* , Vol.50, No.4, (February 2011), pp. 597-611, ISSN 1450-216X

Reza, Abazari. (2009). Solution of Riccati Type Differential Equations Using Matrix Differential Transform. *Journal of Applied Mathematics & Informatics,* Vol.27, No.5-6, (December 2009), pp. 1133-1143

Chapter 5

Numerical Simulation of the Frank-Kamenetskii PDE: GPU vs. CPU Computing

Charis Harley

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46463

1. Introduction

The efficient solution of the Frank-Kamenetskii partial differential equation through the implementation of parallelized numerical algorithms or GPUs (Graphics Processing Units) in MATLAB is a natural progression of the work which has been conducted in an area of practical import. There is an on-going interest in the mathematics describing thermal explosions due to the significance of the applications of such models - one example is the chemical processes which occur in grain silos. Solutions which pertain to the different geometries of such a physical process have different physical interpretations, however in this chapter we will consider the Frank-Kamenetskii partial differential equation within the context of the mathematical theory of combustion which according to Frank-Kamenetskii [16] deals with the combined systems of equations of chemical kinetics and of heat transfer and diffusion. A physical explanation of such a system is often a gas confined within a vessel which then reacts chemically, heating up until it either attains a steady state or explodes.

The focus of this chapter is to investigate the performance of the parallelization power of the GPU vs. the computing power of the CPU within the context of the solution of the Frank-Kamenetskii partial differential equation. GPU computing is the use of a GPU as a co-processor to accelerate CPUs (Central Processing Units) for general purpose scientific and engineering computing. The GPU accelerates applications running on the CPU by offloading some of the compute-intensive and time consuming portions of the code. The rest of the application still runs on the CPU. The reason why the application is seen to run faster is because it is using the extreme parallel processing power of the GPU to boost performance. A CPU consists of 4 to 8 CPU cores while the GPU consists of 100s of smaller cores. Together they operate to crunch through the data in the application and as such it is this massive parallel architecture which gives the GPU its high compute performance.

The methods which will be investigated in this research are implicit methods, such as the Crank-Nicolson method (CN) and the Crank-Nicolson method incorporating the Newton method (CN_N) [26]. These algorithms pose a serious challenge to the implementation of

parallelized architecture as we shall later discuss. We will also consider Rosenbrock methods which are iterative in nature and as was indicated in Harley [22] showed drastically increased running times as the time period over which the problem was considered increased. Further pitfalls which arise when trying to obtain a solution for the partial differential equation in question when using numerical techniques is firstly the singularity which exists at $x = 0$. This complexity may be dealt with through the use of a Maclaurin expansion which splits the problem into two cases: $x = 0$ and $x \neq 0$.

The second hurdle is the nonlinear source term which may be dealt with using different techniques. In this chapter we will implement the Newton method which acts as an updating mechanism for the nonlinear source term and in so doing maintains the implicit nature of the scheme in a consistent fashion. While the incorporation of the Newton method leads to an increase in the computation time for the Crank-Nicolson difference scheme (see [22]) there is also an increase in the accuracy and stability of the solution. As such we find that the algorithms we are attempting to employ in the solution of this partial differential equation would benefit from the processing power of a GPU.

In this chapter we will focus on the implementation of the Crank-Nicolson implicit method, employed with and without the Newton method, and two Rosenbrock methods, namely ROS2 and ROWDA3. We consider the effectiveness of running the algorithms on the GPU rather than the CPU and discuss whether these algorithms can in fact be parallelized effectively.

2. Model

The steady state formulation of the equation to be considered in this chapter was described by Frank-Kamenetskii [16] who later also considered the time development of such a reaction. The reaction rate depends on the temperature in a nonlinear fashion, generally given by Arrhenius' law. This nonlinearity is an important characteristic of the combustion phenomena since without it the critical condition for inflammation would disappear causing the idea of combustion to lose its meaning [16]. Thus, in the case of a thermal explosion, the Arrhenius law is maintained by the introduction of the exponential term which acts as a source for the heat generated by the chemical reaction. As such we are able to write an equation modelling the dimensionless temperature distribution in a vessel as

$$\frac{\partial u}{\partial t} = \nabla^2 u + \delta e^{u/(1+\epsilon u)} \tag{1}$$

where u is a function of the spatial variable x and time t and the Frank-Kamenetskii parameter δ is given by

$$\delta = \frac{Q}{\lambda}\frac{E}{R{T_0}^2} r^2 \kappa \alpha e^{\left(-\frac{E}{RT_0}\right)}. \tag{2}$$

The value of the Frank-Kamanetskii parameter [16] δ is related to the critical temperature at which ignition of a thermal explosion takes place and is thus also referred to as the critical value. At values below its critical value δ_{cr} a steady state is reached for a given geometry and set of boundary conditions whereas an explosion ensues for values above it. The Laplacian operator takes the form

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{k}{x}\frac{\partial}{\partial x}, \qquad 0 < x < 1 \tag{3}$$

where k is indicative of the shape of the vessel within which the chemical reaction takes place: $k = 0, 1$ and 2 represent an infinite slab, infinite circular cylinder and sphere, respectively. The dimensionless parameter $\epsilon = \frac{RT_0}{E}$ is introduced as the critical activation parameter. To be able to speak of combustion the condition $\epsilon \ll 1$ must be satisfied due to the fact that the ambient temperature can normally be seen as much smaller in magnitude than the ignition temperature [16]. Equation (1) for $\epsilon = 0$ was derived by Frank-Kamenetskii [16]. Further work was done by Steggerda [31] on Frank-Kamenetskii's original criterion for a thermal explosion showing that a more detailed consideration of the situation is possible. For small x a solution was derived for the cylindrical system by Rice [27], Bodington and Gray [6] and Chambré [10]. While the steady state case - often termed the Lane-Emden equation of the second kind - has been considered extensively, the time dependent case is also of import and has been studied in [2], [32] and [33].

In this chapter we consider numerical solutions for equation (1) modelling a thermal explosion within a cylindrical vessel, i.e. $k = 1$. A thermal explosion occurs when the heat generated by a chemical reaction is far greater than the heat lost to the walls of the vessel in which the reaction is taking place. As such this equation is subject to certain boundary conditions given at the walls of the vessel. The appropriate boundary conditions for this problem are

$$u(x, 0) = 0, \tag{4}$$

$$\frac{\partial u}{\partial x}(0, t) = 0, \quad (a) \qquad u(R, t) = 0 \quad (b) \tag{5}$$

where $R = 1$ is the radius of the cylinder. The boundary conditions (4) and (5) imply that the temperature at the vessel walls is kept fixed and the solution is symmetric about the origin.

Frank-Kamenetskii [16] obtained a steady state solution to this problem with $\epsilon = 0$. Zeldovich et al. [34] considered similarity solutions admitted by (1) for $k = 1$ that exhibit blow-up in finite time. These kinds of solutions, while noteworthy, have limited significance due to the restricted form of the initial profiles compatible with the similarity solutions. These solutions correspond to very special initial conditions for the temperature evolution profile, limiting the degree to which results obtained in this manner are applicable. This disadvantage has been noted by Anderson et al. [3] while analytically investigating the time evolution of the one-dimensional temperature profile in a fusion reactor plasma. A solution which also models blow-up in finite time has been obtained by Harley and Momoniat [18] via nonlocal symmetries of the steady-state equation.

In Harley [21] a Crank-Nicolson- and hopscotch scheme were implemented for equation (1) subject to (4) and (5) where $\delta = 1$ and $\epsilon = 0$. The nonlinear source term was kept explicit when the Crank-Nicolson method was employed, as commented on by Britz et al. [9] in whose work the nonlinear term was incorporated in an implicit manner in a style more consistent with the Crank-Nicolson method. Britz et al. [9] implemented the Crank-Nicolson scheme with the Newton iteration and showed that it outperformed the explicit implementation of the nonlinearity as in [21] in terms of accuracy. However it does require more computer time as would be expected.

In recent work (see [22]) the Crank-Nicolson method was implemented with the Newton iteration as done by Britz et al. [9] by computing a correction set in each iteration to obtain

approximate values of the dependent variable at the next time step. The efficiency of the Crank-Nicolson scheme, hopscotch scheme (both of these methods were implemented with an explicit and then an implicit discretisation of the source term) and two versions of the Rosenbrock method were compared [22]. Using the `pdepe` function in MATLAB and the steady state solution obtained by Frank-Kamenetskii [16] as a means of comparison, it was found that the incorporation of the Newton method for the Crank-Nicolson- and hopscotch scheme led to increased running times as T, where $0 \leq t \leq T$, increased.

Furthermore, it was shown that while the Crank-Nicolson- and hopscotch method (with or without the implementation of the Newton method) performed well in terms of accuracy for $T = 0.3$ and 0.5, they were in fact able to outperform `pdepe` at $T = 4$. The Rosenbrock methods employed (ROS2 and ROWDA3) performed similarly with regards to accuracy, however showed almost an exponential increase in their running times as T increased, indicating that using the Crank-Nicolson- or hopscotch scheme may be more efficient. Thus, given that the Rosenbrock methods performed even poorer with regards to running time, it seems reasonable to suggest that implementing the Crank-Nicolson- or hopscotch scheme with a Newton iteration is most ideal. The Crank-Nicolson method using the Newton method as a means of maintaining the implicit nature of the source term in the difference scheme has been used by Anderson and Zienkiewicz [2]. In Harley [21] and Abd El-Salam and Shehata [1] the discretisation of the exponential terms were kept explicit, thereby removing the nonlinearity.

As a consequence of these findings and due to the complexity created by the nonlinear source term which serves a critical function in the model, further work regarding faster algorithms for the solution of such an equation are of interest. This chapter will not consider the hopscotch scheme directly as an appropriate method for the solution of the Frank-Kamenetskii partial differential equation due to work done by Feldberg [15] which indicated that for large values of $\beta = \frac{\triangle t}{\triangle x^2}$ the algorithm produces the problem of propagational inadequacy which leads to inaccuracies - similar results were obtained in [22]. Given the improved accuracy of the Crank-Nicolson method incorporating the Newton method [22] - the order of the error for this method is $O(\triangle t^2)$ which is only approximately the case for the Crank-Nicolson method without the Newton iteration incorporated [9] - it seems more fitting to consider an improvement in the computing time of this method. Hence a consideration of such an improvement on the algorithm's current running time will be the focus of this chapter. The means by which we wish to accomplish this is through the use of the the Parallel Computing Toolbox in MATLAB. It is hoped that this is the next step towards creating fast and effective numerical algorithms for the solution of a partial differential equation such as the one originating from the work of Frank-Kamenetskii [16].

3. Executing MATLAB on a GPU

The advantage of using the Parallel Computing Toolbox in MATLAB is the fact that it allows one to solve computationally and data-intensive problems using multicore processors, GPUs, and computer clusters. In this manner one can parallelize numerical algorithms, and in so doing MATLAB applications, without CUDA or MPI programming. Parallelized algorithms such as `parfor`, used within the context of what is usually a `for` loop, allows you to offload

work from one MATLAB session (the client) to other MATLAB sessions, called workers. You can use multiple workers to take advantage of parallel processing and in this way improve the performance of such loop execution by allowing several MATLAB workers to execute individual loop iterations simultaneously. In this context however we are not able to implement in-built MATLAB functions such as `parfor` due to the numerical algorithms which we have chosen to consider. The CN- and CN_N method, both implicit, loop through the index m until $t_0 + m\triangle t = T$. These iterative steps are not independent of each other, i.e to obtain data at the $m+1^{th}$ step the data at the mth step is required. In a similar fashion the ROS2 and ROWDA3 methods also iterate through dependent loops to obtain a solution. As such we attempt to run the code directly on the GPU instead of the CPU in order to decrease the running time of the algorithms.

The Parallel Computing Toolbox in MATLAB allows one to create data on and transfer it to the GPU so that the resulting GPU array can then be used as an input to enhance built-in functions that support them. The first thing to consider when implementing computations on the GPU is *keeping* the data on the GPU so that we do not have to transfer it back and forth for each operation - this can be done through the use of the `gpuArray` command. In this manner computations with such input arguments run on the GPU because the input arguments are already in the GPU memory. One then retrieves the results from the GPU to the MATLAB workspace via the `gather` command. Having to recall the results from the GPU is costly in terms of computing time and can in certain instances make the implementation of an algorithm on the GPU less efficient than one would expect. Furthermore, the manner in which one codes algorithms for GPUs is of vital importance given certain limitations to the manner in which functions of the Toolbox may be implemented (see [25]). More importantly however, is whether the method employed can allow for the necessary adjustments in order to improve its performance. In this chapter we will see that there are some problems with implementing the kind of algorithms considered here on the GPU.

In this chapter we are employing MATLAB under Windows 7 (64 bits) on a PC equipped with an i7 2.2 GHz processor with 32 GB of RAM.

3.1. Crank-Nicolson implicit scheme

We will implement the Crank-Nicolson method while maintaing the explicit nature of the nonlinear source term and also apply the method by computing a correction set in each iteration to obtain approximate values of the dependent variable at the next time step through the use of the Newton method [26]. The methodology will be explained briefly here; the reader is referred to [7–9] for clarification.

When implementing the Crank-Nicolson method we employ the following central-difference approximations for the second-and first-order spatial derivatives respectively

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{\triangle x^2}, \tag{6}$$

$$\frac{\partial u}{\partial x} \approx \frac{u_{n+1}^m - u_{n-1}^m}{2\triangle x} \tag{7}$$

while a forward-difference approximation

$$\frac{\partial u}{\partial t} \approx \frac{u_n^{m+1} - u_n^m}{\triangle t} \tag{8}$$

is used for the time derivative. We implement a Crank-Nicolson scheme by approximating the second-derivative on the right-hand side of (1) by the implicit Crank-Nicolson [12] approximation

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{n+1}^{m+1} - 2u_n^{m+1} + u_{n-1}^{m+1}}{2\triangle x^2} + \frac{u_{n+1}^m - 2u_n^m + u_{n-1}^m}{2\triangle x^2}. \tag{9}$$

In a similar fashion the first-derivative on the right-hand side becomes

$$\frac{\partial u}{\partial x} \approx \frac{u_{n+1}^{m+1} - u_{n-1}^{m+1}}{4\triangle x} + \frac{u_{n+1}^m - u_{n-1}^m}{4\triangle x}. \tag{10}$$

To impose zero-shear boundary conditions at the edges we approximate the spatial first-derivative by the central-difference approximation (7) which leads to the following condition

$$u_{-1}^m = u_1^m. \tag{11}$$

As mentioned before the boundary condition (5a) at $x_0 = 0$ can pose a problem for the solution of equation (1). One could discretise it directly as a forward difference formula, such as the three-point approximation $-3u_0^m + 4u_1^m - u_2^m = 0$, and add this to the set of equations to solve when using the Crank-Nicolson scheme. Alternatively one could use the more accurate symmetric approximation, $u_{-1}^m = u_1^m$, which introduces a 'fictitious point' at $x = -\triangle x$. This however, would lead to another problem due to the singularity in the differential equation at $x_0 = 0$. Instead we choose to overcome this difficulty by using the Maclaurin expansion

$$\lim_{x \to 0} \frac{1}{x} \frac{\partial u}{\partial x} = \left. \frac{\partial^2 u}{\partial x^2} \right|_{x=0} \tag{12}$$

which simplifies the equation for the case $x_0 = 0$. It has been noted by Britz et al. [9] that using (12) turns out to be more convenient and accurate. Due to the fact that the point $x_0 = 0$ would lead to a singularity in equation (1) we structure the code to account for two instances: $x = 0$ and $x \neq 0$. Using (12) for equation (1) we attain the following approximation

$$\frac{\partial u}{\partial t} = 2\frac{\partial^2 u}{\partial x^2} + e^u \tag{13}$$

to equation (1) at $x_0 = 0$. This approximation has been taken into account in the system given by (16) below. Such an approximation has been used in many numerical algorithms. In Crank and Furzeland [13], for instance, they presented a modified finite-difference method which eliminates inaccuracies that occur in the standard numerical solution near singularities. The approximation has also been used by Harley and Momoniat [19] to generate a consistency criteria for initial values at $x_0 = 0$ for a Lane-Emden equation of the second-kind. From the equation under consideration (1) an initial condition for $u(x,t)$ is obtained at $x_0 = 0$ giving

the following

$$(1+2\beta)\, u_0^{m+1} - 2\beta u_1^{m+1} - \triangle t \delta e^{u_0^{m+1}} = (1+2\beta)\, u_0^m + 2\beta u_1^m + \triangle t \delta e^{u_0^m} \tag{14}$$

as the initial difference scheme with $\beta = \frac{\triangle t}{\triangle x^2}$. Implementing the difference approximations discussed above we obtain the general numerical scheme

$$-\frac{\lambda_n}{2} u_{n-1}^{m+1} + (1+\beta)\, u_n^{m+1} - \frac{\gamma_n}{2} u_{n+1}^{m+1} - \triangle t \delta e^{u_n^{m+1}} = \frac{\lambda_n}{2} u_{n-1}^m + (1-\beta)\, u_n^m + \frac{\gamma_n}{2} u_{n+1}^m + \triangle t \delta e^{u_n^m} \tag{15}$$

where $x_n = n\triangle x$ and $\beta = \frac{\triangle t}{\triangle x^2}$ such that $\gamma_n = \beta\left(1 - \frac{1}{2n}\right)$ and $\lambda_n = \beta\left(1 + \frac{1}{2n}\right)$. This difference scheme (15), including the initial difference condition (14), form a system of equations which are to be solved iteratively.

As indicated by the boundary conditions (5a) and (5b) we consider the problem for $x \in [0,1]$ and $t \in [0,T]$. The domain $[0,1]$ is sub-divided into N equidistant intervals termed $\triangle x$, i.e. $0 = x_0 < x_1 < x_2 < \cdots < x_{N-1} < x_N$ where $x_{n+1} = x_n + \triangle x$. In a similar fashion the domain $[0,T]$ is sub-divided into M intervals of equal length, $\triangle t$, through which the scheme iterates. The system will iterate until $t_m + \triangle t = T$, i.e. for $M = T/\triangle t$ steps. The system generated by (15) can be written in compact form as

$$A\mathbf{u}^{m+1} = B\mathbf{u}^m + \triangle t \delta e^{\mathbf{u}^m} + \triangle t \delta e^{\mathbf{u}^{m+1}} \tag{16}$$

and is solved as follows

$$\mathbf{u}^{m+1} = (A)^{-1}\left(B\mathbf{u}^m + \triangle t \delta e^{\mathbf{u}^m} + \triangle t \delta e^{\mathbf{u}^{m+1}}\right). \tag{17}$$

The inverse of A is calculated using the \ operator in MATLAB which is more efficient than the `inv` function. The nonlinear term on the $m+1^{th}$ level is dealt with through an implementation of the Newton method [26] in an iterative fashion as done by Britz et al. [9] and discussed in [8]. The system $\mathbf{J}\delta\mathbf{u} = -\mathbf{F}(\mathbf{u})$ is solved where $\mathbf{F}$ is the set of difference equations created as per (16) such that $\mathbf{F}(\mathbf{u}) = 0$. The starting vector at $t = 0$ is chosen as per the initial condition (4) such that $\mathbf{u} = 0$. The Newton iteration converges within 2-3 steps given that changes are usually relatively small.

3.2. Rosenbrock method

We now consider two particular Rosenbrock methods, ROS2 and ROWDA3, as a means of comparison for the effectiveness of the methods discussed in the previous section. The Rosenbrock methods belong to the class of linearly implicit Runge - Kutta methods [11, 17]. They were used successfully for the numerical solution of non-electrochemical stiff partial differential equations, including equations of interest to electrochemistry. For further information regarding the particulars of such methods interested readers are referred to the numerical literature of [17, 28–30].

The reason for the use of the Rosenbrock methods in this paper is the ease with which they are able to deal with the nonlinear source term and the fact that no Newton iterations are

necessary. The advantages of these methods are great efficiency, stability and a smooth error response if ROS2 or ROWDA3 are used (see [4] for instance) and the ease with which they are able to handle time-dependent and/or nonlinear systems.

We consider equation (1) as the following system

$$\frac{du_n}{dt} == \begin{cases} \frac{(1+k)}{\triangle x^2}\left(2u_1 - 2u_0\right) + \delta e^{u_0} \text{ if } n = 0 \\ \frac{\gamma_n}{\triangle t}u_{n-1} - \frac{2}{\triangle x^2}u_n + \frac{\lambda_n}{\triangle t}u_{n+1} + \delta e^{u_n} \text{ if } n = 1, 2, ..., n-2 \\ -\frac{2}{\triangle x^2}u_{N-1} + \frac{\lambda_{N-1}}{\triangle t}u_{N-2} + \delta e^{u_{N-1}} \text{ if } n = N-1 \end{cases} \tag{18}$$

which along with $0 = u_N$ can be written in the compact form

$$\mathbf{S}\frac{d\mathbf{u}}{dt} = \mathbf{F}(t, \mathbf{u}) \tag{19}$$

where $\mathbf{S} = \text{diag}(1, 1, 1, ..., 1, 0)$ is the selection matrix containing zeros in those positions where the set of differential algebraic equations has an algebraic equation (i.e. zero on the left-hand side of (18)) and unity in those positions corresponding to the ordinary differential equations. The function $\mathbf{F}(t, \mathbf{u})$ can be written as: $\mathbf{F}(t, \mathbf{u}) = \mathbf{Ju} + \mathbf{s}$ where the matrix $\mathbf{J}$ is the Jacobian and the vector $\mathbf{s}$ arises from the constant terms of the set of differential algebraic equations. We can thus write $\mathbf{F}(t, \mathbf{u}) = \mathbf{Ju} + \mathbf{s}$ as

$$= \frac{1}{\triangle t}\begin{bmatrix} -2(1+k)\beta & 2(1+k)\beta & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & \gamma_1 & -2\beta & \lambda_1 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & \gamma_2 & -2\beta & \lambda_2 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \dots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & \gamma_{N-2} & -2\beta & \lambda_{N-2} & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & \gamma_{N-1} & -2\beta & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \\ u_N \end{bmatrix}$$

$$+ \begin{bmatrix} \delta e^{u_0} \\ \delta e^{u_1} \\ \delta e^{u_2} \\ \vdots \\ \delta e^{u_{N-2}} \\ \delta e^{u_{N-1}} \\ 0 \end{bmatrix} \tag{20}$$

such that

$$\mathbf{F}_u = \frac{1}{\triangle t}\begin{bmatrix} (1+k)\beta(2u_1 - 2u_0) + \delta e^{u_0} \\ \gamma_1 u_0 - 2\beta u_1 + \lambda_1 u_2 + \delta e^{u_1} \\ \gamma_2 u_1 - 2\beta u_2 + \lambda_2 u_3 + \delta e^{u_2} \\ \vdots \\ \gamma_{N-2}u_{N-3} - 2\beta u_{N-2} + \lambda_{N-2}u_{N-1} + \delta e^{u_{N-2}} \\ -2\beta u_{N-1} + \lambda_{N-1}u_N - 2 + \delta e^{u_{N-1}} \\ u_N \end{bmatrix}. \tag{21}$$

In order to implement the Rosenbrock methods a number s of $\mathbf{k}_i$ vectors are computed with s the order chosen. The general equation given by (22) is solved iteratively to obtain each vector $\mathbf{k}_i$ for all i specified which will then be used to update the vector $\mathbf{u}$ for the next time step. We use the notation employed in [7] for the general equation to be used to obtain the values for $\mathbf{k}_i$

$$-\frac{\mathbf{M}}{\beta}\mathbf{k}_i = -\frac{\triangle t}{\beta}\mathbf{F}\left(t + \varphi_i \triangle t, \mathbf{u} + \sum_{j=1}^{i-1} a_{ij}\mathbf{k}_j\right) - \frac{\mathbf{S}}{\beta}\sum_{j=1}^{i-1} c_{ij}\mathbf{k}_j - \frac{\kappa_i}{\beta}\triangle t^2 \mathbf{F}_t(t, \mathbf{u}) \tag{22}$$

where we define $M = \frac{\mathbf{S}}{\kappa} - \triangle t \mathbf{F}_u$ were the function $\mathbf{F}$ is applied at partly augmented t and $\mathbf{u}$ values and the time derivative $\mathbf{F}_t$ is zero in this case since the system does not include functions of time. Having calculated the s $\mathbf{k}_i$ vectors the solution is obtained from

$$\mathbf{u}_{m+1} = \mathbf{u}_m + \sum_{i=1}^{s} m_i \mathbf{k}_i \tag{23}$$

where the m_i are weighting factors included in the tables of constants specified for each method (see [4] and [7]).

In this chapter we implement the ROS2 and ROWDA3 methods though there are other variants of the Rosenbrock methods. Lang [24] described a L-stable second-order formula called ROS2. A third-order variant thereof is called ROWDA3 and described by Roche [28] and later made more efficient by Lang [23]. The latter is a method favoured by Bieniasz who introduced Rosenbrock methods to electrochemical digital simulation [4, 5]. For a more detailed explanation and discussion regarding the method and its advantages refer to [7]. The focus of the work done here is with regards to whether the Rosenbrock algorithms lend themselves toward parallelized implementation. It has already been noted that functions such as the `parfor` command cannot be used in this instance. It now remains to consider the method's performance when run on a GPU via the MATLAB Parallel Computing Toolbox.

4. Discussion of numerical results

The results noticed, as per Table 1, indicate the extent to which implementing the code on the GPU slows down overall performance of the CN, CN_N, ROS2 and ROWDA3 methods. The question is why this would be the case. In Table 1 the results for the different methods run on a CPU were obtained by running the code on one CPU only instead of all of those available to MATLAB on the computer employed. This was done to get a better understanding of the one-on-one performance between the processing units, and yet implementing the code on the GPU still led to poor performance.

To gain a better understanding of these results we consider the baseline structure for our CN code:

```
A = gpuArray(A);
B = gpuArray(B);
```

```
u0 = gpuArray(u0);
for m = 1 : T
    b = delta. * dt. * exp((1 + eps. * u0).\(u0));
    u0 = mldivide(A, (B * u0 + b));
end
```

In doing so, we realise that the main components thereof are matrix and elementwise vector operations. In order to understand why we are achieving the results we do (see Table 1) we run a few simple 'test'-codes to consider the speed taken by the CPU vs. the GPU to perform such elementary operations as `C\d` and `d.*f` where C is a matrix and d and f are vectors. In Figure 1 we see the speed of the CPU over the GPU computed as $\frac{CPU\ \ running\ \ time}{GPU\ \ running\ \ time}$. You will notice that as the size of the matrix and corresponding vector increases so too does the speed at which the GPU is able to compute `C\d` allowing it to overtake the CPU. This is what one would expect given that the GPU will only 'kick in' once the CPU is overloaded with data structures too large for it to compute effectively. Thus the efficiency in terms of running time of the code provided above is heavily dependent upon the size of the matrices A and B. At this juncture it is important to remember that we are considering the range $x \in [0, 1]$ with $\triangle x = 0.1$ which means that our A matrix is a 10×10 matrix and as such not large enough to give the GPU the chance to expose its ability to improve the performance of the algorithm. The reason for the choice in the size of the matrix for the problem considered is twofold: (1) it is due to the influence of the ratio $\beta = \triangle t / \triangle x^2$ which one usually tries to keep close to 1 for reasons of stability, and (2) the limitations of memory of the PC being used.

The next step in this evaluative process is to now consider the speed at which vector operations are performed. This was done in a similar fashion to the previous case by considering the speed taken by the CPU and GPU to perform the elementwise operation `d.*f` where d and f are vectors. The ratios of the speeds $\frac{CPU\ \ running\ \ time}{GPU\ \ running\ \ time}$ were also considered for the in-built function `arrayfun` which performs elementwise operations on all its inputs. It can clearly be seen in Figure 2 that the in-built function outperforms the normal $.*$ operation. What is interesting in this case is that the size of the vector required for the GPU to outperform the CPU is very large - we considered vectors of sizes between 200 000 and 201 000 as indicated. For smaller vector lengths the GPU is completely outperformed by the speed at which calculations are done on the CPU. As such, to improve the speed at which these vector calculations are performed we would either (1) have to diminish $\triangle x$ to the degree needed to obtain vectors of the required length (2) or be required to move the vectors from the GPU memory to the CPU memory every time calculations need to be made. The first approach would require a memory capacity beyond that of the computer used here and the second would greatly increase the running time of the algorithm and as such is not worth implementing.

As a means of further investigation we consider the CN code as a test case for the use of the `arrayfun` function. Obviously implementing this in-built function as follows

```
A = gpuArray(A);
B = gpuArray(B);
u0 = gpuArray(u0);
u0 = arrayfun(@myCrank, u0, A, B)
```

$\beta = 0.01$ and $T = 0.3$				
	CN	CN_N	$ROS2$	$ROWDA3$
CPU	7.8449e-05	2.7737e-04	2.3002e-05	9.1288e-05
GPU	0.0014	0.0960	0.0033	0.0063
$\beta = 0.01$ and $T = 5$				
	CN	CN_N	$ROS2$	$ROWDA3$
CPU	1.4783e-05	2.0018e-04	1.5354e-05	6.3574e-05
GPU	8.0940e-04	0.0047	0.0028	0.0047
$\beta = 2$ and $T = 5$				
	CN	CN_N	$ROS2$	$ROWDA3$
CPU	2.4573e-05	0.0033	1.7146e-05	6.8119e-05
GPU	0.0048	0.4731	0.0042	0.0073

Table 1. Running times per iteration of $\triangle t$ for the relevant methods implemented for $\triangle t = 0.0001$, $\triangle x = 0.1$, $\delta = 1$ and $\epsilon = 0$.

$\epsilon = 0.01$			
CN	CN_N	$ROS2$	$ROWDA3$
0.0137	0.0025	0.0059	0.0138
$\epsilon = 0.05$			
CN	CN_N	$ROS2$	$ROWDA3$
0.0133	0.0024	0.0067	0.0149
$\epsilon = 0.1$			
CN	CN_N	$ROS2$	$ROWDA3$
0.0146	0.0025	0.0057	0.0128
$\epsilon = 0.25$			
CN	CN_N	$ROS2$	$ROWDA3$
0.0145	0.0024	0.0059	0.0133

Table 2. The ratio $\frac{CPU \quad running \quad time}{GPU \quad running \quad time}$ for the relevant methods implemented for $\triangle t = 0.0001$, $\triangle x = 0.1$, $\delta = 1$ and $T = 0.3$.

$\delta = 0.5$			
CN	CN_N	$ROS2$	$ROWDA3$
0.0146	0.0012	0.0064	0.0150
$\delta = 1$			
CN	CN_N	$ROS2$	$ROWDA3$
0.0275	0.0026	0.0061	0.0160
$\delta = 2$			
CN	CN_N	$ROS2$	$ROWDA3$
0.0151	0.0030	0.0062	0.0151
$\delta = 3$			
CN	CN_N	$ROS2$	$ROWDA3$
0.0160	0.0042	0.0063	0.0140

Table 3. The ratio $\frac{CPU \quad running \quad time}{GPU \quad running \quad time}$ for the relevant methods implemented for $\triangle t = 0.0001$, $\triangle x = 0.1$, $\epsilon = 0$ and $T = 0.3$.

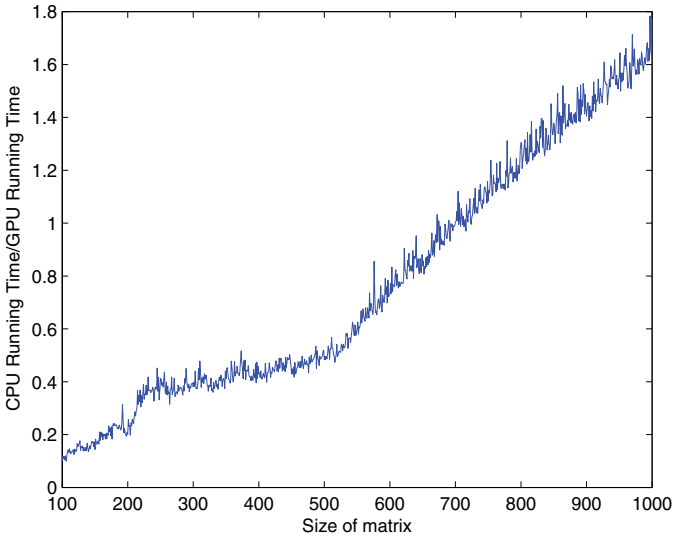


Figure 1. Plot showing the CPU Running Time/GPU Running Time for matrices and corresponding vectors of sizes 100 to 1000.

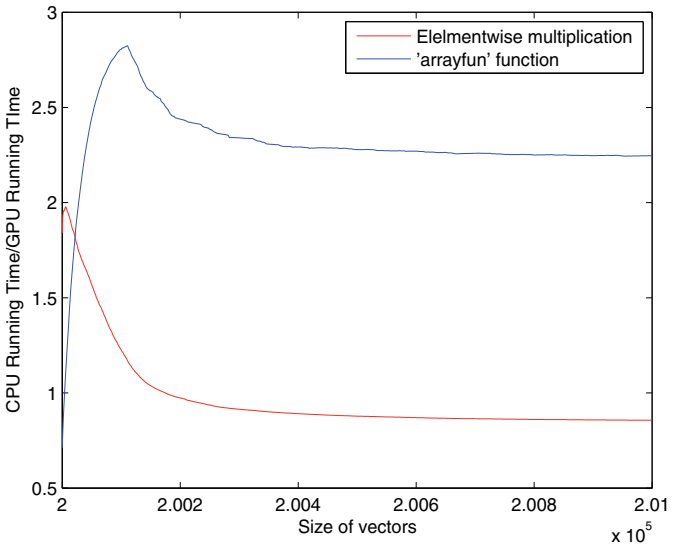


Figure 2. Plot showing the CPU Running Time/GPU Running Time for vectors of sizes 200 000 to 201 000.

where the `@myCrank` function performs the loop through m, instead of the code presented previously produces incorrect results. The results obtained do however support our findings that the `arrayfun` function is able to increase the speed with which elementwise operations are performed. In this instance `arrayfun` is computing on the GPU since the inputs are

all GPU array objects. We found for $T = 10$ with $\triangle t = 0.0001$ as we decreased $\triangle x$ that computing on the GPU was faster than doing so on the CPU: for $\triangle x = 0.1$ and 0.01 the ratios were $\frac{CPU\ \ running\ \ time}{GPU\ \ running\ \ time} = 1.5709$ and $\frac{CPU\ \ running\ \ time}{GPU\ \ running\ \ time} = 6.5906$ respectively. This makes sense given that smaller values of $\triangle x$ would increase the sizes of the matrices A and B and the vectors b and $u0$. As such, it seems likely that using a PC with a greater memory capacity would lead to the GPU outperforming the CPU by a large margin as $\triangle x$ decreases.

4.1. Influence of changing parameter values on the running time of the algorithms

Just a few brief comments upon the results obtained for CN, CN_N, ROS2 and ROWDA3 for varying values of ϵ and δ will be made in this section. Firstly we considered the schemes for $\delta = 1$ and $\epsilon = 0.01, 0.05, 0.1$ and 0.25 and then we also considered the case for $\epsilon = 0$ with $\delta = 0.1, 1, 2$ and 3. The reader will notice considering Tables 2 and 3 that there does not seem to be any noticeable trend to the results obtained. As such the values of ϵ and δ do not seem to have a meaningful impact on the speed at which the algorithms compute.

5. Concluding remarks

The implementation of numerical algorithms such as those considered in this chapter are widely used for the solution of many differential equations which model physical processes and applications. As such it is of vital importance that we be able to perform such calculations at high speed given the requirement of fine grids to improve accuracy. It is in this context that the use of GPUs becomes of prime importance. However it is not simply a matter of running the algorithm on the GPU - the method employed needs to lend itself to being adjusted in the required manner so that the parallel processing power of the GPU may be taken advantage of. Though we found that the numerical methods considered here were not entirely suited to being implemented on the GPU as we would have hoped we were able to explain why this was the case.

This work has investigated the effectiveness of matrix and elementwise operations when run on a GPU vs. a CPU and found that the speed taken to do such operations heavily relies on the choice of $\triangle x$. It was discovered that the introduction of the nonlinear source term is problematic due to the length of time taken to do elementwise calculations on the GPU. While matrix operations were also shown to be slow it was more specifically this aspect of the code which increased the running time.

We also discovered the power of the in-built function `arrayfun` which was able to improve upon the performance of the GPU with regards to computing time to the degree that it outperformed the CPU even for a grid with 'large' $\triangle x$, i.e. small matrices and vectors within the computations. As the grid became finer the performance of the GPU over the CPU improved, indicating the impact of the size of the matrices upon which computations are being performed and the degree to which `arrayfun` is able to improve computations occurring on the GPU. Thus, the manner in which `arrayfun` computes elementwise is extremely efficient and if such a structure could be developed for matrix operations then that would truly allow the performance of the GPU to overtake that of CPU computing.

What the work in this chapter has shown is that the structures of the GPU and the Parallel Computing Toolbox in MATLAB are such that while certain algorithms have the ability to be adjusted for improved performance not all methods do. In particular it seems clear that implicit methods with matrix and vector operations will in fact run much slower on the GPU than the CPU. Thus whether GPU computing is able to improve the performance of a numerical scheme is very much dependent upon the type of computations which need to be done. In our case we discovered that the implicit and nonlinear nature of our numerical schemes do not lend themselves towards improved performance via the implementation of the parallel processing power of a GPU.

Acknowledgements

I would like to thank Mr. Dario Fanucchi for invaluable discussions.

Author details

Charis Harley
Faculty of Science, University of the Witwatersrand, School of Computational and Applied Mathematics, Centre for Differential Equations, Continuum Mechanics and Applications, South Africa

6. References

[1] Abd El-Salam, M. R. & Shehata, M. H. (2005). The numerical solution for reaction diffusion combustion with fuel consumption, *A*ppl. Math. Comp., 160:423Ű-435.

[2] Anderson, C. A.; Zienkiewicz, O. C. (1974). Spontaneous ignition: finite element solutions for steady and transient conditions, *J*. Heat Transfer, 96(3):398–404

[3] Anderson, D.; Hamn*é*n, H.; Lisak, M.; Elevant T. & Persson, H (1991). Transition to thermonuclear burn in fusion plasmas, *P*lasma Physics and Controlled Fusion, 33(10):1145–1159

[4] Bieniasz, L. K. (1999). Finite-difference electrochemical kinetic simulations using the Rosenbrock time integration scheme, *J*ournal of Electroanalytical Chemistry, 469:97–115

[5] Bieniasz, L. K. & Britz, D. (2001). Chronopotentiometry at a Microband Electrode: Simulation Study Using a Rosenbrock Time Integration Scheme for Differential-Algebraic Equations, and a Direct Sparse Solver, *J*ournal of Electroanalytical Chemistry, 503:141–152

[6] Boddington, T. & Gray, P. (1970). Temperature profiles in endothermic and exothermic reactions and interpretation of experimental rate data, *P*roc. Roy. Soc. Lond Ser A - Mat. Phys. Sci., 320(1540):71–100

[7] Britz, D. (2005). *D*igital Simulation in Electrochemistry, 3rd Edition, Lecture Notes in Physics, Springer, 3 − 540 − 23979 − 0, Berlin Heidelberg

[8] Britz, D.; Baronas, R.; Gaidamauskaitė, E. & Ivanauskas, F. (2009). Further Comparisons of Finite Difference Schemes for Computational Modelling of Biosensors, *N*onlinear Analysis: Modelling and Control, 14(4):419–433

[9] Britz, D.; Strutwolf J. & Østerby, O. (2011). Digital simulation of thermal reactions, *A*ppl. Math. and Comp., 218(4), 15:1280–1290

[10] Chambré, P. L. (1952). On the solution of the Poisson-Boltzmann equation with application to the theory of thermal explosions, *J*. Chem. Phys., 20:1795–1797

[11] Chan, Y. N. I.; Birnbaum, I. & Lapidus, L. (1978). Solution of Stiff Differential Equations and the Use of Imbedding Techniques, *I*nd. Eng. Chem. Fundam., 17(3):133–148

[12] Crank J. & Nicolson, E. (1947). A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type, *P*roc. Camb. Phil. Soc., 43:50–67

[13] Crank J. & Furzeland, R. M. (1977). The treatment of boundary singularities in axially symmetric problems containing discs, *J*. Inst. Math. Appl., 20(3):355–370

[14] Evans. D. J. & Danaee, A. (1982). A new group Hopscotch method for the numerical solution of partial differential equations, *S*IAM J. Numer. Anal., 19(3):588–598

[15] Feldberg, S. W. (1987). Propagational inadequacy of the hopscotch finite difference algorithm: the enhancement of performance when used with an exponentially expanding grid for simulation of electrochemical diffusion problems, *J*. Electroanal. Chem., 222:101–106

[16] Frank-Kamenetskii, D. A. (1969). *D*iffusion and Heat Transfer in Chemical Kinetics, Plenum Press, New York

[17] Hairer E. & Wanner, G. (1991). Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems, Springer-Verlag, $3-540-60452-9$, Berlin

[18] Harley, C. & Momoniat, E. (2007). Steady state solutions for a thermal explosion in a cylindrical vessel, *M*odern Physics Letters B (MPLB), 21(14):831–841.

[19] Harley, C. & Momoniat, E. (2008). Instability of invariant boundary conditions of a generalized Lane-Emden equation of the second-kind, *A*pplied Mathematics and Computation, 198:621–633

[20] Harley, C. & Momoniat, E. (2008). Alternate derivation of the critical value of the Frank-Kamenetskii parameter in the cylindrical geometry, *J*ournal of Nonlinear Mathematical Physics, 15(1):69–76

[21] Harley, C. (2010). Explicit-implicit Hopscotch method: The numerical solution of the Frank-Kamenetskii partial differential equation, *J*ournal of Applied Mathematics and Computation, 217(8):4065–4075

[22] Harley, C. (2011). Crank-Nicolson and Hopscotch method: An emphasis on maintaining the implicit discretisation of the source term as a means of investigating critical parameters. Special Issue on 'Nonlinear Problems: Analytical and Computational Approach with Applications', *Abstract and Applied Analysis*, Submitted.

[23] Lang, J. (1996). High-resolution self-adaptive computations on chemical reaction-diffusion problems with internal boundaries, *C*hemical Engineering Science, 51(7):1055–1070

[24] Lang, J. (2001). *A*daptive Multilevel Solution of Nonlinear Parabolic PDE Systems, Springer, 9783540679004, Berlin

[25] The MathWorks, Inc. ©1994-2012. Parallel Computing Toolbox Perform parallel computations on multicore computers, GPUs, and computer clusters, *http://www.mathworks.com/products/parallel-computing/*.

[26] Press, W. H.; Teukolsky, S. A.; Vetterling, W. T. & Flannery, B. P. (1986). *N*umerical Recipes in Fortran, 2nd Edition, Cambridge University Press, $0-521-43064-X$, Cambridge

[27] Rice, O. K. (1940). The role of heat conduction in thermal gaseous explosions, *J*. Chem. Phys., 8(9):727–733

[28] Roche, M. (1988). Rosenbrock methods for differential algebraic equations, *N*umerische Mathematik, 52:45–63

[29] Rosenbrock, H. H. (1963). Some general implicit processes for the numerical solution of differential equations, *T*he Computer Journal, 5(4):329–330

[30] Sandu, A.; Verwer, J. G.; Blom, J.G.; Spee, E. J. & Carmichael, G. R. (1997). Benchmarking Stiff ODE Solvers for Atmospheric Chemistry Problems II: Rosenbrock Solvers, *A*tmospheric Environment, 31:3459–3472

[31] Steggerda, J. J. (1965). Thermal stability: an extension of Frank-Kamenetskii's theory, *J*. Chem. Phys., 43:4446–4448

[32] Zhang, G.; Merkin J. H. & Scott, S. K. (1991). Reaction-diffusion model for combustion with fuel consumption: Ii. Robin boundary conditions, *I*MA J. Appl. Math., 51:69–93

[33] Zhang, G.; Merkin J. H. & Scott, S. K. (1991). Reaction-diffusion model for combustion with fuel consumption: I. Dirichlet boundary conditions, *I*MA J. Appl. Math., 47:33–60

[34] Zeldovich, Y. B.; Barenblatt, G. I.; Librovich, V. B. & Makhviladze, G. M. (1985). The Mathematical Theory of Combustion and Explosions, *C*onsultants Bureau, New York

Chapter 6

Fuzzy Analytical Network Process Implementation with Matlab

Xiaoguang Zhou

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46466

1. Introduction

In many complex decision-making problems, the decision information provided by the decision makers is often imprecise or uncertain due to time limit, lack of data, or the decision makers' limited attention and information processing capabilities. Decision-making in a fuzzy environment means a decision process in which the goals and/or the constraints, but not necessarily the system under control, are fuzzy in nature. This means that the goals and/or the constraints constitute classes of alternatives whose boundaries are not sharply defined. Fuzzy set, whose basic component is a membership function (Zadeh, 1965), was introduced in the following several decades. Fuzzy set theory has been applied successfully in the decision-making field.

Matlab is a suitable tool for solving fuzzy decision-making problems. This chapter is focusing on how to solve a specific class of fuzzy decision-making problem, that is, Fuzzy Analytical Network Process (FANP) by Matlab. Project selection is chosen as an example to illustrate the proposed method. The reason is that project selection is a complex decision-making process. It involves a search from the environment of opportunities, the generation of project options, and the evaluation of multiple attributes, both qualitative and quantitative, by different stakeholders.

There are various mathematic techniques for selecting an optimal project. Mathematical programming models can be used to accomplish this decision. For example, the R&D project selection can be presented based on linear, non-linear, dynamic, goal, and stochastic mathematical programming (Wang & Hwang, 2007). Based on the idea of moments approximation method via linear programming, Fang et al. (2008) proposed a scenario generation approach for the mixed single-stage Research and Development (R&D) projects and multi-stage securities portfolio selection problem, etc.

Also, project selection has been presented in regard with multiple objectives. For instance, Gabriel et al. (2006) developed a multiobjective, integer-constrained optimization model with competing objectives for project selection, and the subjective rank is determined via the Analytic Hierarchy Process (AHP). Ghorbani & Rabbani (2009) proposed a multi-objective algorithm for project selection. Two objective functions have been considered to maximize total expected benefit of selected projects and minimize the summation of the absolute variation of allotted resource between each successive time periods. Gutjahr et al. (2010) presented a multi-objective optimization model for project portfolio selection taking employee competencies and their evolution into account, and so on.

Fuzzy sets theory is utilized to cover the vagueness inherent in the nature of project selection problem as well. For example, Huang et al. (2008) presented a fuzzy analytic hierarchy process method and utilize crisp judgment matrix to evaluate subjective expert judgments. Bhattacharyya et al. (2011) developed a fuzzy multi-objective programming approach to facilitate decision making in the selection of R&D projects. Ebrahimnejad et al. (2011) considered a two-phase decision making approach, combining a modified version of the Analytic Network Process (ANP) method and an improved compromise ranking method under uncertainty. Chang & Lee (2012) proposed a Data Envelopment Analysis (DEA), knapsack formulation and fuzzy set theory integrated model to deal with the project selection , etc.

Some researchers tried the project selection problem in other ways. Dey (2006) proposed a decision support system, which analyses projects with respect to market, technicalities, and social and environmental impact in an integrated framework using analytic hierarchy process. Liesiö et al. (2007) developed the Robust Portfolio Modeling methodology which extends Preference Programming methods into portfolio problems, where a subset of project proposals are funded in view of multiple evaluation criteria. Smith-Perera et al. (2010) proposed an approach to prioritize project portfolio in an efficient and reliable way based on analytic network process method and the information obtained from the experts during the decision-making process. Shakhsi-Niaei et al. (2011) presented a procedure which used the PROMETHEE method linked to a Monte Carlo simulation in order to consider and possibly make lower all kinds of uncertainties of project selection problem in an acceptable complexity level, and so on.

As mentioned above, AHP or ANP was widely used during the process of selecting or evaluating an optimal project (Gabriel et al., 2006; Dey, 2006; Smith-Perera et al., 2010; Ebrahimnejad et al., 2011), and these literatures (Cheng & Li, 2005; Amiri, 2010; Aragonés-Beltrán et al., 2010; Jung & Seo, 2010; etc) are also about project selection based on AHP/ANP. In addition, AHP/ANP was proverbially used in other decision-making fields as well (Kahraman et al., 2006; Lee et al., 2009; Arunraj & Maiti, 2010; Huang et al., 2011; etc.). The main reason is that AHP/ANP can deal with qualitative and quantitative information at the same time, and ANP can take into account the interaction and feedback relationships between criteria and/or indices. However, due to the vagueness and uncertainty on the judgments of decision-makers, the crisp pairwise comparison in the conventional AHP/ANP seems insufficient and imprecise to capture the right judgments of decision-makers.

In this study, a fuzzy logic is introduced for the pairwise comparison of ANP to make up the deficiency of the conventional AHP/ANP, referred to as FANP. The objective of this chapter is to present a FANP-based approach for the construction project selection problem using triangular fuzzy numbers. According to the fuzzy preference programming (FPP) method, local weights of fuzzy pairwise comparison matrices can be achieved. Then an unweighted and weighted supermatrix based on its network structure can be formed. For FANP, the key steps are to calculate the local weights and the limit supermatrix. Both of them can be solved by Matlab. A case will be given by the proposed method, and Matlab codes will be provided as well.

2. What's FANP?

The ANP, introduced by Saaty, is a generalization of the AHP (Saaty, 1996). ANP is the first mathematical theory that makes it possible to deal with all kinds of dependences and feedbacks by replacing hierarchies with networks (Saaty, 1996). ANP allows for complex inter-relationships among decision dimensions and attributes. The ANP feedback approach replaces hierarchies with networks in which the relationships between dimensions are not easily represented as higher or lower, dominant or subordinate, direct or indirect. For instance, not only does the importance of the criteria determine the importance of the attributes, as in a hierarchy, but also the importance of the attributes may have impact on the importance of the criteria. A hierarchical structure with a linear top-to-bottom form is not suitable for a complex system.

As we know, AHP/ANP has been proposed as a suitable multi-criteria decision analysis tool for project selection and evaluation. However, the conventional AHP/ANP-based decision model seems to be ineffective in dealing with the inherent fuzziness or uncertainty in judgment during the pairwise comparison process. Although the use of the discrete scale of 1-9 to represent the verbal judgment in pairwise comparisons has the advantage of simplicity, it does not take into account the uncertainty associated with the mapping of one's perception or judgment to a number. In real-life decision-making situations, the decision makers or stakeholders could be uncertain about their own level of preference, due to incomplete information or knowledge, complexity and uncertainty within the decision environment. Such situations will occur when selecting and evaluating an optimal project. Therefore, it's better to make project selection and assessment under fuzzy conditions. This chapter will focus on FANP in fuzzy decision-making based on triangular fuzzy numbers. Actually, some researchers have focused on decision-making based on FANP (Promentilla et al., 2008 ; Ayağ & Özdemir, 2009 ; Boran & Goztepe, 2010; Dağdeviren & Yüksel, 2010; Pires et al., 2011 ; Ju et al., 2012 ; etc.).

2.1. Triangular fuzzy number

A fuzzy set is a class of objects with a continuum of grades of membership. Such a set is characterized by a membership function, which assigns to each object a grade of membership ranging between zero and one.

A triangular fuzzy number (TFN) M is shown in Fig. 1. A TFN is denoted simply as (l, m, u). The parameters l, m and u, respectively, denote the smallest possible value, the most promising value, and the largest possible value that describe a fuzzy event. Each TFN has linear representations on its left and right side such that its membership function can be defined as

$$u_M(x) = \begin{cases} (x-l)/(m-l) & l \le x \le m \\ (u-x)/(u-m) & m \le x \le u \\ 0 & otherwise \end{cases} \tag{1}$$

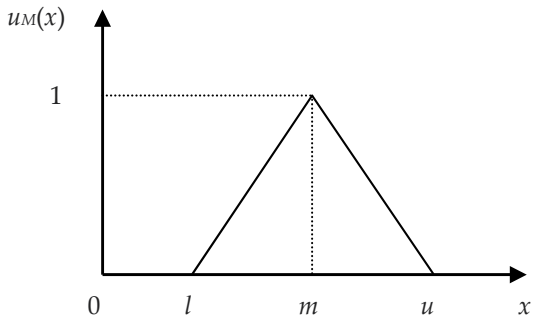


Figure 1. A triangular fuzzy number M

2.2. Fuzzy preference programming method

A number of methods have been developed to handle fuzzy comparison matrices. For instance, Van Laarhoven & Pedrycz (1983) suggested a fuzzy logarithmic least squares method to obtain the fuzzy weights from a triangular fuzzy comparison matrix. Buckley (1985) utilized the geometric mean method to calculate fuzzy weights. Chang (1996) proposed an extent analysis method, which derives crisp weights for fuzzy comparison matrices. Xu (2000) brought forward a fuzzy least squares priority method (LSM). Csutora & Buckley (2001) came up with a Lambda-Max method, which is the direct fuzzification of the well-known $k$max method. Mikhailov (2003, 2004) developed a fuzzy preference programming method, which also derives crisp weights from fuzzy comparison matrices. Srdjevic (2005) proposed a multicriteria approach for combining prioritization methods within the AHP, including additive normalization, eigenvector, weighted least-squares, logarithmic least-squares, logarithmic goal programming and fuzzy preference programming. Wang et al. (2006) presented a modified fuzzy logarithmic least square method. Yu & Cheng (2007) developed a multiple objective programming approach for the ANP to obtain all local priorities for crisp or interval judgments at one time, even in an inconsistent situation. Huo et al. (2011) proposed new parametric prioritization methods (PPMs) to determine a family of priority vectors in AHP.

FPP method, as a reasonable and effective means, is adopted in this study. This method can acquire the consistency ratios of fuzzy pairwise comparison matrices without additional

study, and the local weights can be easily solved by Matlab software. The stages of Mikhailov's fuzzy prioritization approach are given as follows (Mikhailov, 2003).

Consider a prioritization problem with n elements, where the pairwise comparison judgements are represented by normal fuzzy sets or fuzzy numbers. Supposing that the decision-maker can provide a set $F=\{\tilde{a}_{ij}\}$ of $m \le n(n-1)/2$ fuzzy comparison judgements, $i=1, 2, \ldots, n-1$; $j=2, 3, \ldots, n$; $j>i$, represented as triangular fuzzy numbers $\tilde{a}_{ij}=(l_{ij}, m_{ij}, u_{ij})$. The problem is to derive a crisp priority vector $w=(w_1, w_2, \ldots, w_n)^T$, such that the priority ratios w_i/w_j are approximately within the scopes of the initial fuzzy judgments, or

$$l_{ij} \tilde{\le} \frac{w_i}{w_j} \tilde{\le} u_{ij} \tag{2}$$

where the symbol $\tilde{\le}$ denotes the statement "fuzzy less or equal to".

Each crisp priority vector w satisfies the double-side inequality (2) with some degree, which can be measured by a membership function, linear with respect to the unknown ratio w_i/w_j

$$u_{ij}\left(\frac{w_i}{w_j}\right) = \begin{cases} \dfrac{(w_i/w_j) - l_{ij}}{m_{ij} - l_{ij}}, & \dfrac{w_i}{w_j} \le m_{ij}, \\ \dfrac{u_{ij} - (w_i/w_j)}{u_{ij} - m_{ij}}, & \dfrac{w_i}{w_j} \ge m_{ij}. \end{cases} \tag{3}$$

The membership function (3) is linearly increasing over the interval $(-\infty, m_{ij})$ and linearly decreasing over the interval (m_{ij}, ∞). The function takes negative values when $w_i/w_j<l_{ij}$ or $w_i/w_j>u_{ij}$, and has a maximum value $u_{ij}=1$ at $w_i/w_j=m_{ij}$. Over the range (l_{ij}, u_{ij}), the membership function (3) coincides with the fuzzy triangular judgment (l_{ij}, m_{ij}, u_{ij}).

The solution to the prioritization problem by the FPP method is based on two main assumptions. The first one requires the existence of non-empty fuzzy feasible area P on the $(n-1)$ dimensional simplex Q^{n-1}

$$Q^{n-1} = \{(w_1, w_2, \ldots w_n) \mid w_i > 0, \sum_{i=1}^{n} w_i = 1\}, \tag{4}$$

defined as an intersection of the membership functions, similar to (3) and the simplex hyperplane (4). The membership function of the fuzzy feasible area is given by

$$u_P(w) = \min_{ij} \left\{ u_{ij}(w) \middle| i = 1, 2, \ldots, n-1; j = 2, 3, \ldots, n; j > i \right\}. \tag{5}$$

If the fuzzy judgements are very inconsistent, then $u_P(w)$ could take negative values for all normalized priority vectors $w \in Q^{n-1}$.

The second assumption of the FPP method specifies a selection rule, which determines a priority vector, having the highest degree of membership in the aggregated membership

function (5). It can easily be proved that $u_P(w)$ is a convex set, so there is always a priority vector $w^* \in Q^{n-1}$ that has a maximum degree of membership

$$\lambda^* = u_P(w^*) = \max_{w \in Q^{n-1}} \min_{ij} \left\{ u_{ij}(w) \right\}. \tag{6}$$

The maximum prioritization problem (6) can be represented in the following way:

$$\begin{gathered} \text{Max}\, \lambda \\ \lambda \le u_{ij}(w), i = 1,2,\ldots,n-1; j = 2,3,\ldots,n; j > i, \\ \sum_{k=1}^{n} w_k = 1, w_k > 0, k = 1,2,\ldots,n. \end{gathered} \tag{7}$$

Taking the specific form of the membership functions (3) into consideration, the problem (7) can be further transformed into a bilinear program of the type

$$\begin{gathered} \text{Max}\, \lambda \\ (m_{ij} - l_{ij})\lambda w_j - w_i + l_{ij} w_j \le 0, \\ (u_{ij} - m_{ij})\lambda w_j + w_i - u_{ij} w_j \le 0, \\ \sum_{k=1}^{n} w_k = 1, w_k > 0, k = 1,2,\ldots,n. \\ i = 1,2,\ldots,n-1; j = 2,3,\ldots,n; j > i. \end{gathered} \tag{8}$$

The optimal solution to the non-linear problem above (w^*, λ^*) might be obtained by employing some appropriate numerical method for non-linear optimization. The optimal value λ^*, if it is positive (the maximum value is one), indicates that all solution ratios satisfy the fuzzy judgment completely, which means that the initial set of fuzzy judgments is rather consistent. A negative value of λ^* shows that the solutions ratios approximately satisfy all double-side inequalities (2). Therefore, the optimal value λ^* can be used for measuring the consistency of the initial set of fuzzy judgments.

3. Proposed project selection and evaluation framework

This study proposes an analytic approach based on the fuzzy ANP to assist in project selection and evaluation. We first identify the selection and evaluation criteria. Then we present the evaluation model in the following subsections.

3.1. Criteria of project selection

Some researchers investigated which factors have an effect on project selection. For example, Mohanty (1992) pointed out that a potential project should have four important features: minimum investment, low complex of technology, short period and high returns. Lin & Chen (2004) suggested that the contractors should consider the essence of bidding, competition, the value of tendering opportunities, resources, and corporate reputation.

Kumar (2006) pointed out that project selection belongs to the strategic level for any organizations, and today it is seriously influenced by environmental and social factors. Wang et al. (2009) suggested that during project selection, the evaluation index system can be divided into two parts: the bid/non-bid and which project to bid, then make a comprehensive evaluation, etc.

With the interaction and feedback relationships between dimensions and/or attribute-enablers being considered, a four-level evaluation index system is presented, as shown in Fig. 2. The first level is the objective, which is to find out an optimal project; the second level is the dimensions including project profitability, project risk, project owners and project bidding competition; the third level is attribute-enablers, 15 indicators included; and the lowest level is the alternatives.

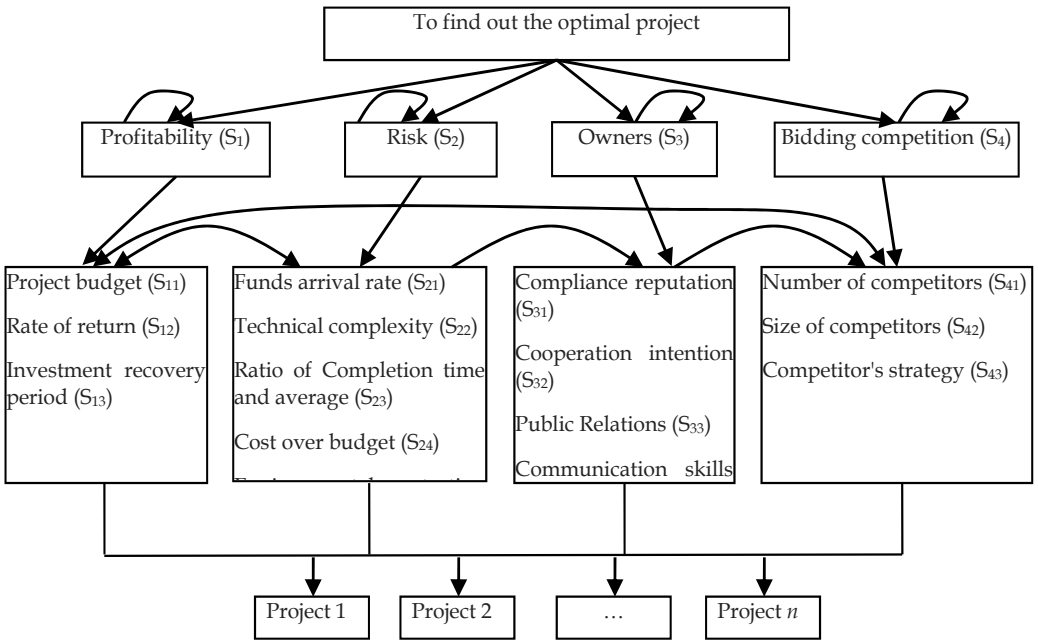


Figure 2. The ANP structure for construction project selection

Generally, if project profitability (S_1) has an effect on project bidding competition (S_4), then a line with arrow from S_4 to S_1 is added. If the sub-criteria of project risk (S_2) has interaction with itself, then S_2 is inner dependent, and an arc with arrow is added to S_2.

3.2. FANP-based approach

The fuzzy ANP-based approach is presented step-by-step as follows:

Step 1. Model construction and problem structuring. With the relationships among dimensions and attribute-enablers being considered, a four levels selection and evaluation index system is proposed, as shown in Fig. 2.

During the process of project selection and evaluation, experts tend to specify their preferences in the form of natural language expressions. The fuzzy linguistic variable, whose value represents the range from natural to artificial language, is a variable that reflects different aspects of human language. When the values or meanings of a linguistic factor are being reflected, the resulting variable must also reflect appropriate modes of change for that linguistic factor. Moreover, variables describing a human word or sentence can be divided into numerous linguistic criteria, such as equally important, moderately important, important, very important or absolutely important. For the purposes of the present study, a 9-point scale is presented for the relative importance of pairwise comparison, as shown in Table 1.

Step 2. Establishing pairwise comparison matrices by decision committee using the linguistic scales for relative importance given in Table 1. For example, complication of technique (S_{11}) and maturity of technique (S_{12}) are compared using the question "How important is the complication of technique when it is compared with the maturity of technique at the dimension of technique risk?" and the answer is "moderately important (MI)", so this linguistic scale is placed in the relevant cell against the triangular fuzzy numbers (2, 3, 4). All the fuzzy evaluation matrices are produced in the same manner.

Linguistic scale for importance	Triangular fuzzy scale	Triangular fuzzy reciprocal scale
Equally important(EI)	(1, 1, 1)	(1, 1, 1)
Intermediate1(IM_1)	(1, 2, 3)	(1/3, 1/2, 1)
Moderately important(MI)	(2, 3, 4)	(1/4, 1/3, 1/2)
Intermediate2(IM_2)	(3, 4, 5)	(1/5, 1/4, 1/3)
Important(I)	(4, 5, 6)	(1/6, 1/5, 1/4)
Intermediate3(IM_3)	(5, 6, 7)	(1/7, 1/6, 1/5)
Very important(VI)	(6, 7, 8)	(1/8, 1/7, 1/6)
Intermediate4(IM_4)	(7, 8, 9)	(1/9, 1/8, 1/7)
Absolutely important(AI)	(9, 9, 9)	(1/9, 1/9, 1/9)

Table 1. Linguistic scales for relative importance

Step 3. Calculating the local weights of the factors and sub-factors taking part in the second and third levels of the ANP model by FPP method according to formulation (8).

Step 4. Constructing an unweighted supermatrix based on the interdependencies in the network. The supermatrix is a partitioned matrix, in which each submatrix is composed of a set of relationships between dimensions and attribute-enablers in the graphical model.

Step 5. Acquiring the weighted supermatrix. Because in each column it consists of several eigenvectors which of them sums to one (in a column of a stochastic), and hence the entire column of the matrix may sum to an integer greater than one, the unweighted supermatrix needs to be stochastic to derive the weighted supermatrix.

Step 6. Obtaining the limit supermatrix. The weighted supermatrix will not be in a steady state until the row values of which converge to the same value for each column of the matrix, then the limit supermatrix is achieved.

$$\overline{W} = \lim_{t \to \infty} W^t. \tag{9}$$

Step 7. Calculating the comprehensive weights. A comprehensive weight of each index can be obtained by multiplying the weight of the criterion level indicator, the weight of independent sub-criterion and the weight of interdependent sub-index.

$$w_{ij} = P_i \times A_{ij}^D \times A_{ij}^I, \tag{10}$$

where w_{ij} is the comprehensive weight of each factor, P_i is relative importance weight of dimension i on final goal; A_{ij}^D, relative importance weight for attribute-enabler j of dimension i, and for the dependency (D) relationships within attribute-enabler's component level; A_{ij}^I, stabilized relative importance weight for attribute-enabler j of dimension i, and for the independency (I) relationships within attribute-enabler's component level; i=1, 2, …, m; j=1, 2, …, n.

Step 8. Selecting an optimal construction project. The equation of desirability index, D_k for alternative k is calculated using the following equation:

$$D_k = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} S_{kij} \tag{11}$$

where w_{ij} is the comprehensive weight; S_{ikj}, relative impact of construction project alternative k on attribute-enabler j of dimension i of selection network.

4. How to use Matlab during the process of decision-making with FANP?

Two key steps in the process of FANP will be solved by Matlab. One is acquiring local weights of fuzzy pairwise comparison matrices (step 3); the other is obtaining the limit supermatrix (step 6). Both of them are associated with matrix calculation. Matlab is selected for its excellent performance on matrix operation and data processing.

4.1. Acquiring local weights of fuzzy pairwise comparison matrices

$$\begin{aligned}
&\text{Max}\,\lambda \\
&(m_{ij} - l_{ij})\lambda w_j - w_i + l_{ij} w_j \le 0, \\
&(u_{ij} - m_{ij})\lambda w_j + w_i - u_{ij} w_j \le 0, \\
&\sum_{k=1}^{n} w_k = 1, w_k > 0, k = 1, 2, \ldots, n. \\
&i = 1, 2, \ldots, n-1; j = 2, 3, \ldots, n; j > i.
\end{aligned}$$

As mentioned before, the local weights of fuzzy pairwise comparison matrices are acheived by FPP method. That is, the local weights will be obtained by solving the non-linear problem above.

As criteria and sub-criteria have different numbers, there are different orders of fuzzy pairwise comparison matrices, such as matices of order (2×2), (3×3), …, (n×n). Therefore, the local weights and consistency index may be derived from matrix of different orders. Function definition and non-linear program calculation will be first concerned in this section, then some examples will be given to illustrate the proposed method.

4.1.1. Function definition

To acquire the local weights of fuzzy comparison matrices, some functions need to be defined. A procedure can be saved as the format - ".m" file in Matlab. The name of which can be used directly when you need to call it.

To obtain the local weights, a main program file is developed, named as "networkmain.m". The local weights can be easily acquired by inputting "networkmain" in the command window. As there are different forms of comparison matrices, we need to change slightly the main program file for matrices of different orders.

Matlab contains a lot of functions to solve linear and non-linear problems. For instance, non-linear problems can be solved by function "fmincon", which is also the key function of the file "networkmain.m". The full expression of the function is "fmincon (fun, $x0$, A, b, Aeq, beq, VLB, VUB, nonlcon)". During the process of decision-making with FANP, in accordance with the FPP method, the fuzzy pairwise comparison matrices first need to be transformed into non-linear programming formats. Then it can be solved by the function "fmincon". The detail description of the function is as follows:

"Fun" is the objective function of a non-linear problem. A variable in the objective function is marked as $x(i)$. If the total number of variables is n, then they are correspondingly named as $x(1)$, $x(2)$, …, $x(n)$. There are (n+1) variables for a (n×n) comparison matrix, including n local weights and a consistency index λ. The objective function in FPP method is to acquire the maximum value, but the default standard objective function of "fmincon" in Matlab is to find the minimum value, so it is necessary to convert $x(n+1)$ into $-x(n+1)$ in the function "fmincon". Of course, it might be solved by changing the constraints instead of changing the objective function as well.

"$x0$" is the initial value of the non-linear problem, and it has the same scale as the number of variables. Every local weight $x(i)$ takes values in the range [0, 1], and their sum satisfies $x(1)+x(2)+\ldots+x(n)=1$. Consistency index $x(n+1)$ takes values in the range (-∞, 1].

"A and b" are the coefficients of linear inequality constraint $Ax<=b$. As there are no linear inequality constraints in FANP, a and b can be ignored or replaced with two empty arrays.

"Aeq and beq" are both the coefficients of linear equality constraint Aeq*x=beq. According to FANP, the sum of local weights should be one, then

$$x(1) + x(2) + \ldots + x(n) + 0x(n+1) = 1,$$

where $x(1)$, $x(2)$, …, $x(n)$ are the first, second, …, nth local weight respectively, and $x(n+1)$ is the consistency index. Then we have Aeq=[1 1 … 1 0] and beq=[1].

"VLB and VUB" are the upper and lower bounds of the variables. According to the FPP method and FANP, all the local weights have a lower bound of zero, and the lower bound of consistency index is negative infinity. Since all the upper bounds are subject to the constraints, they can be replaced with empty arrays. In matlab, the postive and negative infinity symbols are named as inf and –inf respectively.

"nonlcon" is the non-linear constraints, including non-linear inequality constaint c and non-linear equality constraint ceq. As there is no non-linear equality constaint for FPP method, we can let ceq=[ ].

To solve the non-linear problem (8), the following main program file "networkmain.m" is developed.

```
Aeq=[1 1 … 1 0];
beq=[1];
VLB = [0; 0;…; 0; -inf];
VUB = [ ];
x0 = [0.1; 0.2; …; 1];
OPT = optimset('LargeScale', 'off');
[x, fval] = fmincon('networkf', x0, [ ], [ ],Aeq, beq, VLB, VUB, 'networknonlcon', OPT)
```

The LargeScale option specifies a preference for which algorithm to use. It is only a preference because certain conditions must be met to use the large-scale algorithm. For this function "fmincon", we choose the medium-scale algorithm. LargeScale use the medium-scale algorithm when set to 'off'. Two files, "networkf.m" and "networknonlcon.m", are called in the procedure above. They are defined for objective function and non-linear constraints independently.

The program of "networkf.m" is developed as follows:

```
function f =networkf(x);
f = -x(n+1);
```

where the value of function f is related to n. For example, for a matrix of order 3, n=3, f=-x(4); for a matrix of order 4, n=4, f=-x(5). Therefore, objective function f has different formats as a result of different sizes of matrices.

According to the FPP method, every triangular fuzzy number (l_{ij}, m_{ij}, u_{ij}) needs to be transformed into the following inequality constraints.

$(m_{ij} - l_{ij})*x(n+1)*x(j) - x(i) + (l_{ij})*x(j) \leq 0;$
$(u_{ij} - m_{ij})*x(n+1)*x(j) + x(i) - (u_{ij})*x(j) \leq 0.$

As we know, a triangular fuzzy comparison matrix is symmetric. Therefore, for the following matrix, we only need to consider the constraints above the diagonal.

$$\begin{pmatrix} (l_{11},m_{11},u_{11}) & (l_{12},m_{12},u_{12}) & \dots & (l_{1n},m_{1n},u_{1n}) \\ (l_{21},m_{21},u_{21}) & (l_{22},m_{22},u_{22}) & \dots & (l_{2n},m_{2n},u_{2n}) \\ \vdots & \vdots & \vdots & \vdots \\ (l_{n1},m_{n1},u_{n1}) & (l_{n2},m_{n2},u_{n2}) & \dots & (l_{nn},m_{nn},u_{nn}) \end{pmatrix}$$

For instance, for a 3-by-3 fuzzy comparison matrix, only three elments need to be taken into account, which is, (l_{12}, m_{12}, u_{12}), (l_{13}, m_{13}, u_{13}) and (l_{23}, m_{23}, u_{23}). Then, the corresponding "networknonlcon.m" file is given by

```
function [c, ceq] = networknonlcon3(x);

c = [
    (m12-l12)*x(4)*x(2)-x(1)+(l12)*x(2);
    (u12-m12)*x(4)*x(2)+x(1)-(u12)*x(2);
    (m13-l13)*x(4)*x(3)-x(1)+(l13)*x(3);
    (u13-m13)*x(4)*x(3)+x(1)-(u13)*x(3);
    (m23-l23)*x(4)*x(3)-x(2)+(l23)*x(3);
    (u23-m23)*x(4)*x(3)+x(2)-(u23)*x(3);
    ];
ceq = [ ];
```

The programs for (4×4), (5×5), …, (n×n) matrices can be developed in the same manner. Several examples for acquiring the local weights are given as follows.

4.1.2. Local weights of a fuzzy pairwise comparison matrix of order 3

Example 1. How to solve the local weights of the following fuzzy pairwise comparison matrix of order 3?

$$\begin{pmatrix} (1,1,1) & (1/6,1/5,1/4) & (1/4,1/3,1/2) \\ (4,5,6) & (1,1,1) & (2,3,4) \\ (2,3,4) & (1/4,1/3,1/2) & (1,1,1) \end{pmatrix}$$

First, the initial parameter of the main function "networkmain" needs to be set for this (3×3) matrix, which has three local weights, named as $x(1)$, $x(2)$ and $x(3)$, and a consistency index, referred to as $x(4)$. That means there are four variables in linear equality constraint. The corresponding "networkmain.m" file is as follows:

```
Aeq=[1 1 1 0];
beq=[1];
VLB = [0; 0; 0; -inf];
VUB = [ ];
x0 = [0.2; 0.5; 0.3; 1];
OPT = optimset('LargeScale', 'off');
[x, fval] = fmincon('networkf', x0, [ ], [ ],Aeq, beq, VLB, VUB, 'networknonlcon3', OPT)
```

As it mentioned before, the opposite number of consistency index in the function "network" is taken. The corresponding objective function file "networkf.m" is as follows:

```
function f = networkf(x);
f = -x(4);
```

Since the triangular fuzzy comparison matrix is symmetric, only three constraints above the diagonal need to be considered, that is (1/6, 1/5, 1/4), (1/4, 1/3, 1/2) and (2, 3, 4). The "networknonlcon3.m" file for this matrix is as follows:

```
function [c,ceq] = networknonlcon3(x);

c = [
    (1/5-1/6)*x(4)*x(2)-x(1)+(1/6)*x(2);
    (1/4-1/5)*x(4)*x(2)+x(1)-(1/4)*x(2);
    (1/3-1/4)*x(4)*x(3)-x(1)+(1/4)*x(3);
    (1/2-1/3)*x(4)*x(3)+x(1)-(1/2)*x(3);
    (3-2)*x(4)*x(3)-x(2)+(2)*x(3);
    (4-3)*x(4)*x(3)+x(2)-(4)*x(3);
    ];
ceq = [ ];
```

```
>> networkmain
Optimization terminated: first-order optimality measure less
 than options.TolFun and maximum constraint violation is less
 than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
                                       1
                                       4
                                       5

x =

    0.1128
    0.6265
    0.2607
    0.4031


fval =

   -0.4031
```

Figure 3. The local weights and consistency index of Example 1

In Fig. 3, x is the optimal solution and fval is the optimal value. The first three values of x are the local weights of the matrix, and the last one is the consistency index.

The final step, is to run "networkmain" in the command window to acquire the local weights. The local weights $x(1)$, $x(2)$, $x(3)$ are 0.1128, 0.6265, 0.2607 respectively, as shown in Figure 3. The consistency index $x(4)$ is 0.4031>0, which means the fuzzy comparison matrix has a good consistency, and the results are acceptable.

4.1.3. Local weights of a fuzzy pairwise comparison matrix of order 4

Example 2. How to solve the local weights of the following fuzzy pairwise comparison matrix of order 4?

$$\begin{pmatrix} (1,1,1) & (3/2,2,5/2) & (1,3/2,2) & (5/2,3,7/2) \\ (2/5,1/2,2/3) & (1,1,1) & (2/3,1,2) & (3/2,2,5/2) \\ (1/2,2/3,1) & (1/2,1,3/2) & (1,1,1) & (2,5/2,3) \\ (2/7,1/3,2/5) & (2/5,1/2,2/3) & (1/3,2/5,1/2) & (1,1,1) \end{pmatrix}$$

The calculation process for a matrix of order 4 is similar to that of a matrix of order 3. A matrix of order 4 has four local weights, named as $x(1)$, $x(2)$, $x(3)$ and $x(4)$, and a consistency index, referred as $x(5)$. The following program is the corresponding "networkmain.m" file for this matrix.

```
Aeq=[1 1 1 1 0];
beq=[1];
VLB = [0; 0; 0; 0; -inf];
VUB = [ ];
x0 = [0.1; 0.4; 0.2; 0.3; -1];
OPT = optimset('LargeScale', 'off');
[x, fval] = fmincon('networkf', x0, [ ], [ ],Aeq, beq, VLB, VUB, 'networknonlcon4', OPT)
```

The corresponding objective function file for this matrix is as follows:

```
function f = networkf(x);
f = -x(5);
```

Six constraints for this fuzzy pairwise comparison matrix need to be calculated, that is, (3/2, 2, 5/2), (1, 3/2, 2), (5/2, 3, 7/2), (2/3, 1, 2), (3/2, 2, 5/2) and (2, 5/2, 3). The "networknonlcon4.m" file for this matrix of order 4 is as follows:

```
function [c, ceq] = networknonlcon4(x);
c = [
    (2-3/2)*x(5)*x(2) - x(1) + (3/2)*x(2);
    (5/2 - 2)*x(5)*x(2) + x(1) - (5/2)*x(2);
    (3/2-1)*x(5)*x(3) - x(1) + (1)*x(3);
```

```
(2 - 3/2)*x(5)*x(3) + x(1) - (2)*x(3);
(3-5/2)*x(5)*x(4) - x(1) + (5/2)*x(4);
(7/2-3)*x(5)*x(4) + x(1) - (7/2)*x(4);
(1-2/3)*x(5)*x(3) - x(2) + (2/3)*x(3);
(2-1)*x(5)*x(3) + x(2) - (2)*x(3);
(2-3/2)*x(5)*x(4) - x(2) + (3/2)*x(4);
(5/2-2)*x(5)*x(4) + x(2) - (5/2)*x(4);
(5/2 - 2)*x(5)*x(4) - x(3) + (2)*x(4);
(3 - 5/2 )*x(5)*x(4) + x(3) - (3)*x(4);
];
ceq = [ ];
```

Finally, "networkmain" is run in the command panel to obtain the local weights. The optimal solutions are $x(1)$=0.3891, $x(2)$=0.2229, $x(3)$=0.2685, $x(4)$=0.1196, $x(5)$=0.4910, as shown in Figure 4. The consistency index $x(5)$ is 0.4910>0, which means the fuzzy comparison matrix has a good consistency, and the results are acceptable.

```
>> networkmain
Optimization terminated: first-order optimality measure less than options.TolFun
 and maximum constraint violation is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
                                     1
                                     6
                                     7
                                    11

x =

    0.3891
    0.2229
    0.2685
    0.1196
    0.4910


fval =

   -0.4910
```

Figure 4. The local weights and consistency index of Example 2

where x is the optimal solution and fval is the optimal value. The first four values of x are the local weights of the matrix, and the last one is the consistency index.

4.1.4. Local weights of a fuzzy pairwise comparison matrix of order n

Example 3. How to solve the local weights of the following fuzzy pairwise comparison matrix of order n?

$$\begin{pmatrix} (l_{11},m_{11},u_{11}) & (l_{12},m_{12},u_{12}) & \dots & (l_{1n},m_{1n},u_{1n}) \\ (l_{21},m_{21},u_{21}) & (l_{22},m_{22},u_{22}) & \dots & (l_{2n},m_{2n},u_{2n}) \\ & & \vdots & \\ (l_{n1},m_{n1},u_{n1}) & (l_{n2},m_{n2},u_{n2}) & \dots & (l_{nn},m_{nn},u_{nn}) \end{pmatrix}$$

A (n×n) matrix has (n+1) variables, including n local weights and one consistency index, named as $x(1)$, $x(2)$, …, $x(n+1)$. Linear equality constraint is as follows:

$$x(1) + x(2) + \dots + x(n) + 0*x(n+1) = 1$$

Then, Aeq = [1 1 … 1 0], beq = [1].

The lower bounds of the local weights are zero, and the lower bound of consistency index is negative infinity. There is no specific upper bound. Then we have

VLB = [0; 0;…; 0; -inf]; VUB = [ ].

The initial values of the variables can be arbitrary in the range of feasible region. If different initial values lead to different results, it means the nonlinear problem has multiple optimal solutions. Then, we have

```
x0 = [1; 1; …; 1],
[x, fval] = fmincon('networkf', x0, [ ], [ ],Aeq, beq, VLB, VUB, 'networknonlcon').
```

The corresponding objective function file networkf.m is as follows:

```
function f = networkf(x);
f = -x(n+1);
```

For the non-linear constraints, only those elements above the diagonal need to be considered. That is, these triangular fuzzy numbers (l_{ij}, m_{ij}, u_{ij}) need to be taken into account, $i<j$; i=1, 2, …, n; j= 2, 3,…, n.

The "networknonlcon$n$.m" file in this case is as follows:

```
function [c,ceq] = networknonlconn(x);
c = [
    (m12-l12)*x(n+1)*x(2)-x(1)+(l12)*x(2);
    (u12-m12)*x(n+1)*x(2)+x(1)-(u12)*x(2);
    (m13-l13)*x(n+1)*x(3)-x(1)+(l13)*x(3);
    (u13-m13)*x(n+1)*x(3)+x(1)-(u13)*x(3);
        …
    (m1n-l1n)*x(n+1)*x(n)-x(1)+(l1n)*x(n);
```

$(u_{1n}-m_{1n})*x(n+1)*x(n)+x(1)-(u_{1n})*x(n);$
$(m_{23}-l_{23})*x(n+1)*x(3)-x(2)+(l_{23})*x(3);$
$(u_{23}-m_{23})*x(n+1)*x(3)+x(2)-(u_{23})*x(3);$
…
$(m_{(n-1)n}-l_{(n-1)n})*x(n+1)*x(n)-x(n-1)+(l_{(n-1)n})*x(n);$
$(u_{(n-1)n}-m_{(n-1)n})*x(n+1)*x(n)+x(n-1)-(u_{(n-1)n})*x(n);$
];
ceq = [ ];

Finally, we run "networkmain" in the command panel to obtain the local weights. If the consistency index is positive, the fuzzy comparison matrix has a good consistency and the results are acceptable. Otherwise, we need to modify the fuzzy pairwise comparison matrix until it is satisfied with consistency requirement.

4.2. Obtaining limit supermatrix

A limit supermatrix is a weighted supermatrix in a stable state. The weighted supermatrix may be convergent or not. If it is convergent, the limit supermatrix can be achieved. Unfortunately, it is usually not convergent, and then a periodic result is obtained. Under this condition, the limit supermatrix will be achieved only after the periodicity of the supermatrix is determined.

4.2.1. The program for acquiring stable limit supermatrix

To obtain a limit supermatrix, four functions named as "fanp_limitedsupermatrix.m", "fanp_multiMatrix.m", "fanp_circulantCheck.m" and "fanp_equal.m" are developed. The first file is the main program by which limit supermatrix can be solved; supermatrix multiplication is implemented by the second file; the third file is used to determine whether a supermatrix is stable or periodic; and the last one is used to test whether a supermatrix after iterations is the same as it was before or not. The second and third files will be called in the main procedure, and the last one will be used in the third program. File "fanp_limitedsupermatrix.m" is given as follows:

```
function B = fanp_limitedsupermatrix( );
weightedsupermatrix = [0.00000,0.00000,0.00000,0.63400,0.25000,0.40000;
        0.00000,0.00000,0.00000,0.19200,0.25000,0.20000;
        0.00000,0.00000,0.00000,0.17400,0.50000,0.40000;
        0.63700,0.58200,0.13600,0.00000,0.00000,0.00000;
        0.10500,0.10900,0.65400,0.00000,0.00000,0.00000;
        0.25800,0.30900,0.21000,0.00000,0.00000,0.00000];
newMatrix = fanp_multiMatrix(weightedsupermatrix,weightedsupermatrix);
matrixRecord = weightedsupermatrix;
times = 1;
matrixRecord(:, :, 2) = newMatrix;
circulantCheckResult = fanp_circulantCheck(matrixRecord, newMatrix)
while circulantCheckResult(1) == 0
```

```
times = times + 1;
    disp(times)
    newMatrix = fanp_multiMatrix(newMatrix,weightedsupermatrix);
    matrixRecord(:, :, times+1) = newMatrix;
    circulantCheckResult = fanp_circulantCheck(matrixRecord, newMatrix);
end
disp('total multied :')
disp(times)
disp('times')
```

where the "weighted supermatrix" can be arbitrary, which is derived from an unweighted supermatrix. It needs to use a semicolon to separate each row of the supermatrix. "newMatrix" is the new matrix after an iteration. The variable "time" is to keep count of iterations. Variable "matrixRecord" is a three-dimensional variable used to record the output of each iteration.

After each iteration, "matrixRecord" is called to check whether the "newMatrix" is stable or periodic. Whenever the "newMatrix" reaches a stable or periodic state, the program terminates. Otherwise, the program will keep on iterating. Finally, the total number of iterations will be displayed.

The program for a matrix or supermatrix multiplication is as follows:

```
function array3 = fanp_multiMatrix(array1, array2)
  n = size(array1);
  for i = 1: n
    for j = 1: n
      sum = 0;
      for m = 1: n
        sum = sum + array1(i, m)*array2(m, j);
      end
      array3(i, j) = sum;
    end
  end
```

where the parameters "array1" and "array2" are the results of current iteration, and the return value "array3" records the result of a new iteration.

The following function "fanp_circulantCheck" is for determining whether the new supermatrix is stable or periodic. Function "fanp_equal" will be called in this procedure.

```
function [circulantFlag,rLength,limitedMatrix,cycles] = fanp_circulantCheck(matrixRecord,
newMatrix);
circulantFlag = 0;
limitedMatrix = [ ];
    cycles = 1;
a = size(matrixRecord);
```

```
rLength = a(3);
for i = rLength-1 : -1 : 1
   if fanp_equal(matrixRecord(:, :, i),newMatrix) == 1
      disp('cycle started...')
      circulantFlag = 1;
      limitedMatrix = matrixRecord(:,:,i);
      for j = i + 1 : rLength - 1
         limitedMatrix = limitedMatrix + matrixRecord(:, :, j);
         cycles = cycles + 1;
      end
      limitedMatrix = limitedMatrix / cycles;
      disp('stable matrix  : ')
      limitedMatrix
      disp('cycle : ')
      cycles
      disp('cycle start times : ')
      i
      return
   end
end
```

In the program above, “matrixRecord“ and “newMatrix“ are the input parameters, and “circulantFlag“ is the output and the sign of a cycle. If the supermatrix has a cycle, “1“ is returned. The limit supermatrix, cycles and iterative times of the cycle starting will be displayed.

Comparisons will be made between the iterative output and the existing results one by one. If any two of them are equal, then the cycle of the supermatrix occurs. In this case, the final limit supermatrix is acquired by dividing the summation of all the supermatrices in the cycle by the value of period. For a supermatrix without a cycle, we can assume that its period is one. Therefore, whatever the results of the comparison, we can use the procedure above to obtain the ultimate limit supermatrix.

The following function “fanp_equal“ is used to test whether the new supermatrix is the same as the former one or not.

```
function equalFlag = fanp_equal(array1,array2)
   n = size(array1);
   for i=1: n
         for j=1: n
               if array1(i, j) ~= array2(i, j)
                     disp('the two matrix are not equal')
                     equalFlag = 0;
                     return
                     end
         end
```

```
end
disp('the two matrix are equal')
equalFlag = 1;
return
```

where elements of "array1" and "array2" will be compared one by one. If they are equal, return 1; otherwise, return 0.

4.2.2. Acquiring the limit supermatrix from a supermatrix without a cycle

Example 4. Acquiring the limit supermatrix from the following weighted supermatrix.

$$\begin{pmatrix} 0.0000 & 0.2000 & 0.0000 & 0.0000 \\ 0.2000 & 0.0000 & 0.5000 & 0.0000 \\ 0.4000 & 0.4000 & 0.0000 & 1.0000 \\ 0.4000 & 0.4000 & 0.0000 & 0.0000 \end{pmatrix}$$

As mentioned before, the weighted supermatrix can be specified in the function "fanp_limitedsupermatrix". The result obtained by running it in the command window, as shown in Fig. 5.

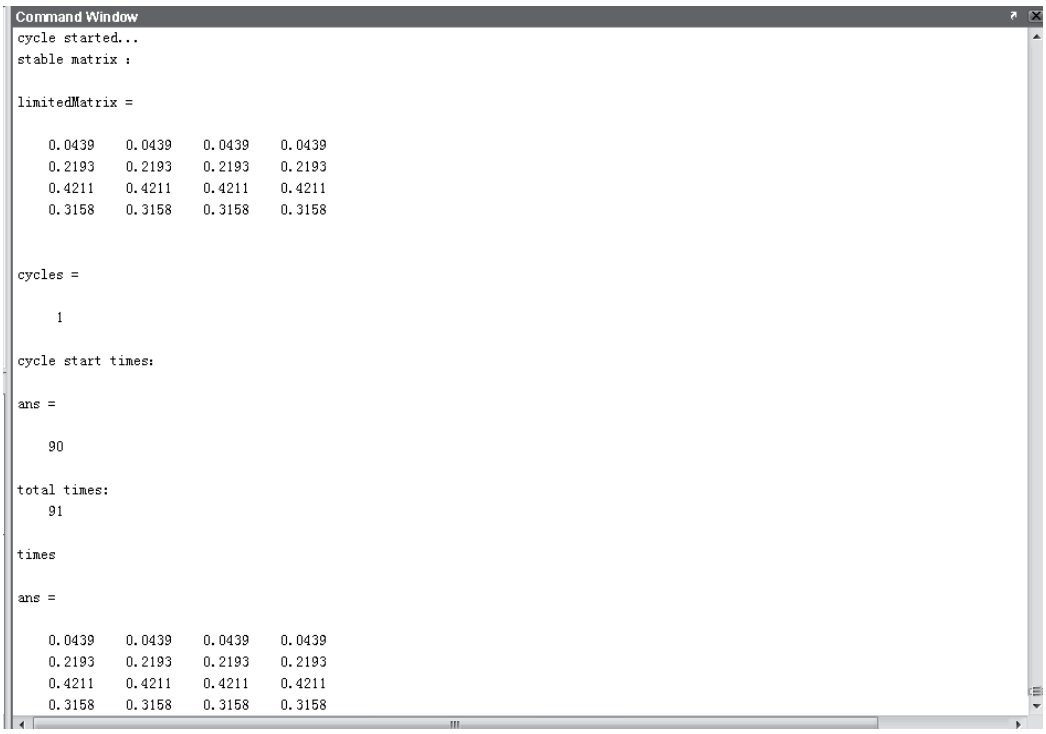


Figure 5. The operation result of Example 4

According to the result, the limit supermatrix is

$$\begin{pmatrix} 0.0439 & 0.0439 & 0.0439 & 0.0439 \\ 0.2193 & 0.2193 & 0.2193 & 0.2193 \\ 0.4211 & 0.4211 & 0.4211 & 0.4211 \\ 0.3158 & 0.3158 & 0.3158 & 0.3158 \end{pmatrix}.$$

where "cycles" is 1 means the limit supermatrix is not periodic, and "cycle start times" is 90 means the cycle appears at the ninetieth iteration. The total number of iterations is 91. The limit supermatrix is obtained at the end of ninetieth iteration though the program was implemented one more time. The local weights are (0.0439, 0.2193, 0.4211, 0.3158) for this supermatrix.

4.2.3. Acquiring the limit supermatrix from a supermatrix with a cycle

Example 5. Acquiring the limit supermatrix from the following supermatrix

$$\begin{pmatrix}
0.000 & 0.250 & 0.266 & 0.086 & 0.269 & 0.100 & 0.250 & 0.269 & 0.000 & 0.000 & 0.000 & 0.000 & 0.133 & 0.133 & 0.083 \\
0.083 & 0.000 & 0.067 & 0.268 & 0.085 & 0.200 & 0.125 & 0.085 & 0.000 & 0.000 & 0.000 & 0.000 & 0.133 & 0.133 & 0.167 \\
0.250 & 0.083 & 0.000 & 0.146 & 0.146 & 0.200 & 0.125 & 0.146 & 0.000 & 0.000 & 0.000 & 0.000 & 0.067 & 0.067 & 0.083 \\
0.123 & 0.106 & 0.104 & 0.000 & 0.263 & 0.132 & 0.230 & 0.206 & 0.290 & 0.315 & 0.298 & 0.250 & 0.000 & 0.000 & 0.000 \\
0.052 & 0.039 & 0.033 & 0.233 & 0.000 & 0.084 & 0.128 & 0.115 & 0.032 & 0.035 & 0.033 & 0.028 & 0.000 & 0.000 & 0.000 \\
0.052 & 0.058 & 0.104 & 0.087 & 0.093 & 0.000 & 0.071 & 0.115 & 0.065 & 0.057 & 0.081 & 0.120 & 0.000 & 0.000 & 0.000 \\
0.075 & 0.106 & 0.059 & 0.127 & 0.093 & 0.230 & 0.000 & 0.064 & 0.081 & 0.057 & 0.055 & 0.074 & 0.000 & 0.000 & 0.000 \\
0.032 & 0.024 & 0.033 & 0.053 & 0.051 & 0.054 & 0.071 & 0.000 & 0.032 & 0.035 & 0.033 & 0.028 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.288 & 0.116 & 0.086 & 0.043 & 0.088 & 0.058 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.286 & 0.000 & 0.321 & 0.268 & 0.150 & 0.154 & 0.156 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.143 & 0.157 & 0.000 & 0.146 & 0.097 & 0.056 & 0.084 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.071 & 0.056 & 0.063 & 0.000 & 0.043 & 0.036 & 0.036 \\
0.177 & 0.102 & 0.102 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.083 & 0.111 \\
0.100 & 0.056 & 0.176 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.111 & 0.000 & 0.222 \\
0.056 & 0.176 & 0.056 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.223 & 0.250 & 0.000
\end{pmatrix}$$

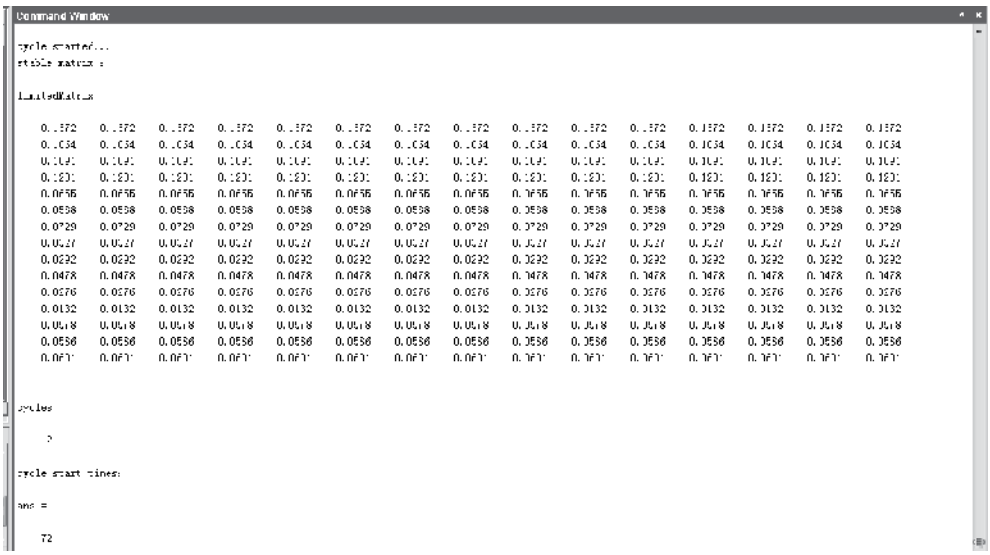


Figure 6. The operation result of Example 5

Specify the weighted supermatrix in the function "fanp_limitedsupermatrix" and run the program in the command window, the result is shown in Fig. 6. Where "cycles" is 2 means the limit supermatrix is periodic, and the period is 2. "cycle start times" is 72 means the cycle appears since the 72nd iteration, and the limit supermatrix is obtained at the end of the 73rd iteration. According to the FPP method, the local weights are (0.1372, 0.1064, 0.1091, 0.1231, 0.0655, 0.0588, 0.0729, 0.0327, 0.0292, 0.0478, 0.0276, 0.0132, 0.0578, 0.0586, 0.0601). This example is actually the weighted supermatrix of the following case in section 5, and the result is the limit supermatrix of the case.

5. Case study

Supposing that a company has the opportunity to select an optimal project from a number of alternatives. Through pre-test, three projects, named as D_1, D_2 and D_3, need further evaluation. A cross-functional project team consists of various departments working to select the best project. Firstly the selection criteria are identified. Then according to the FANP method, the optimal alternative is derived. The decision-making process of choosing an optimal project based on FANP is as follows:

Step 1. Model construction and problem structuring. Taking the interaction among dimensions and attribute-enablers into account, a four-level evaluation index system is proposed, as shown in Fig. 2.

Step 2. Pairwise comparison matrices among dimensions/attributes are formed by the decision committee using the scales given in Table 1, and the scores of the three projects are determined as well. For instance, Table 2. is the pairwise comparison matrix for the profitability (S_1), risk (S_2), owners (S_3) and Bidding competition (S_4) at the dimension of choosing an optimal project. All the fuzzy comparison matrices are produced in the same way.

optimal project	S_1	S_2	S_3	S_4
S_1	EI	$1/IM_2$	1/MI	$1/IM_1$
S_2	IM_2	EI	IM_1	IM_1
S_3	MI	$1/IM_1$	EI	$1/IM_1$
S_4	IM_1	$1/IM_1$	IM_1	EI

Table 2. The comparison matrix using linguisitc scales at the dimension of optimal project

Expert opinions will be converted into the corresponding triangular fuzzy numbers, as shown in Table 3.

optimal project	S_1	S_2	S_3	S_4	Local weights
S_1	(1, 1, 1)	(1/5, 1/4, 1/3)	(1/4, 1/3, 1/2)	(1/3, 1/2, 1)	0.0989
S_2	(3, 4, 5)	(1, 1, 1)	(1, 2, 3)	(1, 2, 3)	0.4240
S_3	(2, 3, 4)	(1/3, 1/2, 1)	(1, 1, 1)	(1/3, 1/2, 1)	0.2544
S_4	(1, 2, 3)	(1/3, 1/2, 1)	(1, 2, 3)	(1, 1, 1)	0.2226
λ=0.6667					

Table 3. The comparison matrix using TFNs at the dimension of optimal project

Step 3. Local weights of the factors and sub-factors which take part in the second and third levels of the ANP model, provided in Fig. 2, are calculated by FPP method. For example, according to equation (8), the local weights of Table 3 can be achieved by solving the following non-linear programming.

$$
\begin{aligned}
&\text{Max}\,\lambda \\
&(1/20)\lambda w_2 - w_1 + (1/5)w_2 \le 0; \\
&(1/12)\lambda w_2 + w_1 - (1/3)w_2 \le 0; \\
&(1/12)\lambda w_3 - w_1 + (1/4)w_3 \le 0; \\
&(1/6)\lambda w_3 + w_1 - (1/2)w_3 \le 0; \\
&(1/6)\lambda w_4 - w_1 + (1/3)w_4 \le 0; \\
&(1/2)\lambda w_4 + w_1 - w_4 \le 0; \\
&\lambda w_3 - w_2 + w_3 \le 0; \\
&\lambda w_3 + w_2 - 3w_3 \le 0; \\
&\lambda w_4 - w_2 + w_4 \le 0; \\
&\lambda w_4 + w_2 - 3w_4 \le 0; \\
&(1/6)\lambda w_4 - w_3 + (1/3)w_4 \le 0; \\
&(1/2)\lambda w_4 + w_3 - w_4 \le 0; \\
&w_1 + w_2 + w_3 + w_4 = 1; \\
&w_1, w_2, w_3, w_4{}^3 \ge 0.
\end{aligned}
$$

	S_{11}	S_{12}	S_{13}	S_{21}	S_{22}	S_{23}	S_{24}	S_{25}	S_{31}	S_{32}	S_{33}	S_{34}	S_{41}	S_{42}	S_{43}
S_{11}	0.000	0.750	0.800	0.171	0.538	0.200	0.500	0.538	0.000	0.000	0.000	0.000	0.400	0.400	0.250
S_{12}	0.250	0.000	0.200	0.536	0.170	0.400	0.250	0.170	0.000	0.000	0.000	0.000	0.400	0.400	0.500
S_{13}	0.750	0.250	0.000	0.293	0.293	0.400	0.250	0.293	0.000	0.000	0.000	0.000	0.200	0.200	0.250
S_{21}	0.368	0.318	0.313	0.000	0.526	0.263	0.458	0.412	0.581	0.631	0.595	0.500	0.000	0.000	0.000
S_{22}	0.155	0.118	0.099	0.467	0.000	0.169	0.256	0.230	0.065	0.070	0.066	0.056	0.000	0.000	0.000
S_{23}	0.155	0.173	0.313	0.174	0.186	0.000	0.143	0.230	0.129	0.115	0.162	0.241	0.000	0.000	0.000
S_{24}	0.226	0.318	0.176	0.253	0.186	0.460	0.000	0.128	0.161	0.115	0.110	0.148	0.000	0.000	0.000
S_{25}	0.095	0.073	0.099	0.107	0.102	0.108	0.143	0.000	0.065	0.070	0.066	0.056	0.000	0.000	0.000
S_{31}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.575	0.231	0.171	0.130	0.263	0.174
S_{32}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.571	0.000	0.644	0.536	0.450	0.460	0.467
S_{33}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.286	0.314	0.000	0.293	0.290	0.169	0.253
S_{34}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.143	0.111	0.125	0.000	0.130	0.108	0.107
S_{41}	0.535	0.307	0.306	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.250	0.333
S_{42}	0.299	0.168	0.527	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.333	0.000	0.667
S_{43}	0.167	0.525	0.167	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.667	0.750	0.000

Table 4. The unweighted supermatrix

As mentioned before, the non-linear programming can be solved by Matlab. The optimal solutions are w_1=0.0989, w_2=0.4240, w_3=0.2544, w_4=0.2226, and λ=0.6667, which shows the

experts' opinions are of good consistency, and the comparison result is acceptable, as shown in Table 3. All the local weights are acquired in the same manner.

Step 4. According to the interdependencies among dimensions and attribute-enablers, an unweighted supermatrix is built, as shown in Table 4.

Step 5. Randomize the unweighted supermatrix to derive the weighted supermatrix.

Step 6. Multiply the weighted supermatrix by itself until the values of each row converge to the same value for every column of the supermatrix. Then we choose any column from the stable limit supermatrix as the local weights of interdependency indicators, as shown in Table 5. It can be solved by Matlab, and the process of calculation is the same as Example 5 in the former section.

	S_{11}	S_{12}	S_{13}	S_{21}	S_{22}	S_{23}	S_{24}	S_{25}	S_{31}	S_{32}	S_{33}	S_{34}	S_{41}	S_{42}	S_{43}
S_{11}	0.137	0.137	0.137	0.137	0.137	0.137	0.137	0.137	0.137	0.137	0.137	0.137	0.137	0.137	0.137
S_{12}	0.106	0.106	0.106	0.106	0.106	0.106	0.106	0.106	0.106	0.106	0.106	0.106	0.106	0.106	0.106
S_{13}	0.109	0.109	0.109	0.109	0.109	0.109	0.109	0.109	0.109	0.109	0.109	0.109	0.109	0.109	0.109
S_{21}	0.123	0.123	0.123	0.123	0.123	0.123	0.123	0.123	0.123	0.123	0.123	0.123	0.123	0.123	0.123
S_{22}	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066	0.066
S_{23}	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059
S_{24}	0.073	0.073	0.073	0.073	0.073	0.073	0.073	0.073	0.073	0.073	0.073	0.073	0.073	0.073	0.073
S_{25}	0.033	0.033	0.033	0.033	0.033	0.033	0.033	0.033	0.033	0.033	0.033	0.033	0.033	0.033	0.033
S_{31}	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029	0.029
S_{32}	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048	0.048
S_{33}	0.028	0.028	0.028	0.028	0.028	0.028	0.028	0.028	0.028	0.028	0.028	0.028	0.028	0.028	0.028
S_{34}	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013
S_{41}	0.058	0.058	0.058	0.058	0.058	0.058	0.058	0.058	0.058	0.058	0.058	0.058	0.058	0.058	0.058
S_{42}	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059	0.059
S_{43}	0.060	0.060	0.060	0.060	0.060	0.060	0.060	0.060	0.060	0.060	0.060	0.060	0.060	0.060	0.060

Table 5. The limit supermatrix

Step 7. Calculate the comprehensive weights of each index, as shown in Table 6.

Step 8. According to equation (11), the scores of each alternative can be calculated, D_1=0.31, D_1=0.33, D_3=0.36, as shown in Table 6. Therefore, we can choose project D_3 as the best one.

6. Conclusions

Taking the interaction and feedback relationships between criteria and/or indicators into account, an evaluation index system for selecting a construction project is proposed. With the uncertain and inaccurate information during the evaluation process being considered, an evaluation and selection model based on fuzzy analytic network process method is presented. The weights of the indices, including the weights of the criterion level indicators, the weights of independent sub-indices and the weights of dependent sub-indices are determined by the fuzzy preference programming method. Meanwhile, an unweighted supermatrix based on its network structure is built for interactional indicators, and the convergent limit supermatrix is calculated after randomizing the unweighted supermatrix. Accordingly, the comprehensive weight of each index and the final score of each alternative

can be calculated. Then we can choose the optimal alternative. A numerical example is given by the proposed method as well.

Two key steps in the process of decision-making with FANP are solved by Matlab. One is acquiring local weights of the fuzzy pairwise comparison matrices; the other is obtaining the limit supermatrix. Matlab is selected for its excellent performance on data processing and matrix operation. Compared with the existing research results, the proposed method fully takes into consideration the interaction and feedback relationships between the dimensions and/or attributes, and it uses triangular fuzzy numbers to represent the preference opinions of experts. It helps to make a more accurate and scientific decision.

index	P_j	A_{ij}^{I}	A_{ij}^{D}	w	w'	S_{1ij}	S_{2ij}	S_{3ij}	d_1	d_2	d_3
S_{11}	0.099	0.538	0.137	0.007	0.104	0.381	0.333	0.286	0.039	0.035	0.030
S_{12}	0.099	0.170	0.106	0.002	0.025	0.300	0.300	0.400	0.008	0.008	0.010
S_{13}	0.099	0.293	0.109	0.003	0.045	0.263	0.368	0.368	0.012	0.017	0.017
S_{21}	0.424	0.361	0.123	0.019	0.268	0.286	0.333	0.381	0.076	0.089	0.102
S_{22}	0.424	0.243	0.066	0.007	0.096	0.267	0.400	0.333	0.026	0.038	0.032
S_{23}	0.424	0.147	0.059	0.004	0.052	0.304	0.304	0.391	0.016	0.016	0.020
S_{24}	0.424	0.147	0.073	0.005	0.065	0.214	0.357	0.429	0.014	0.023	0.028
S_{25}	0.424	0.102	0.033	0.001	0.020	0.375	0.250	0.375	0.008	0.005	0.008
S_{31}	0.254	0.228	0.029	0.002	0.024	0.273	0.318	0.409	0.007	0.008	0.010
S_{32}	0.254	0.571	0.048	0.007	0.099	0.364	0.318	0.318	0.036	0.031	0.031
S_{33}	0.254	0.124	0.028	0.001	0.012	0.286	0.333	0.381	0.004	0.004	0.005
S_{34}	0.254	0.077	0.013	0.000	0.004	0.333	0.286	0.381	0.001	0.001	0.001
S_{41}	0.223	0.170	0.058	0.002	0.031	0.400	0.250	0.350	0.012	0.008	0.011
S_{42}	0.223	0.300	0.059	0.004	0.056	0.421	0.263	0.316	0.023	0.015	0.018
S_{43}	0.223	0.529	0.060	0.007	0.101	0.286	0.333	0.381	0.029	0.034	0.038
The score of alternative D_k									0.31	0.33	0.36

Table 6. The comprehensive weights and the ranking of the alternatives

Author details

Xiaoguang Zhou
Dongling School of Economics and Management, University of Science and Technology Beijing, Beijing China

Acknowledgement

The author is very grateful to the editor, Dr. Vasilios Katsikis for his valuable suggestions, and Robert Ulbrich who gives constructive comments and suggestions on English grammar,

which have been very helpful in improving the book. This work was supported by "the Fundamental Research Funds for Chinese Central Universities (No. FRF-BR-11-009A)".

7. References

Amiri, Morteza Pakdin. (2010). Project selection for oil-fields development by using the AHP and fuzzy TOPSIS methods. *Expert Systems with Applications*, vol. 37, pp. 6218-6224

Aragonés-Beltrán, P.; Chaparro-González, F.; Pastor-Ferrando, J. P. & Rodríguez-Pozo, F. (2010). An ANP-based approach for the selection of photovoltaic solar power plant investment projects. *Renewable and Sustainable Energy Reviews*, vol. 14, pp. 249-264

Arunraj, N. S. & Maiti, J. (2010). Risk-based maintenance policy selection using AHP and goal programming. *Safety Science*, vol. 48, pp. 238-247

Ayağ, Z. & Özdemir, R. G. (2009). A hybrid approach to concept selection through fuzzy analytic network process. *Computers & Industrial Engineering*, vol. 56, pp. 368–379

Bhattacharyya, Rupak.; Kumar, Pankaj. & Kar, Samarjit. (2011). Fuzzy R&D portfolio selection of interdependent projects. *Computers and Mathematics with Applications*, vol. 62, pp. 3857-3870

Boran, Semra. & Goztepe, Kerim. (2010). Development of a fuzzy decision support system for commodity acquisition using fuzzy analytic network process. *Expert Systems with Applications*, vol. 37, pp. 1939-1945

Buckley, J. J. (1985). Fuzzy hierarchical analysis. *Fuzzy Sets and Systems*, vol. 17, pp. 233-247

Chang, D. Y. (1996). Applications of the extent analysis method on fuzzy AHP. *European Journal of Operational Research*, vol. 95, pp. 649-655

Chang, P. T. & Lee, J. H. (2012). A fuzzy DEA and knapsack formulation integrated model for project selection. *Computers & Operations Research*, vol. 39, pp. 112–125

Cheng, E. W. L. & Li, H. (2005). Analytic Network Process Applied to Project Selection. *Journal of Construction Engineering and Management*, vol. 131 (4), pp. 459-466.

Csutora, R. & Buckley, J. J. (2001). Fuzzy hierarchical analysis: The Lamda-Max method. *Fuzzy Sets and Systems*, vol. 120, pp. 181-195

Dağdeviren, Metin. & Yüksel, İhsan. (2010). A fuzzy analytic network process (ANP) model for measurement of the sectoral competititon level (SCL). *Expert Systems with Applications*, vol. 37, pp. 1005-1014

Dey, Prasanta Kumar. (2006). Integrated project evaluation and selection using multiple-attribute decision-making technique. *Int. J. Production Economics*, vol. 103, pp. 90–103

Ebrahimnejad, S.; Mousavi, S. M.; Tavakkoli-Moghaddam, R.; Hashemi, H. & Vahdani, B. (2011). A novel two-phase group decisionmaking approach for construction project selection in a fuzzy environment. *Applied Mathematical Modelling*. doi: 10. 1016/j. apm. 2011. 11. 050

Fang, Yong.; Chen, Lihua. & Fukushima, Masao. (2008). A mixed R&D projects and securities portfolio selection model. *European Journal of Operational Research*, vol. 185, pp. 700-715

Gabriel, Steven A.; Kumar, Satheesh.; Ordóňez, Javier. & Nasserian, Amirali. (2006). A multiobjective optimization model for project selection with probabilistic considerations. *Socio-Economic Planning Sciences*, vol. 40, pp. 297-313

Ghorbani, S. & Rabbani, M. (2009). A new multi-objective algorithm for a project selection problem. *Advances in Engineering Software*, vol. 40, pp. 9-14

Gutjahr, Walter J.; Katzensteiner, Stefan,; Reiter, Peter,; Stummer, Christian. & Denk, Michaela. (2010). Multi-objective decision analysis for competence-oriented project portfolio selection. *European Journal of Operational Research*, vol. 205, pp. 670-679

Huang, Chi-Cheng.; Chu, Pin-Yu. & Chiang, Yu-Hsiu. (2008). A fuzzy AHP application in government-sponsored R&D project selection. *Omega*, vol. 36, pp. 1038-1052

Huang, Ivy B.; Keisler, Jeffrey. & Linkov Igor. (2011). Multi-criteria decision analysis in environmental sciences- Ten years of applications and trends. *Science of the Total Environment*, vol. 409, pp. 3578-3594

Huo, Liang-an.; Lan, Jibin. & Wang, Zhongxing. (2011). New parametric prioritization methods for an analytical hierarchy process based on a pairwise comparison matrix. *Mathematical and Computer Modelling*, vol. 54, pp. 2736-2749

Ju, Yanbing.; Wang, Aihua. & Liu, Xiaoyue. (2012). Evaluating emergency response capacity by fuzzy AHP and 2-tuple fuzzy linguistic approach. *Expert Systems with Applications*, vol. 39, pp. 6972-6981

Jung, U. & Seo, D. W. (2010). An ANP approach for R&D project evaluation based on interdependencies between research objectives and evaluation criteria. *Decision Support Systems*, vol. 49, pp. 335-342

Kahraman, Cengiz.; Ertay, Tijen. & Büyüközkan, Gülçin. (2006). A fuzzy optimization model for QFD planning process using analytic network approach. *European Journal of Operational Research*, vol. 171, pp. 390-411

Kumar, D. P. (2006). Integrated project evaluation and selection using multiple-attribute decision-making technique. *International Journal of Production Economics*, vol. 103, pp. 90-103

Lee, Hakyeon.; Kim, Chulhyun.; Cho, Hyunmyung. & Park, Yongtae. (2009). An ANP-based technology network for identification of core technologies: A case of telecommunication technologies. *Expert Systems with Applications*, vol. 36: 894-908.

Liesiö, Juuso.; Mild, Pekka. & Salo, Ahti. (2007). Preference programming for robust portfolio modeling and project selection. *European Journal of Operational Research*, vol. 181, pp. 1488-1505

Lin, C. T. & Chen, Y. T. (2004). Bid/no-bid decision-making-a fuzzy linguistic approach. *International Journal of Project Management*, vol. 22, pp. 585-593

Mikhailov, L. (2003). Deriving priorities from fuzzy pairwise comparison judgements. *Fuzzy Sets and Systems*, vol. 134, pp. 365-385

Mikhailov, L. (2004). Group prioritization in the AHP by fuzzy preference programming method. *Computers & Operations Research*, vol. 31, pp. 293-301

Mohanty, R. P. (1992). Project selection by a multiple-criteria decision making method: An example from a developing country. *International Journal of Project Management*, vol. 10, pp. 31-38

Pires, Ana.; Chang, Ni-Bin. & Martinho, Graça. (2011). An AHP-based fuzzy interval TOPSIS assessment for sustainable expansion of the solid waste management system in Setúbal Peninsula, Portugal. *Resources, Conservation and Recycling*, vol. 56, pp. 7-21

Promentilla, Michael Angelo B.; Furuichi, T. ; Ishii, K. & Tanikawa, N. (2008). A fuzzy analytic network process for multi-criteria evaluation of contaminated site remedial countermeasures. *Journal of Environmental Management*, vol. 88, pp. 479–495

Saaty, T. L. (1996). *Decision-making with Dependence and Feedback: The Analytic Network Process*, RWS Publications, Pittsburgh, PA

Shakhsi-Niaei, M.; Torabi, S. A. & Iranmanesh, S. H. (2011). A comprehensive framework for project selection problem under uncertainty and real-world constraints. *Computers & Industrial Engineering*, vol. 61, pp. 226-237

Smith-Perera, Aida.; García-Melón, Mónica.; Poveda-Bautista, Rocío. & Pastor-Ferrando, Juan-Pascual. (2010). A Project Strategic Index proposal for portfolio selection in electrical company based on the Analytic Network Process. *Renewable and Sustainable Energy Reviews*, vol. 14, pp. 1569-1579

Srdjevic, Bojan. (2005). Combining different prioritization methods in the-analytic hierarchy process synthesis. *Computers & Operations Research*, vol. 32, pp. 1897-1919

Van Laarhoven, P. J. M. & Pedrycz, W. (1983). A fuzzy extension of Saaty's priority theory. *Fuzzy Sets and Systems*, vol. 11, pp. 229-241

Wang, J. & Hwang, W. L. (2007). A fuzzy set approach for R&D portfolio selection using a real option valuation model. *Omega*, vol. 35, pp. 247-57

Wang, J.; Xu,Y. J. & Li, Z. (2009). Research on project selection system of pre-evaluation of engineering design project bidding. *International Journal of Project Management*, vol. 27, pp. 584-599

Wang, Y. M.; Elhag, T. M. S. & Hua, Z. S. (2006). A modified fuzzy logarithmic least squares method for fuzzy analytic hierarchy process. *Fuzzy Sets and Systems*, vol. 157, pp. 3055-3071

Xu, R. (2000). Fuzzy least-squares priority method in the analytic hierarchy process. *Fuzzy Sets and Systems*, vol. 112, pp. 359-404

Yu, Jing-Rung. & Cheng, Sheu-Ji. (2007). An integrated approach for deriving priorities in analytic network process. *European Journal of Operational Research*, vol. 180, pp. 1427-1432

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, vol. 8, pp. 338-353

Chapter 7

Fractal Dimension Estimation Methods for Biomedical Images

Antonio Napolitano, Sara Ungania and Vittorio Cannata

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/48760

1. Introduction

The use of medical images has its main aim in the detection of potential abnormalities. This goal is accurately achieved with the synergy between the ability in recognizing unique image patterns and finding the relationship between them and possible diagnoses. One of the methods used to aid this process is the extrapolation of important features from the images called texture; texture is an important source of visual information and is a key component in image analysis.

The current evolution of both texture analysis algorithms and computer technology made boosted development of new algorithms to quantify the textural properties of an image and for medical imaging in recent years. Promising results have shown the ability of texture analysis methods to extract diagnostically meaningful information from medical images that were obtained with various imaging modalities such as positron emission tomography (PET) and magnetic resonance imaging (MRI). Among the texture analysis techniques, fractal geometry has become a tool in medical image analysis. In fact, the concept of fractal dimension can be used in a large number of applications, such as shape analysis[1] and image segmentation[2]. Interestingly, even though the fact that self-similarity can hardly be verified in biological objects imaged with a finite resolution, certain similarities at different spatial scales are quite evident. Precisely, the fractal dimension offers the ability to describe and to characterize the complexity of the images or more precisely of their texture composition.

2. Fractals

2.1. Fractal geometry

A fractal is a geometrical object characterized by two fundamental properties: *Self-similarity* and *Hausdorff Besicovich dimension*. A self-similar object is exactly or approximately similar to a part of itself and that can be continuously subdivided in parts each of which is (at least approximately) a reduced-scale copy of the whole. Furthermore, a fractal generally

shows irregular shapes that cannot be simply described by Euclidian dimension, but, fractal dimension (fd) has to be introduced to extend the concept of dimension to these objects. However, unlike topological dimensions the fd can take non-integer values, meaning that the way a fractal set fills its space is qualitatively and quantitatively different from how an ordinary geometrical set does.

Nature presents a large variety of fractal forms, including trees, rocks, mountains, clouds, biological structures, water courses, coast lines, galaxies[3]. Moreover, it is possible to construct mathematical objects which satisfy the condition of self-similarity and that present fd (Figure 1).

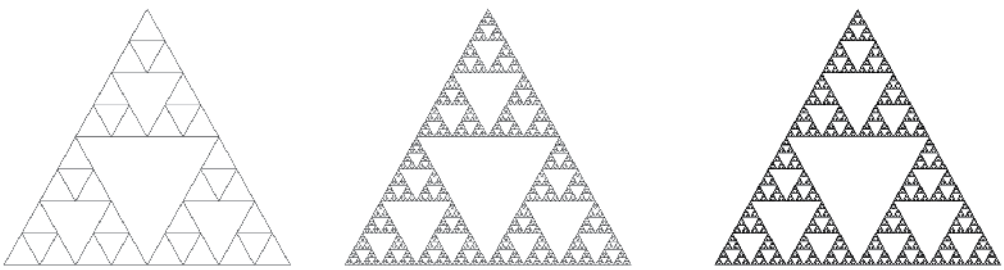


Figure 1. Sierpinski triangle: starting with a simple initial configuration of units or with a geometrical object then the simple seed configuration is repeatedly added to itself in such way that the seed configuration is regarded as a unit and in the new structure these units are arranged with respect to each other according to the same symmetry as the original units in the seed configuration. And so on.

The objects in Figure 1 are self-similar since a part of the object is similar to the whole and the fractal dimension can be calculated by the equation:

$$D = \frac{\log N}{\log S} \tag{1}$$

where N is the number of the auto-similar parts in which an object can be subdivided and S is the scaling, that is, the factor needed to observe N auto-similar parts. According to the Eq.1, the following values are obtained for the Koch fractal and the Sierpinski triangle:

$$D_{Koch} = \frac{\log 4}{\log 3} \approx 1.26 \qquad D_{Sierpinski} = \frac{\log 3}{\log 3} \approx 1.58 \tag{2}$$

In mathematics, no universal definition of fd exists and the several definitions of fd may lead to different results for the same object. Among the wide variety of fd definitions that have been introduced, the Hausdorff dimension D_H is surely the most important and the most widely used[4]. Such definition can be theoretically applied to every fractal set but has the disadvantage it cannot always be easily determined by computational methods.

2.2. Hausdorff dimension D_H

Hausdorff dimension D_H was introduced in 1918 by mathematical Felix Hausdorff [3]. Since many of the technical developments used to compute the Hausdorff dimension for highly irregular sets were obtained by Abram Samoilovitch Besicovitch, D_H is sometimes called Hausdorff-Besicovitch dimension.

Hausdorff formulation[3] is based on the construction of a particular measure, H_δ^D, representing the uniform density of the fractal object.

Intuitively we can sum up the construction as follows: let be A a fractal and $C(r, A) = \{B_1....B_k\}$ a complete coverage of A consisting of spheres of diameter smaller than a given r that approximate A, so $\delta_i = \delta_i(B_i) < r$.

We define the Hausdorff measure as the function H_δ^D that identifies the smallest of all the covering spheres for A with $\delta < r$:

$$H_\delta^D(A) = \omega_D \lim_{r \to 0} \{inf \sum_i \delta_i^D\} \tag{3}$$

with ω_D volume of the unit sphere in R^D for integer D.

We obtain an approximate measurement of A, the so-called *course-grained volume*[4].

In the one-dimensional case ($D = 1$), H_δ^D supplies the length of set A measured with a ruler of length r. The shorter the ruler, the longer the length measured, a paradox known as the *coastline paradox*[3].

Hence, when $r \to 0$ the effective length of A is well approximated. Limit for small r calculated for other values of D, however, lead to a degenerate H_δ^D:

$$\mathcal{H}_\delta^D \to 0 \quad \text{and} \quad \mathcal{H}_\delta^D \to \infty \tag{4}$$

Therefore, D_H can be defined as the transition point for the function H_δ^D monotonically decreasing with D:

$$D_H(A) = inf_{D>0} \{H_\delta^D(A) = 0\} \tag{5}$$

with H_δ^D the D-dimensional Hausdorff measure given by Eq. 3.

The *course-grained volume* defined by Eq. 3 normally presents a scaling like:

$$H_\delta^D \sim r^{(D-D_H)} \tag{6}$$

that provides a method to estimate the dimension D_H.

In the uni-dimensional case $D = 1$ we can easily obtain:

$$L_\delta^D \sim r^{(1-D_H)} \qquad \text{with } L_\delta^D = \textit{measured length} \tag{7}$$

from which we derive D_H.

3. Methods

Although the definition of Hausdorff dimension is particularly useful to operatively define the fd, that presents difficulties when implementing it. In fact, determining the lower bound value of all coverings, as defined in Eq. 5, can be quite complex. For example, let's consider the uni-dimensional case, in which we want to compute the fd of a coastline (Koch Curve). According to Eq. 3 in the case of $D = 1$ the coastline length is measured by a ruler of length r. Accuracy of the measure increase with decreasing r. For $r \to 0$ the coastline will have infinite length. Similar arguments can be applied to $D = 2$; for $r \to 0$ the measure of $H_\delta^D \to 0$.

This discussion implies that our coastline (ex. Koch Curve) will have a fd value more than one-dimensional and less than two- dimensional. For this reason, the fd is considered as the transition point (the lower bound value in Eq. 5) between $H_\delta^D \to 0$ and $H_\delta^D \to \infty$.

Several computational approaches have been developed to avoid the need of defining the lower bound at issue. Therefore many strategies accomplished the fd computation by retrieving it from the scaling of the object's bulk with its size. In fact, object's bulk and its size have a linear relationship in a logarithmic scale so that the slope of the best fitting line may provide an accurate estimation of this relationship. By using this log-log graph, called *Richardson's plot*, the requirement of knowing the infimum over all coverings is relaxed.

Several approaches have been developed to estimate fractal dimension of images. In particular, this section will introduce two fractal analysis strategies: the *Box Counting Method* and the *Hand and Dividers Method*.

These methods overcome the problem by choosing as covering a simple rectangle fixed grid in order to obtain an upper bound on D_H.

Five algorithms for a practical fd calculation based on these methods will also be presented.

3.1. Box counting method

The most popular method using the best fitting procedure is the so-called *Box Counting Method*[5][6]. Given a fractal structure A embedded in a d-dimensional volume the box-counting method basically consists of partitioning the structure space with a d-dimensional fixed-grid of square boxes of equal size r.

The number $N(r)$ of nonempty boxes of size r needed to cover the fractal structure depends on r:

$$N(r) \sim r^{-D} \tag{8}$$

The box counting algorithm hence counts the number $N(r)$ for different values of r and plot the log of the number $N(r)$ versus the log of the actual box size r. The value of the box-counting dimension D is estimated from the Richardson's plot best fitting curve slope.

$$-D = \lim_{r \to 0} \frac{\log N(r)}{\log r} \tag{9}$$

Figure 2 shows the Box counting method for the Koch Curve.

Several algorithms[7][8][9] based on box counting method have been developed and widely used for fd estimation, as it can be applied to sets with or without self-similarity. However, in computing fd with this method, one either counts or does not count a box according to whether there are no points or some points in the box. No provision is made for weighting the box according to the number of points belonging to the fractal and inside the current box.

3.2. Hand and dividers method

Useful features and information can be deducted from the contours of structures belonging to an image and there is a number of techniques that can be used when estimating the boundary fractal dimension.

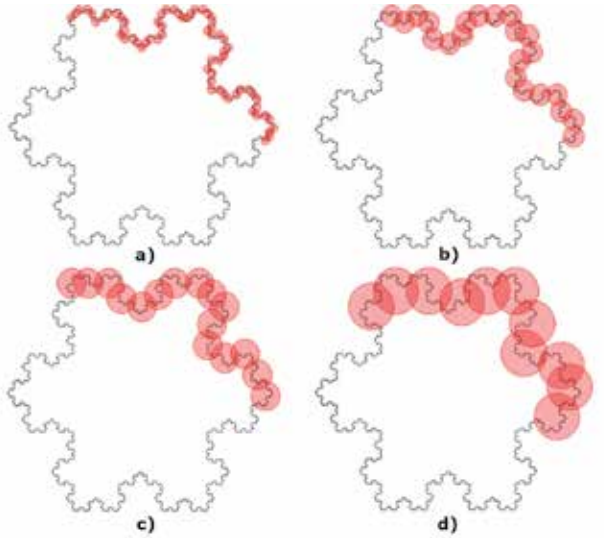


Figure 2. The Box-counting method applied to the Koch Curve with box size r = 0.4 (a); r = 1 (b); r = 1.4 (c); r= 2 (d)

The most popular methods are all based on the *Hand and Divers Method* which was originally introduced by Richardson[10] and successively developed by Mandelbrot[11].

The Richardson method employs the so-called *walking technique* consisting of "walking" around the boundary of the structure with a given step length.

The actual structure boundary is so approximated by a polygon whose length is equal to:

$$l(\epsilon) = \epsilon\, n(\epsilon) \tag{10}$$

In a nutshell, it corresponds to the length of the single step multiplied by the number of steps needed to complete the walk.

The process is then reiterated for different step lengths:

$$P_i = l(\epsilon_i) = \epsilon_i\, n(\epsilon_i) \tag{11}$$

With P_i the perimeter calculated with steps of length ϵ_i.

The object's boundary fd D is finally estimate from:

$$D = 1 - m \tag{12}$$

where m is the slope of the Richardson's plot.

The perimeter length of the boundary depends on the step length used so that a large step provides a rough estimation of the perimeter whereas a smaller step can take into account finer details of the contour.

Consequently, if the step length ϵ decreases the perimeter P increases.

In practice, the perimeter length is obtained by constructing a generally irregular polygon which approximate the border. Let δB be the set of coordinates of object boundary and let be

ϵ a fixed step length. Given a starting point, an arbitrary contour point (x_s, y_s), the next point on the boundary (x_{s_2}, y_{s_2}) in a fixed direction (e.g. clockwise) is the point that has a distance

$$d_i = \sqrt{(x_s - x_{s_2})^2 + (y_s - y_{s_2})^2} \tag{13}$$

as close as possible to ϵ.

The reached point then becomes the new starting point and is used to locate the next point on the boundary that satisfies the previous condition. This process is repeated until the initial starting point is reached.

The sum of all distances dj corresponds to the irregular polygon perimeter (Figure3).

A number of different perimeters for each polygon at each fixed step length are used to build the Richardson's plot and the slope of its best linear fit is exploited to estimate the fd.

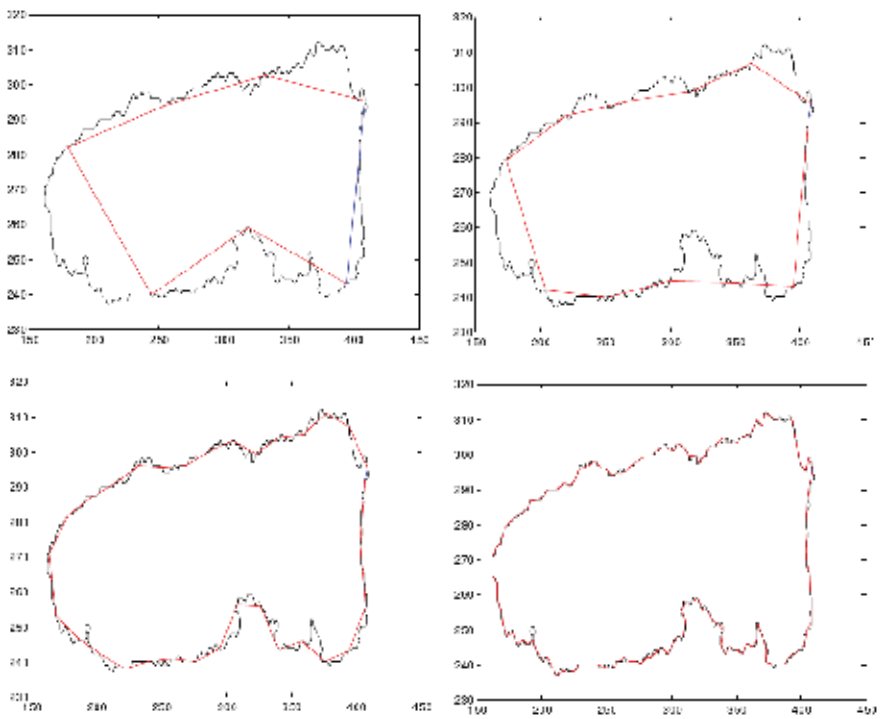


Figure 3. Walking technique applied to a coastline with different step lengths.

4. Algorithms

All Hand and dividers techniques rely on the same identical principle that attempt to approximate the border perimeter with a different polygons. However, since the point coordinates belonging to border set are discrete, all the implemented methods differ in the choice of which point in the set has a distance that better approximate the step length.

The following two methods are the implementations of two different choices about how to overcome this particular issue.

4.1. HYBRID algorithm

The HYBRID algorithm is a computer implementation of Hand and Dividers method developed by Clark[12]. Let δB be the boundary of the object whose fd we wish to compute. The main part of the method focuses on the perimeter estimation and the corresponding Richardson's plot is then attained by reiterating this hard core part at different step size. Figure 4 shows the flow chart of the method.

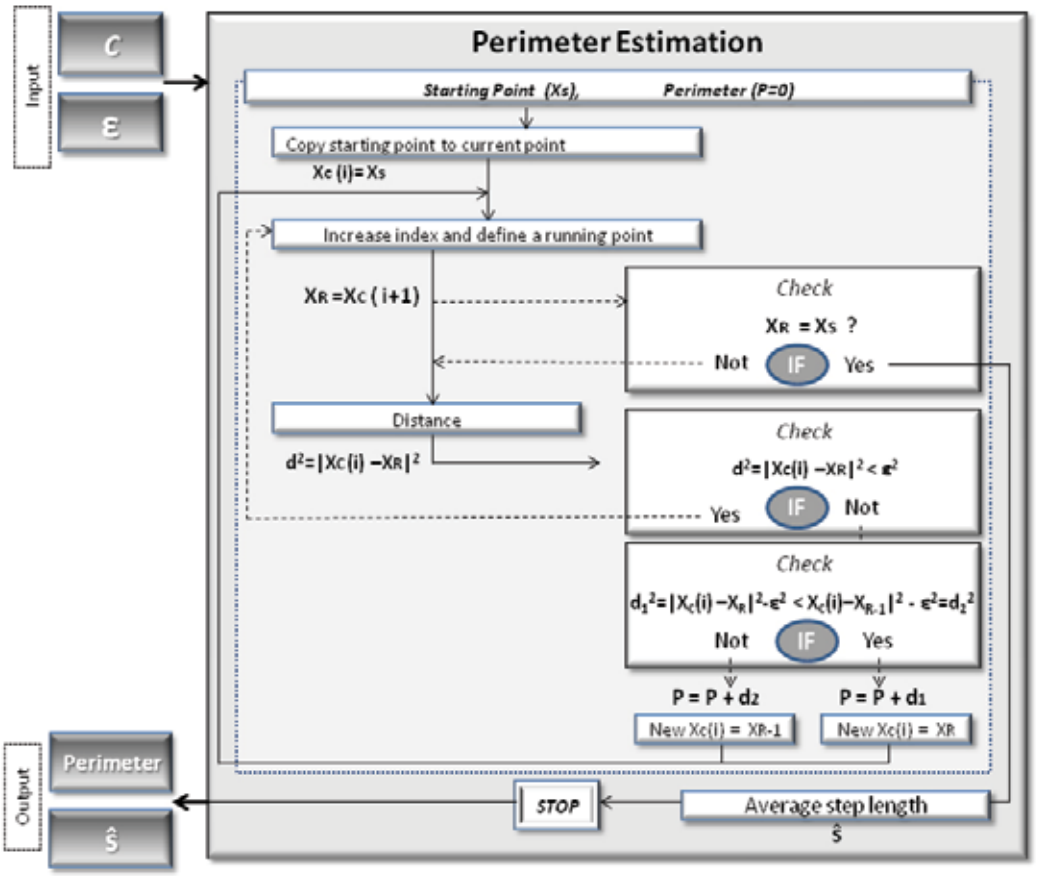


Figure 4. Perimeter estimation by HYBRID method flowchart: an arbitrary *starting point* S (x_S, y_S) on the boundary line is chosen and copied in a new variable, which is called *current point* C $(x_C(i), y_C(i))$. The index i runs through the total number of coordinate points and is iteratively increased defining the *running point* R with coordinates (x_R, y_R). The distance d between S and R is calculated and a check on d when smaller than a fixed step length ϵ is done. The process is repeated until a boundary point whose distance from (x_S, y_S) is larger than the step ϵ is reached. The next pivot point on the boundary line is determined by choosing between the two points the closest to the step length. The distance is then stored and this point becomes the new starting point in order to calculate the next pivot point and so on, until the initial starting point S is reached.

Given an arbitrary *starting point* S and its coordinates (x_S, y_S) on the boundary, the algorithm searches for the next pivot point. In particular the starting point is copied into a *current point*,

C (x_C, y_C), which identifies all points having a mutual distance of about ϵ. The actual point running through the entire border is indicated as *running point* R (x_R, y_R).

Therefore the program searches for a specific running point having a distance from C as near as possible to the step ϵ. In particular, in the HYBRID method the real step may be chosen to be longer or shorter than the fixed step depending on the minimum deviation from it. Similarly once the running point hits a contour point having a distance from the actual current one bigger than the size step, the choice is made between that point and the preceding one.

Afterward, the computed distance between these two points R and C is stored and the running point becomes the new current point.

The procedure continues until the initial starting point is reached. Obviously it is likely that after a complete walking the starting point S may be reached before having hit the following current point C. In other words, there may not be a multiple of step size ϵ so that the final incomplete step length r is added to the others stored distances, whose sum represent the boundary's perimeter. Since the fixed step length is adapted every time during the perimeter computation, its averaged value is then computed and used in the Richardson's plot.

4.2. EXACT algorithm

The EXACT algorithm was proposed for the first time by Clark in 1986[12]. As it will be shown, this method requires a longer computational time by providing a simpler solution to the choice of the best current points.

Similarly to the previous method the entire perimeter estimation is displayed in the flow chart of Figure 5.

The procedure is very similar to the one used for the previous method. As before (see Figure 5), the end of the step may not coincide with the digitized coordinates of the boundary.

The way the EXACT method attempts to overcome this problem relies on the assumption of piecewise linearity, meaning that all the points on the contour can be joined by a series of straight line[13, 14] (see Figure 6 (a)).

The location of the next current point C on the boundary from the one previously determined is schematically illustrated in Figure 6 (b).

The procedure starts from an arbitrary starting point (x_S, y_S) and the algorithm searches for the next pivot point. In particular the starting point is copied into a *current point*, C (x_C, y_C), which identifies all points having a mutual distance of about ϵ. The actual point running through the entire border is indicated as *running point* R (x_R, y_R).

The distance from the current point to each point on the contour line is then calculated until the step length ϵ falls between two consecutive boundary points, (x_R, y_R) and (x_{R+1}, y_{R+1}) for which:

$$\sqrt{(x_R - x_C)^2 + (y_R - y_C)^2} < \epsilon < \sqrt{(x_{R+1} - x_C)^2 + (y_{R+1} - y_C)^2} \tag{14}$$

The exact position of the point N with coordinates (x, y) is deduced by a process of geometric interpolation between the two consecutive running points (x_R, y_R) and (x_{R+1}, y_{R+1}). This

point then becomes the new current point and is used to calculate the next boundary point and so on.

The process is stopped when we come back to the initial starting point (x_S, y_S) in order to obtain a polygon as is shown in Figure 8.

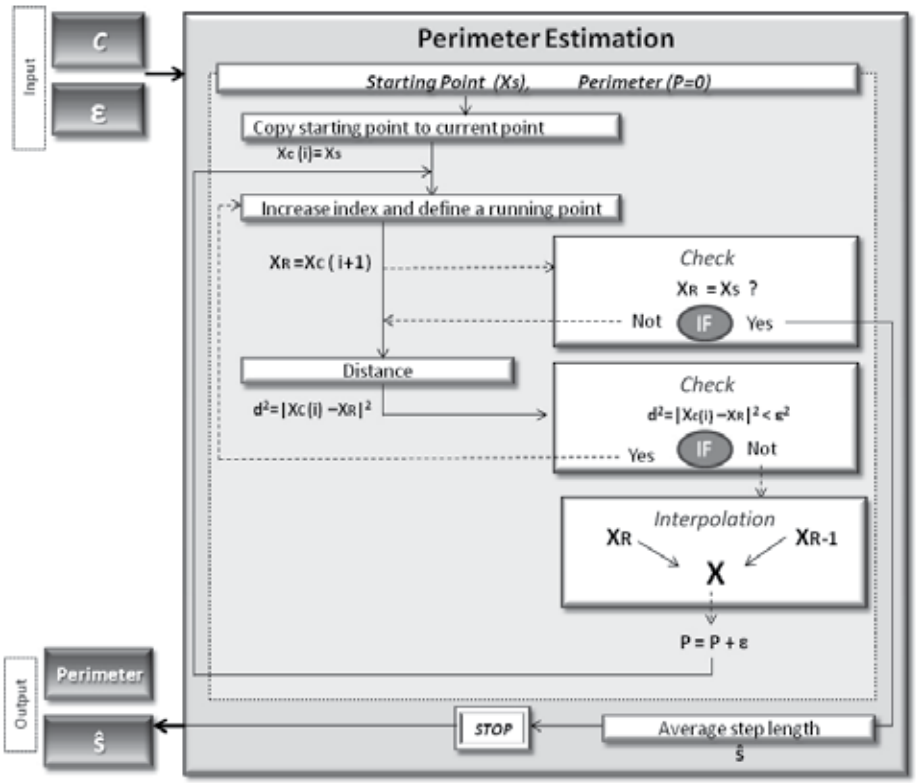
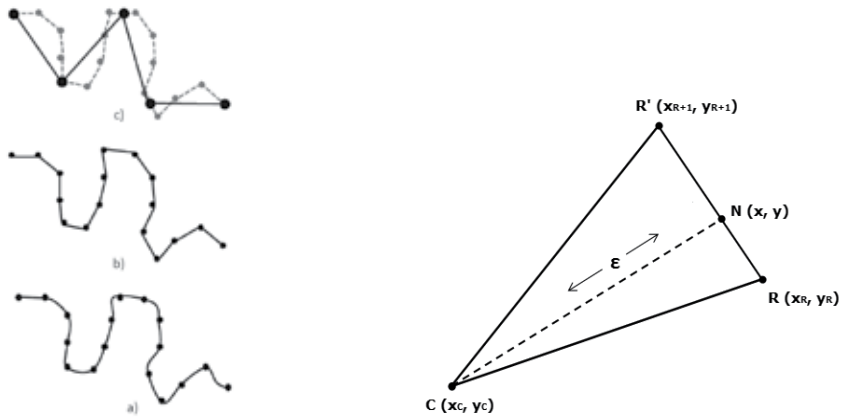


Figure 5. Perimeter estimation by EXACT method flowchart: an arbitrary *starting point* S (x_s, y_s) on the boundary line is chosen and copied in a new variable, which is called *current point* C $(x_c(i), y_c(i))$. The index i runs through the total number of coordinate points and is iteratively increased defining the *running point* R with coordinates (x_R, y_R). The distance d between S and R is calculated and a check on d when smaller than a fixed step length ϵ is done. The process is repeated until a boundary point whose distance from (x_s, y_s) is larger than the step ϵ is reached. The exact position of the next pivot point (x, y) on the boundary line is determined by interpolating the two consecutivepoints (x_R, y_R) and (x_{R+1}, y_{R+1}).
The point (x, y) becomes the new starting point in order to calculate the next pivot point and so on, until the initial starting point S is reached.

The perimeter length of the polygon is found by adding the final incomplete step length to the sum of the other step lengths needed to entirely cover the boundary.

The procedure is then repeated for different step lengths[15].

The results, i.e., perimeter lengths versus step lengths, are plotted on a log-log Richardson's Plot. From the slope of the fitting line on the Richardson's plot we obtain the fd of the examined boundary[1, 4, 12, 16–21]



(a) (b)

Figure 6. a) The piece-wise linear assumption (a) (b) and the EXACT algorithm (c); b)Geometric EXACT interpolation scheme, with S starting point given by (x_C, y_C) coordinates, R and R' two consecutive boundary running points respectly given by (x_R, y_R) and (x_{R+1}, y_{R+1}) coordinates, N new current point obtained by the interpolation between R and R' and ϵ a fixed step length.

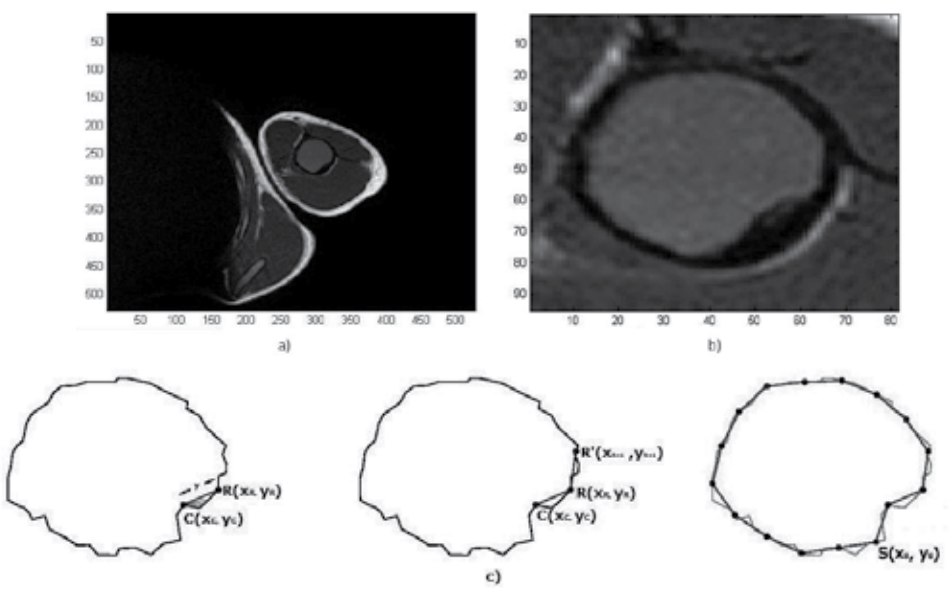


Figure 7. MRI image of an aneurismatic bone cyst (a), (b). Walking technique applied to an aneurysmatic bone cyst boundary (c).

4.3. Box-counting algorithm

The Box-counting algorithm implementation of box-counting method relies on the basic idea of covering a given digital binary image with a set of measuring boxes of sizes S and then to count the number of boxes which actually contain the image.

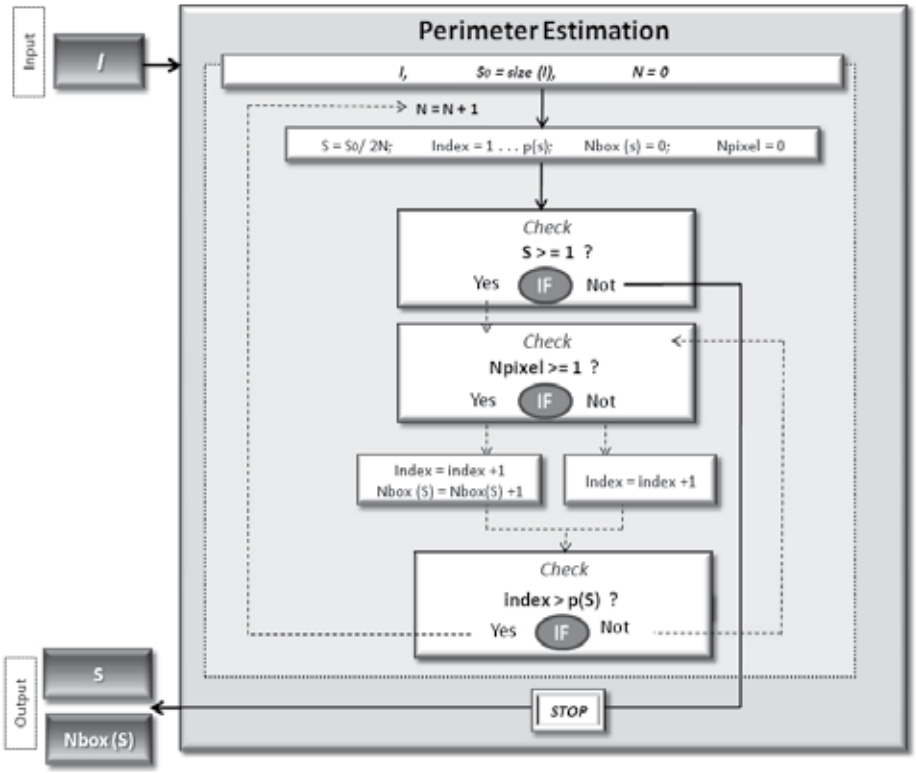


Figure 8. The Box-counting algorithm flowchart: given an image I, its size, S_0, is set as the maximum size from which the computer program starts to calculate the others decreasing box-sizes according to $S = S_0/2^N$. The S value has minimum value which is equal to the pixel size. The number p indicates the total number of box size S. The next step is a check on whether at least one pixel is in the box: if the box is non-empty, the check is stopped when one pixel is found. The procedure continues until the maximum index $p(s)$ is reached. Then the number $Nboxes(S)$ for a given size S is stored and the process restarts with a different box size. When the minimum box size is reached the program stops and gives the output variables of $Nboxes$ and the size value. Using the Eq. 8 the fractal dimension D can be estimate, from the least square linear fit.

Figure 8 shows the flow chart for box-counting fd estimation and for different box sizes. Moreover, since the procedure of size scaling ($S = S_0/2^N$ with N number of iterations) may be not always applicable to any image matrix size, image padding with background pixels is performed.

Therefore the final image I has a dimension that is a power of 2. This can be easily implemented by using *padarray* matlab function.

4.4. Differential Box-counting algorithm (DBC)

The box counting method is an extremely powerful tool for fd computation; in fact, it is easy to implement as well as flexible and robust.

However, a major limitation lies on the fact that the counting process of nonempty boxes implies its use only for binary images rather than gray scale ones. An extension of the

standard approach to gray scale images is called the *Differential Box Counting (DBC)* and has been proposed in 1994 by N. Sarkar and Chaudhuri[8].

In the DBC method, a gray level image I is considered as a 3-D spatial surface with (x, y) denoting the pixels spatial coordinates and the third axis z the pixels gray level.

As for the standard box counting, the $M \times M$ image matrix is partitioned into non-overlapping $s \times s$-sized boxes, where s is an integer falling in the interval $[M/2, 1]$.

Then, the scale of each block is $r = s$. On each block there is a column of boxes of size $s \times s \times s'$, with s' denoting the height of a single box. Named G the total number of gray levels in I, hence s' is defined by the relationship $G/s' = M/s$[7].

Let numbers $1, 2, 3...$ be assigned to the boxes so to group the gray levels. Let the minimum and the maximum gray level of the image in the $(i, j)th$ grid fall in box number k and l, respectively.

The number of boxes covering this block is calculated as:

$$n_r(i, j) = l - k + 1 \tag{15}$$

In Figure 9 for example $s = s' = 3$, hence $n_r = 3 - 1 + 1$. Extending to the contribution from

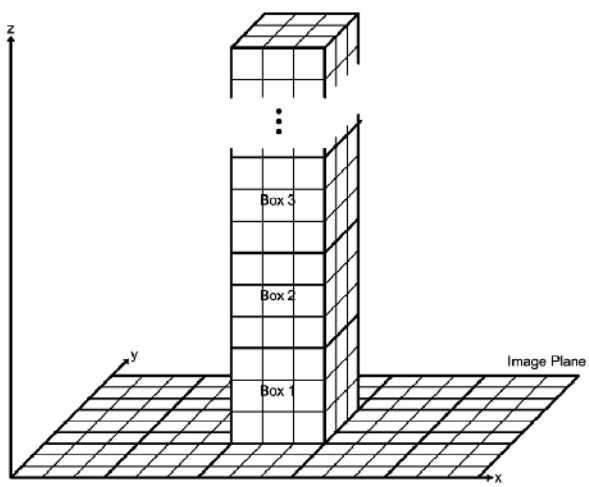


Figure 9. Example of DBC method application for determining the number of boxes of size $s \times s \times s$, when $s = 3$.

all blocks:

$$N_r = \sum_{i,j} n_r(i, j) \tag{16}$$

The Eq. 16 is computed for different box size s (so for different r) and the values of N_r are plotted versus the values of r in a log-log plot.

A matlab implementation of DBC can make use of functions such as *blockproc* or *colfilt* in order to make the box partitioning and apply the Eq. 15.

The DBC procedure has some weak points in the method used to select an appropriate box height[7], since the values of s is limited to the image size and s' is limited by the number of blocks of size $s \times s$ in which the image is divided.

Secondly, the box number calculation may lead to overestimate the number of boxes needed to cover the surface. Let A and B be the pixels associated with the minimum and the maximum gray level of the block respectively, as is illustrated in Figure 10.

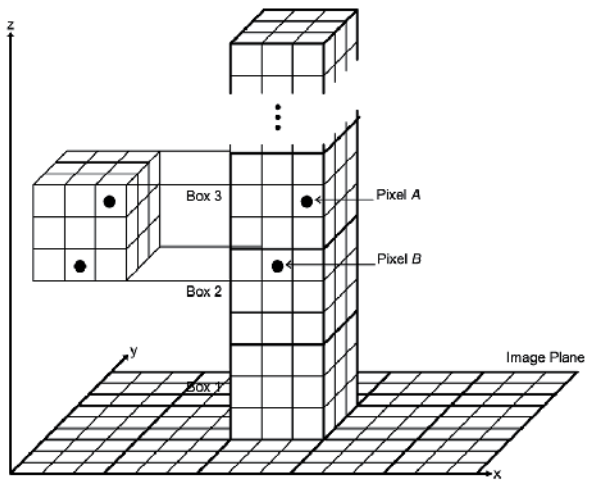


Figure 10. Example of DBC method application with boxes of $s \times s \times s$, when $s = 3$. The two pixels A and B, denoting the maximum and the minimum gray levels of the block, are assigned in two differents boxes, having distance in eight direction smaller than the box size $s = 3$.

According to DBC procedure, the two pixels are assigned in boxes 2 and boxes 3. The distance between A and B is smaller than 3, which is the size of the box.

Hence, when calculating Eq. 15, the block can be covered by a single box but its pixels with minimum and maximum gray levels fall into two different boxes.

To solve the aforementioned problems some modifications was proposed by J. Li, Q. Du and C. Sun[7]. Given a digital image I of size $M \times M$, a new scale r is defined instead of r, i.e. $r' = r/c$ where $c1$ is a positive real number.

In particular, let μ and σ be the mean and the standard deviation of I respectively. Hence, if the greater part of image pixels fall into the interval of gray level within $[\mu - a\sigma, \mu + a\sigma]$, where a is a positive integer, the height of the boxes is given by:

$$r' = \frac{r}{1 + 2a\sigma} \tag{17}$$

If dz is the height of the boxes in the direction of z, the number of the column of boxes on a single image block correspond to the integer part of $(dz/r' + 1)$ instead of $(dz/r + 1)$ as in the original DBC method. Thus, since $r' < r$, the residual part of dz/r' is smaller than that of dz/r.

As a result, the errors introduced using r' are smaller than in the original DBC method. A box with smaller height is chosen when a higher intensity variation is present on the image surface. So the improved method uses, in general, finer scales to count[7].

Moreover, the use of dz instead of z to count the number of boxes leads to the following modification of Eq. 15:

$$n_r = \begin{cases} ceil(\frac{l-k}{r'}), & \\ 1, & l = k \end{cases} \tag{18}$$

with ceil(.) denoting the function rounds the elements of the quantity into (.) to the nearest integers greater or equal to it.

Eq. 18 relies a new way to count the number of boxes that cover the $(i, j)th$ block surface in which the boxes are assigned to the minimum gray level to the block rather than gray level 0[7].

As an example, suppose that the $(i, j)th$ block is covered by a column boxes with the size $3x3x3$. If the pixels A and B represent the maximum and the minimum gray levels of the block, the two pixels will be assigned as in Figure 10.

According to Eq. 18 the number of counted boxes is $n_r = 1$, which is exactly the number of boxes covering the block.

As in standard box counting method, after having determined the number nr(i,j) for each block, the total number of boxes N_r covering the full image surface is computed for different scales r. Plotting the linear fit of log N_r versus the log r (Richardson's plot) the fd is finally estimated.

5. Applications and discussion

Each described method has been implemented in Matlab 2010a and applied to either well-known fractals or biomedical images.

The results on the hand and dividers methods are shown in the table 1. The computed values are also compared to the theoretical fd values. The computational time for a 2.50 GHz 5i CPU is also shown.

The value ranges for the step size are not displayed but they were automatically chosen based upon the computation of the structure's maximum caliber diameter which is defined as the major axis of an ellipse in which the structure can be embedded. The range was then running from the 40% of the maximum caliber diameter to the minimum step defined as the maximum distance between any two contiguous border points.

In practice, both EXACT and HYBRID methods computed the different step sizes by scaling each time the maximum step by a^k with k the number of the iteration. The chosen value of $a = 1.2$ is a compromise between a sufficient number of fitting points and the need to avoid too small variations of the step size so to duplicate perimeter estimation. The latter usually occurs in HYBRID method for it hits the same current points if the step does not vary enough in two consecutive iterations.

The parameter's estimation uncertainty is also shown in the table 1; that is calculated from the fitting accuracy based upon standard linear regression.

The number of data points used in the Richardson's plot was about 60 and two examples of that computation using EXACT and HYBRID are shown in Figure 12.

On the table 2 the computation results for the box counting method are also shown. The type of the displayed values are similar to the previous ones with the exception of Box counting uncertainty. In fact, the way an image can be partitioned into several boxes may affect the final computation of the number of nonempty boxes.

To investigate the variability of the fd for different box partitioning layouts, random box subdivisions have been applied. Therefore, the results on the table 2 show the standard deviation of the different computed fds and the mean values for each fractal at issue. In general, that variability is more pronounced in images having rougher resolution.

Fractal	fd_{theo}	fd_{exp}	*Time* (sec)	*BC error*	*Image size*
Apollonian Gasket	1.3057	1.408	1.5	0.001	2000×2000
Sierpinski	1.5849	1.587	0.3	0.005	1000×1000
Dragon	2.0000	1.747	7.2	0.006	3670×3978
Hexaflake	1.7719	1.640	1.6	0.011	1050×1050

Table 1. Tabular of results for box counting method application.

Fractal	fd_{theo}	fd_{exp}	*Time* (sec)	*BC error*	*Vector size*
Twin Dragon Hybrid	1.5236	1.466	8.6	0.006	117005
Twin Dragon Exact	1.5236	1.465	11.5	0.006	117005
Dragon Hybrid	1.5236	1.474	11.1	0.005	115665
Dragon Exact	1.5236	1.462	12.8	0.004	115665
Koch Hybrid	1.2619	1.276	31.2	0.004	786433
Koch Exact	1.2619	1.260	154.9	0.003	786433
Gosper Hybrid	1.1292	1.133	3.8	0.001	23280
Gosper Exact	1.1292	1.128	4.7	0.001	23280

Table 2. Tabular of results for walking-based methods application.

In general, the EXACT and the HYBRID methods appeared to be more precise than the box counting method but on the other hand they have a less wide range of applicability. However, this is also the reason of the fortune of the box counting methods compared to the others. Also, HYBRID technique is computationally less expensive than EXACT especially when the number of border points is quite large. The use of a variable step length which can be shorter or longer than the fixed step size leads to a larger variability and so to a Richardoson's plot having a less accurate fitting. That has effects on the uncertainties of the parameter to estimate. Because of that, a more careful choice of the step size range is needed in the case of HYBRID method.

Importantly, it is quite clear that the choice of the starting point may also affect the perimeter value as the following currents points will depend upon this. A test on 80 random starting points for the Gosper Island fractal revealed that the fd computation performed with the HYBRID method appeared to be more stable than the one with EXACT.

As for walking method, in box counting the process of scaling from the maximum box size is limited by the pixel size so in principle a gross resolution might be the reason of a bad estimate of fd. It is noteworthy that the tests performed do not show any correlation between resolution and fd accuracy; that may be also caused by the fact that some fractals such as dragon does not reproduce the real fractal at small scales.

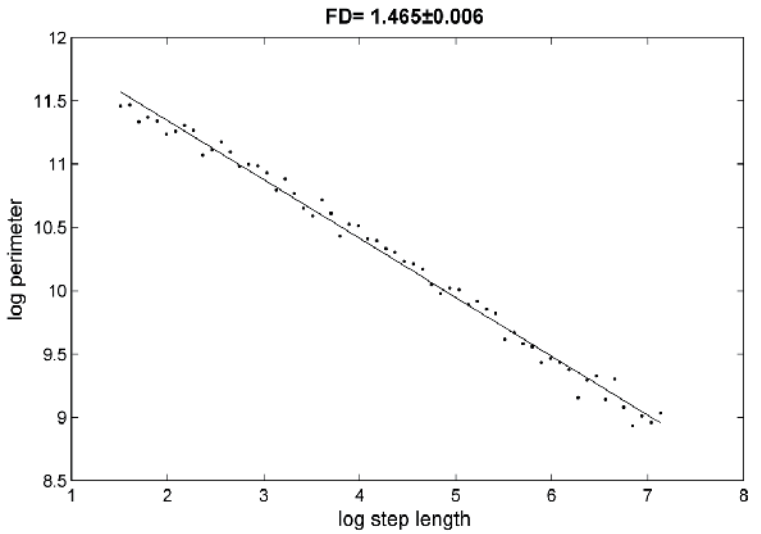


Figure 11. EXACT method apllied to the twin dragon fractal: Richardson's Plot.

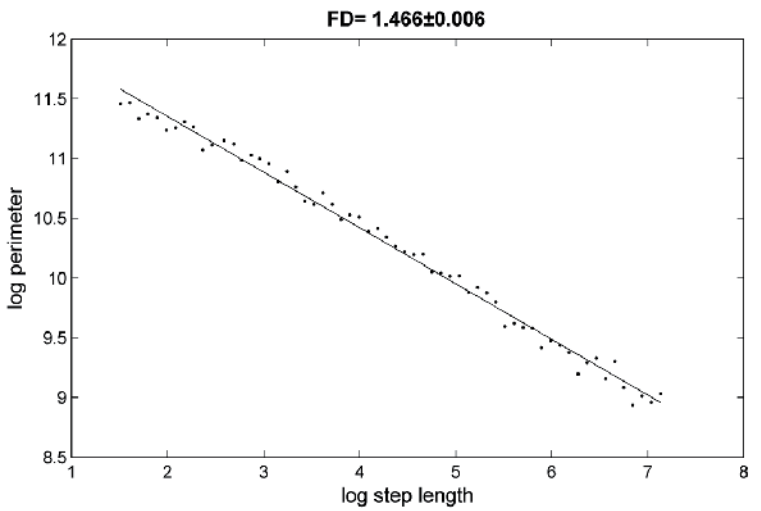


Figure 12. HYBRID method apllied to the twin dragon fractal: Richardson's Plot.

An application of the DBC method on a x-ray image is also shown in Figure 13 where breast cancer mammography image has been processed. The method uses a sliding technique as implemented in *blockproc* or *colfilt* matlab functions so to produce an image rather than a single fd value as previously described.

The second DBC method shows higher contrast in the area of the cancer and consequently lower fd values. Due to the enormous amount of linear fitting performed for an image size of 3450×3100 the computational time reached 15 minutes.

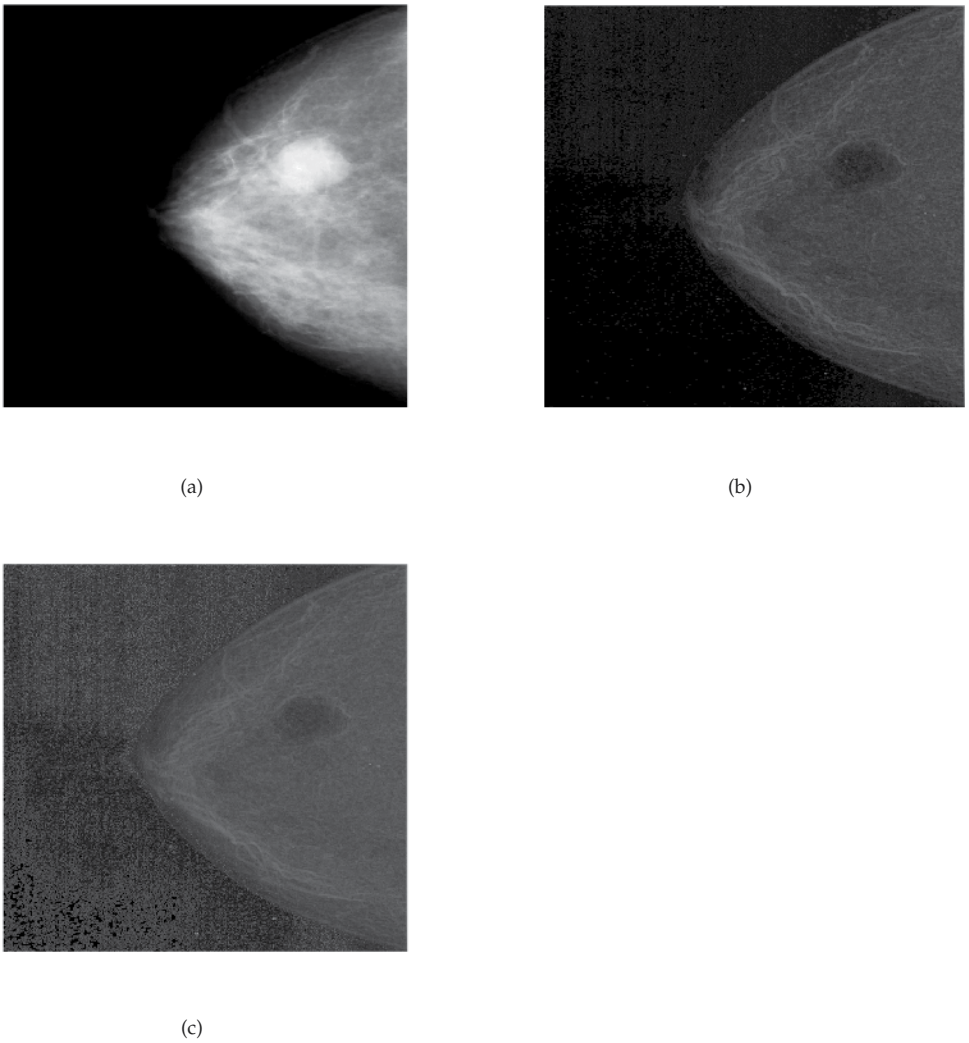


Figure 13. High resolution mammography image (a); fd recostruction image by standard Differential Box Counting (DBC) (b); fd reconstruction image by modified DBC (c).

6. Conclusions

In this chapter some of the most widely used and robust methods for fractal dimension estimation as well as their performances have been described. For few of them a detailed description of the algorithm has been also reported to make much easier for a beginner to start and implement his own Matlab code. Computational time is not excessively long to necessitate compiled functions such as C-mex files but that can be an advantage when using very high resolution images. The use of the described algorithms is obviously not restricted to the sole field of the image processing but it can be applied with some changes to any data analysis.

Author details

Antonio Napolitano, Sara Ungania and Vittorio Cannata
Department of Occupational Health and Safety, Medical Physics, Bambino Gesù Children's Hospital, Rome, Italy

7. References

[1] J. Orford and W. Whalley, *The use of the fractal dimension to quantify the morphology of irregular-shaped particles*, [Sedimentology, 30, 655-668], (1983).
[2] J. Keller et al., *Texture description and segmentation through fractal geometry*, [Computer Vision, Graphics and Image Processing, 45, 150-166], (1989).
[3] F. Hausdorff, *Dimension und ausseres Mass*, [Math. Annalen 79, 157] (1919).
[4] J. Theiler, *Estimating fractal dimension*, [7, 6/June 1990/J. Opt. Soc. Am. A] (1989).
[5] B. Mandelbrot, *The Fractal Geometry of Nature*, W.H.Freeman and Company, New York, (1983).
[6] S. Deepa,T. Tessamma *Fractal Features based on Differential Box Counting Method for the Categoritazion of Digital Mammograms*, [International Journal of Computer Information System and Industrial Management Applications, 2, 011-019], (2010).
[7] J. Li, Q. Du, *An improved box-counting method for image fractal dimension estimation*, [Pattern Recognition, 42, 2460-2469], (2009).
[8] N. Sarker, B. B. Chaudhuri, *An efficient differential box-counting approach to compute fractal dimension of image*, [IEEE Transaction on Systems, Man, and Cybernetics, 24, 115-120], (1994).
[9] A. P. Pentland, *Fractal-based description of natural scenes*, [IEEE Transaction on Pattern Analysis and Machine Intelligence, 6, 661-674], (1984).
[10] L. F. Richardson, *Fractal growth phenomena*, [Ann. Arbor, Mich. : The Society 6, 139] (1961).
[11] B. Mandelbrot, *How long is the coast of Britain? Statistical self-similarity and fractional dimension*, [Science 155, 636-638] (1967).
[12] N. Clark, *Three techiques for implementing digital fractal analysis of particle shape*, [Powder Technology 46, 45] (1986).
[13] M. Allen, G. J. Brown, N. J. Miles, *Measurement of boundary fractal dimensions: review of current techinques*, University of Nottingham , UK (1994).
[14] M. Allen, *Ph.D. Thesis*, University of Nottingham , UK (1994).
[15] P. Podsiadlo, G. W. Stachowiak, *Evaluation of boundary fractal methods for the characterization of wear particles*, [Wear 217(1) , 24-34] (1998).
[16] J. D. Farmer, E. Ott. and J. A. Yorke, *The dimension of chaotic attractors*, [Physica 7D, 153] (1983).
[17] B. Kaye, *A Random Walk Through Fractal Dimension*, [VCH Verlagsgesellschaft] (1986)
[18] L. Niemeyer, L. Pietronero and H. J. Wiesmann, *Fractals dimension of dielectric breakdown*, [Phys. Rev. Lett. 52, 1033] (1984).
[19] H. Schwarz and E. Exner, *The implementation of the concept of fractal dimension on semi-automatic image analyzer*, [Powder Technology, 27, 207-213], (1980).
[20] J. Russ, *Fractals surfaces*, [Plenum Press] (1994).
[21] H. von Koch, *Sur une courbe continue sans tangente, obtenue par une construction geometrique elementaire*, Archiv for Matemat., [Astron. och Fys. 1, 681-702] (1904).

Chapter 8

MATLAB Aided Option Replication

Vasilios N. Katsikis

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/45723

1. Introduction

In reality markets are incomplete in the sense that perfect replication of contingent claims using only the underlying asset and a riskless bond is impossible. In other words, that is perfect risk transfer is not possible since some payoffs cannot be replicated by trading in marketed securities. From the work of Ross in [21], it is evident that whenever the payoff of every call or put option can be replicated then the securities market is complete. In addition, an important implication of the aforementioned work of Ross, is the existence of options that cannot be replicated by the primitive securities when markets are incomplete. In [7], the authors came to the conclusion that Ross's result is, in fact, a negative result since it asserts that in an incomplete market one cannot expect to replicate the payoff of each option even if the underlying asset is traded. In the same paper, it is proved that if the number of securities is less than half the number of states of the world, then (generically) not a single option can be replicated by traded securities. In [10], the author extended the aforementioned result in [7], to accommodate cases where the condition on the number of primitive securities is not imposed. In particular, it is proved that if there exists no binary payoff vector in the asset span, then for each portfolio there exists a set of nontrivial exercise prices of full measure such that any option on the portfolio with an exercise price in this set is non-replicated. Furthermore, note that the results of Ross for two-date security markets with finitely many states holds for security markets with more than two dates, see [8, 9].

It is well accepted that the lattice theoretic ideas are the most important technical contributions of the large literature on infinite-dimensional modern mathematical finance (for example lattice theoretic ideas in general equilibrium theory). However, ordered vector spaces that are not lattice ordered arise naturally in models of portfolio trading. Moreover, if available securities have smooth payoffs, then the portfolio space is never a vector lattice. It should be pointed out that since call and put options are vector lattice operations in the space of contingent claims, their replication by available securities requires a vector lattice structure in the portfolio space. There is a large literature on vector lattice theory related to mathematical economics; see for instance [1–7, 17–20, 22, 23].

On the other hand, there is an obvious need for properly structured high performance computational methods on vector lattices. Moreover, the main concern is to describe, in

computational terms, and then solve problems arising from mathematical economics such as portfolio insurance and option replication. A lot of work in this area has been done in [11, 12, 14**?** –16].

In this chapter, we focus to the option replication problem. We consider an incomplete market of primitive securities, meaning that some call and put options need not be marketed and our objective is to describe an efficient method for computing maximal submarkets that replicate any option. Even though, there are several important results on option replication they cannot provide a method for the determination of the replicated options. By using the theory of vector-lattices and positive bases it is provided a procedure in order to determine the set of securities with replicated options. In particular, it is shown that the union of all maximal replicated submarkets (i.e., submarkets Y, such that any option written on the elements of Y can be replicated and Y is as large as possible) defines a set of elements such that any option written on these elements is replicated.

In [11–14, 16], it was shown that it is possible to construct computational methods in order to efficiently compute vector sublattices and lattice-subspaces of $\mathbb{R}^m$ as well as in the general case of $C[a,b]$. In addition, these methods has been successfully applied in portfolio insurance and in completion of security markets.

Here we consider a two-period security market X with a finite number m of states and a finite number of primitive securities with payoffs in $\mathbb{R}^m$ and we construct computational methods in order to determine maximal replicated submarkets of X by using the theory of vector sublattices and lattice-subspaces. Moreover, in the theory of security markets it is a usual practice to take call and put options with respect to the riskless bond $\mathbf{1} = (1, 1, ..., 1)$. Then, the completion $F_{\mathbf{1}}(X)$ of X by options is the subspace of $\mathbb{R}^m$ generated by all options written on the elements of $X \cup \{\mathbf{1}\}$. Since the payoff space is $\mathbb{R}^m$, which is a vector lattice, in the case where $\mathbf{1} \in X$ then $F_{\mathbf{1}}(X)$ is exactly the vector sublattice generated by X. If, in addition, X is a vector sublattice of $\mathbb{R}^m$ then $F_{\mathbf{1}}(X) = X$ therefore any option is replicated, unfortunately this situation is extremely rare.

A recent article, [15], provided a computationally efficient method for computing maximal submarkets that replicate any option. In particular, by using computational methods and techniques from [11–14] in order to determine vector sublattices and their positive bases, it is presented a procedure in order to calculate the set of securities with replicated options. The aforementioned article emphasizes the most important interrelationship between the theory of vector lattices, positive bases, projection bases and the problem of option replication.

The material in this chapter is spread out in 5 sections. Section 2 is divided in two subsections; in the first the fundamental properties of lattice-subspaces and vector sublattices are presented, whereas in the second we introduce the basic results for vector sublattices, positive bases and projection bases of $\mathbb{R}^k$ together with the solution to the problem of whether a finite collection of linearly independent, positive vectors of $\mathbb{R}^k$ generates a lattice-subspace or a vector sublattice. In section 3, there are three subsections where it is discussed the theoretical background for option replication. Also, section 3 emphasis the most important interrelationship between positive bases, projection bases and the problem of option replication. Section 4 presents an algorithm for calculating maximal submarkets that replicate any option followed by the corresponding computational approach. Conclusions and research directions are provided in Section 5.

In this chapter, all the numerical tasks have been performed using the Matlab 7.8 (R2009a) environment on an Intel(R) Pentium(R) Dual CPU T2310 @ 1.46 GHz 1.47 GHz 32-bit system with 2 GB of RAM memory running on the Windows Vista Home Premium Operating System.

2. Basic results in the theory of positive bases and projection bases of $\mathbb{R}^m$

In this section, a brief introduction is provided to the theory of vector lattices of $\mathbb{R}^m$. Furthermore, we present some basic results related to the theory of positive bases and projection bases of $\mathbb{R}^m$.

2.1. Preliminaries and notation

Initially, we recall some definitions and notation from the vector lattice theory. Let $\mathbb{R}^m = \{x = (x(1), x(2), ..., x(m)) | x(i) \in \mathbb{R}, \textit{for each } i\}$, where we view $\mathbb{R}^m$ as an ordered space. The *pointwise order* relation in $\mathbb{R}^m$ is defined by

$$x \leq y \textit{ if and only if } x(i) \leq y(i), \textit{ for each } i = 1, ..., m.$$

The positive cone of $\mathbb{R}^m$ is defined by $\mathbb{R}^m_+ = \{x \in \mathbb{R}^m | x(i) \geq 0, \textit{for each } i\}$ and if we suppose that X is a vector subspace of $\mathbb{R}^m$ then X ordered by the pointwise ordering is an *ordered subspace* of $\mathbb{R}^m$, with positive cone $X_+ = X \cap \mathbb{R}^m_+$. By $\{e_1, e_2, ..., e_m\}$ we shall denote the usual basis of $\mathbb{R}^m$. A point $x \in \mathbb{R}^m$ is an *upper bound, (resp. lower bound)* of a subset $S \subseteq \mathbb{R}^m$ if and only if $y \leq x$ (resp. $x \leq y$), for all $y \in S$. For a two-point set $S = \{x, y\}$, we denote by $x \vee y$ (resp. $x \wedge y$) the *supremum* of S i.e., its least upper bound (resp. the *infimum* of S i.e., its greatest lower bound). Thus, $x \vee y$ (resp. $x \wedge y$) is the componentwise maximum (resp. minimum) of x and y defined by

$$(x \vee y)(i) = \max\{x(i), y(i)\} \; ((x \wedge y)(i) = \min\{x(i), y(i)\}), \; \textit{for all } i = 1, ..., m.$$

For any $x = (x(1), x(2), ..., x(m)) \in \mathbb{R}^m$, the set $supp(x) = \{i | x(i) \neq 0\}$ is the *support* of x. The vectors $x, y \in \mathbb{R}^m$ have *disjoint supports* if $supp(x) \cap supp(y) = \emptyset$.

An ordered subspace Z of $\mathbb{R}^m$ is a *vector sublattice* or a *Riesz subspace* of $\mathbb{R}^m$ if for any $x, y \in Z$ the supremum and the infimum of the set $\{x, y\}$ in $\mathbb{R}^m$ belong to Z.

Assume that X is an ordered subspace of $\mathbb{R}^m$ and $B = \{b_1, b_2, ..., b_\mu\}$ is a basis for X. Then B is a *positive basis* of X if for each $x \in X$ it holds: x is positive if and only if its coefficients in the basis B are positive. In other words, B is a positive basis of X if the positive cone X_+ of X has the form,

$$X_+ = \{x = \sum_{i=1}^{\mu} \lambda_i b_i | \lambda_i \geq 0, \textit{for each } i\}.$$

Then, for any $x = \sum_{i=1}^{\mu} \lambda_i b_i$ and $y = \sum_{i=1}^{\mu} \varrho_i b_i$ we have $x \leq y$ if and only if $\lambda_i \leq \varrho_i$ for each $i = 1, 2, ..., \mu$. A positive basis $B = \{b_1, b_2, ..., b_\mu\}$ is a *partition of the unit* if the vectors b_i have disjoint supports and $\sum_{i=1}^{\mu} b_i = 1$.

Recall that a nonzero element x_0 of X_+ is an extremal point of X_+ if, for any $x \in X, 0 \leq x \leq x_0$ implies $x = \lambda x_0$, for a real number λ. Since each element b_i of the positive basis of X is an extremal point of X_+, a positive basis of X is unique in the sense of positive multiples.

The existence of positive bases is not always ensured, but in the case where X is a vector sublattice of $\mathbb{R}^m$ then X always has a positive basis. If $B = \{b_1, b_2, ..., b_\mu\}$ is a positive basis for a vector sublattice X the lattice operations in X, namely $x \vee y$ for the supremum and $x \wedge y$ for the infimum of the set $\{x, y\}$ in X, are given by

$$x \vee y = \sum_{i=1}^{\mu} \max\{\lambda_i, \varrho_i\} b_i \textit{ and } x \wedge y = \sum_{i=1}^{\mu} \min\{\lambda_i, \varrho_i\} b_i,$$

for each $x = \sum_{i=1}^{\mu} \lambda_i b_i, y = \sum_{i=1}^{\mu} \varrho_i b_i \in X$.

Suppose that L is a finite dimensional subspace of $C(\Omega)$ generated by a set $\{z_1, z_2, ..., z_r\}$ of linearly independent positive vectors of $C(\Omega)$. If Z is the sublattice of $C(\Omega)$ generated by L and $\{b_1, ..., b_\mu\}$ is a positive basis for Z ($\mu = dim(Z)$) then, a *projection basis* $\{\tilde{b_1}, \tilde{b_2}, ..., \tilde{b_r}\}$ of Z is a basis for L such that its elements are projections of the elements of the positive basis $\{b_1, ..., b_\mu\}$. In our current work we consider that $\Omega = \{1, 2, ..., m\}$ hence $C(\Omega) = \mathbb{R}^m$.

For an extensive presentation of vector sublattices as well as for notation not defined here we refer to [11–13, 15**?** , 16] and the references therein.

2.2. Theoretical background

In this section we present theoretical results for positive bases and projection bases in $\mathbb{R}^m$. Given a collection $x_1, x_2, ..., x_n$ of linearly independent, positive vectors of $\mathbb{R}^m$ we define the function,

$$h : \{1, 2, ..., m\} \to \mathbb{R}^n \textit{ such that } h(i) = (x_1(i), x_2(i), ..., x_n(i))$$

and the function,

$$\beta : \{1, 2, ..., m\} \to \mathbb{R}^n \textit{ such that } \beta(i) = \frac{h(i)}{\|h(i)\|_1} \tag{1}$$

for each $i \in \{1, 2, ..., m\}$ with $\|h(i)\|_1 \neq 0$. We shall refer to β as the *basic function* of the vectors $x_1, x_2, ..., x_n$. The set

$$R(\beta) = \{\beta(i) | i = 1, 2, ..., m, \textit{ with } \|h(i)\|_1 \neq 0\},$$

is the *range* of the basic function and the *cardinal number*, $cardR(\beta)$, of $R(\beta)$ is the number of different elements of $R(\beta)$.

Suppose that Z denotes the sublattice of $\mathbb{R}^m$ generated by $X = [x_1, x_2, ..., x_n]$. We shall denote by $P_1, P_2, ..., P_n, P_{n+1}, ..., P_\mu$ an enumeration of $R(\beta)$ such that the first n vertices $P_1, P_2, ..., P_n$ are linearly independent and $\mu = \dim(Z)$. Note that such an enumeration always exists. The notation, A^T stands for the transpose of a matrix A. A procedure in order to construct the sublattice Z is given by the following theorem.

Theorem 1. *Suppose that the above assumptions are satisfied. Then,*

(*i*) *X is a vector sublattice of $\mathbb{R}^m$ if and only if $R(\beta)$ has exactly n points (i.e., $\mu = n$). Then a positive basis $b_1, b_2, ..., b_n$ for X is defined by the formula*

$$(b_1, b_2, ..., b_n)^T = A^{-1}(x_1, x_2, ..., x_n)^T,$$

where A is the $n \times n$ matrix whose ith column is the vector P_i, for each $i = 1, 2, ..., n$. It is clear that in such a case Z and X coincide.

(ii) Let $\mu > n$. If $I_s = \beta^{-1}(P_s)$, and

$$x_s = \sum_{i \in I_s} \|h(i)\|_1 e_i, \ s = n+1, n+2, ..., \mu,$$

then

$$Z = [x_1, x_2, ..., x_n, x_{n+1}, ..., x_\mu],$$

is the vector sublattice generated by $x_1, x_2, ..., x_n$ and $\dim Z = \mu$.
For a positive basis $\{b_1, b_2, ..., b_\mu\}$ of Z, consider the basic function γ of $\{x_1, x_2, ..., x_\mu\}$ with range, $R(\gamma) = \{P_1', P_2', ..., P_\mu'\}$. Then, the relation

$$(b_1, b_2, ..., b_\mu)^T = B^{-1}(x_1, x_2, ..., x_\mu)^T \tag{2}$$

where B is the $\mu \times \mu$ matrix with columns the vectors $P_1', P_2', ..., P_\mu'$, defines a positive basis for Z.

The notion of the projection basis is important for our study. Furthermore, in the following, we are interested for a projection basis that corresponds to a positive basis. Let $\{z_1, z_2, ..., z_r\}$ be a set of linearly independent and positive vectors of $\mathbb{R}^m$ then by using Theorem 1 we construct the sublattice Z of $\mathbb{R}^m$ generated by these vectors. If $\dim(Z) = \mu$, by Theorem 1, a positive basis $\{b_1, b_2, ..., b_\mu\}$ of Z can be determined. The basic result for calculating the projection basis that corresponds to the positive basis $\{b_1, b_2, ..., b_\mu\}$ of Z is the following theorem.

Theorem 2. *Suppose that β is the basic function of the vectors $\{z_1, z_2, ..., z_r\}$ and $P_1, P_2, ..., P_r, P_{r+1}, ..., P_\mu$ is an enumeration of the range of β such that the first r vectors $P_1, P_2, ..., P_r$ are linearly independent and suppose also that $z_{r+1}, ..., z_\mu$ are the new vectors constructed in Theorem 1. If $L = [z_1, z_2, ..., z_r]$ is the subspace of $\mathbb{R}^m$ generated by the vectors $z_1, z_2, ..., z_r$ then,*

(i) $Z = L \oplus [z_{r+1}, ..., z_\mu]$,

(ii) $\{b_{r+1}, b_{r+2}, ..., b_\mu\} = \{2z_{r+1}, 2z_{r+2}, ..., 2z_\mu\}$,

(iii) If $b_i = \widetilde{b_i} + b_i'$, with $\widetilde{b_i} \in L$ and $b_i' \in [z_{r+1}, ..., z_\mu]$, for each $i = 1, 2, ..., r$, then $\{\widetilde{b_1}, \widetilde{b_2}, ..., \widetilde{b_r}\}$ is a basis for L which is given by the formula

$$(\widetilde{b_1}, \widetilde{b_2}, ..., \widetilde{b_r})^T = A^{-1}(z_1, z_2, ..., z_r)^T,$$

where A is the $r \times r$ matrix whose ith column is the vector P_i, for $i = 1, 2, ..., r$. This basis, $\{\widetilde{b_1}, \widetilde{b_2}, ..., \widetilde{b_r}\}$ is called the projection basis of L and has the property: The r first coordinates of any element $x \in L$ expressed in terms of the basis $\{b_1, b_2, ..., b_\mu\}$ coincide with the corresponding coordinates of x in the projection basis, i.e.,

$$x = \sum_{i=1}^{\mu} \lambda_i b_i \in L \Rightarrow x = \sum_{i=1}^{r} \lambda_i \widetilde{b_i}$$

Suppose that Z is the sublattice generated by a collection $z_1, z_2, ..., z_r$ of linearly independent, positive vectors of $\mathbb{R}^m$ and $\{d_1, d_2, ..., d_\mu\}$ is a positive basis for Z.

Then, by Theorem 2, if

$$(\widetilde{d_1}, \widetilde{d_2}, ..., \widetilde{d_r})^T = A^{-1}(z_1, z_2, ..., z_r)^T,$$

where A is the $r \times r$ matrix whose ith column is the vector P_i, for $i = 1, 2, ..., r$ then $\{\widetilde{d}_1, \widetilde{d}_2, ..., \widetilde{d}_r\}$ is the projection basis of $L = [z_1, z_2, ..., z_r]$. The projection basis $\{\widetilde{d}_1, \widetilde{d}_2, ..., \widetilde{d}_r\}$ is called the *projection basis of X corresponding to the basis* $\{d_1, d_2, ..., d_\mu\}$. The following proposition allows us to determine a positive basis and its corresponding projection basis. Moreover, the calculated positive basis is a partition of the unit.

Proposition 1. *Suppose that $\{d_i\}$ is the basis of Z given by equation (2) of Theorem 1 and $\{\widetilde{d}_i\}$ is the projection basis of $L = [z_1, z_2, ..., z_r]$ corresponding to the basis $\{d_i\}$. Then $\{b_i = \frac{d_i}{\|d_i\|_\infty} | i = 1, 2, ..., \mu\}$ is the positive basis of Z which is a partition of the unit and $\{\widetilde{b}_i = \frac{\widetilde{d}_i}{\|d_i\|_\infty} | i = 1, 2, ..., n\}$ is the projection basis of L corresponding to the basis $\{b_i\}$ of Z.*

In the following, we shall denote by $\mathbf{1}$ the vector $\mathbf{1} = (1, 1, ..., 1)$. A vector x is a *binary vector* if $x \neq \mathbf{0} = (0, 0, ..., 0)$, $x \neq \mathbf{1}$ and $x(i) = 0$ or $x(i) = 1$, for any i. Let $\{b_i | i = 1, 2, ..., \mu\}$ be a positive basis of Z which is a partition of the unit and let $\{\widetilde{b}_i | i = 1, 2, ..., n\}$ be the projection basis of L corresponding to the basis $\{b_i\}$. A partition $\delta = \{\sigma_i | i = 1, 2, ..., k\}$ of $\{1, 2, ..., n\}$ is *proper* if for any $r = 1, 2, ..., k$, the vector $w_r = \sum_{i \in \sigma_r} \widetilde{b}_i$ is a binary vector with $\sum_{r=1}^{k} w_r = \mathbf{1}$. If there is no proper partition of $\{1, 2, ..., n\}$ strictly finer than δ, then we say that δ is a *maximal proper partition* of $\{1, 2, ..., n\}$.

Example 1. Let $\{b_i | i = 1, 2, 3\}$ be a positive basis such that the corresponding projection basis is the following

$$\widetilde{b}_1 = (\frac{1}{2}, 1, 0, 1, 0),\ \widetilde{b}_2 = (\frac{1}{2}, 0, 0, 0, 1),\ \widetilde{b}_3 = (0, 0, 1, 0, 0).$$

We calculate the partitions of $\{1, 2, 3\}$ which are the following:

$$\delta_1 = \{\sigma_1, \sigma_2\},\ \textit{where}\ \sigma_1 = \{1\},\ \sigma_2 = \{2\ 3\}$$

$$\delta_2 = \{\sigma_1, \sigma_2\},\ \textit{where}\ \sigma_1 = \{2\},\ \sigma_2 = \{3\ 4\}$$

$$\delta_3 = \{\sigma_1, \sigma_2\},\ \textit{where}\ \sigma_1 = \{3\},\ \sigma_2 = \{1\ 2\}$$

$$\delta_4 = \{\sigma_1, \sigma_2, \sigma_3\},\ \textit{where}\ \sigma_1 = \{1\},\ \sigma_2 = \{2\},\ \sigma_3 = \{3\}.$$

Then δ_1 is not a proper partition since $w_1 = \sum_{i \in \sigma_1} \widetilde{b}_i = \widetilde{b}_1$ and $\widetilde{b}_1$ is not a binary vector. Similarly, δ_2 is not a proper partition. On the other hand δ_3 is a proper partition since $w_1 = \sum_{i \in \sigma_1} \widetilde{b}_i = \widetilde{b}_3$ and $\widetilde{b}_3$ is a binary vector, $w_2 = \sum_{i \in \sigma_2} \widetilde{b}_i = \widetilde{b}_1 + \widetilde{b}_2 = (1, 1, 0, 1, 1)$ which is a binary vector and $w_1 + w_2 = (1, 1, 1, 1, 1)$. Notice that δ_3 is strictly finer than δ_4, hence δ_3 is a maximal proper partition of $\{1, 2, 3\}$.

3. Option replication

In this section we shall discuss the economic model of our study. Moreover, first we discuss an inductive method for calculating the completion of security markets. So, if $\mathbf{1} \in X$ then it is possible to determine a basic set of marketed securities i.e., a set of linearly independent and positive vectors of X and the sublattice of $\mathbb{R}^m$ generated by a basic set of marketed securities is $F_{\mathbf{1}}(X)$. Finally, $F_{\mathbf{1}}(X)$ has a positive basis which is a partition of the unit. Under these observations we present an algorithmic procedure in order to determine maximal submarkets that replicate any option.

3.1. The economic model

In our economy there are two time periods, $t = 0, 1$, where $t = 0$ denotes the present and $t = 1$ denotes the future. We consider that at $t = 1$ we have a finite number of states indexed by $s = 1, 2, ..., m$, while at $t = 0$ the state is known to be $s = 0$.

Suppose that, agents trade $x_1, x_2, ..., x_n$ non-redundant securities in period $t = 0$, then the future payoffs of $x_1, x_2, ..., x_n$ are collected in a matrix

$$A = \left[x_i(j)\right]_{i=1,2,...n}^{j=1,2,...,m} \in \mathbb{R}^{m \times n}$$

where $x_i(j)$ is the payoff of one unit of security i in state j. In other words, A is the matrix whose columns are the non-redundant security vectors $x_1, x_2, ..., x_n$. It is clear that the matrix A is of full rank and the *asset span* is denoted by $X = Span(A)$. So, X is the vector subspace of $\mathbb{R}^m$ generated by the vectors x_i. That is, X consists of those income streams that can be generated by trading on the financial market. A *portfolio* is a column vector $\theta = (\theta_1, \theta_2, ..., \theta_n)^T$ of $\mathbb{R}^n$ and the *payoff* of a portfolio θ is the vector $x = A\theta \in \mathbb{R}^m$ which offers payoff $x(i)$ in state i, where $i = 1, ..., m$. A vector in $\mathbb{R}^m$, is said to be *marketed* or *replicated* if x is the payoff of some portfolio θ (called the *replicating portfolio* of x), or equivalently if $x \in X$. If $m = n$, then markets are said to be *complete* and the asset span coincides with the space $\mathbb{R}^m$. On the other hand, if $n < m$, the markets are *incomplete* and some state contingent claim cannot be replicated by a portfolio.

In the following, we shall denote by $\mathbf{1}$ the *riskless bond* i.e., the vector $\mathbf{1} = (1, 1, ..., 1)$. A vector x is a *binary vector* if $x \neq \mathbf{0} = (0, 0, ..., 0)$, $x \neq \mathbf{1}$ and $x(i) = 0$ or $x(i) = 1$, for any i. The *call option* written on the vector $x \in \mathbb{R}^m$ with exercise price α is the vector $c(x, a) = (x - \alpha\mathbf{1})^+ = (x - \alpha\mathbf{1}) \vee \mathbf{0}$. The *put option* written on the vector $x \in \mathbb{R}^m$ with exercise price α is the vector $p(x, a) = (\alpha\mathbf{1} - x)^+ = (\alpha\mathbf{1} - x) \vee \mathbf{0}$. If y is an element of a Riesz space then the following lattice identities hold, $y = y^+ - y^-$ and $y^- = (-y)^+$. It is clear that $x - \alpha\mathbf{1} = (x - \alpha\mathbf{1})^+ - (x - \alpha\mathbf{1})^- = (x - \alpha\mathbf{1})^+ - (\alpha\mathbf{1} - x)^+ = c(x, \alpha) - p(x, \alpha)$. Therefore we have the identity

$$x - a\mathbf{1} = c(x, a) - p(x, a),$$

which is called *put-call parity*.

If both $c(x, \alpha) > 0$ and $p(x, \alpha) > 0$, we say that the call option $c(x, \alpha)$ and the put option $p(x, \alpha)$ are *non trivial* and the exercise price α is a *non trivial exercise price* of x. If $c(x, \alpha)$ and $p(x, \alpha)$ belong to X then we say that $c(x, \alpha)$ and $p(x, \alpha)$ are *replicated*. If we suppose that $\mathbf{1} \in X$ and at least one of $c(x, a)$, $p(x, a)$ is replicated, then both of them are replicated since, $x - \alpha\mathbf{1} = c(x, \alpha) - p(x, \alpha)$.

3.2. Completion of security markets

We shall discuss the problem of completion by options of a two-period security market in which the space of marketed securities is a subspace of $\mathbb{R}^m$. The present study involves vector sublattices generated by a subset B of $\mathbb{R}^m$ of positive, linearly independent vectors. A computational solution to this problem is provided by using the `SUBlat` Matlab function from [16].

Let us assume that in the beginning of a time period there are n securities traded in a market. Let $\mathcal{S} = \{1, ..., m\}$ denote a finite set of states and $x_j \in \mathbb{R}^m_+$ be the payoff vector of security j in m states. The payoffs $x_1, x_2, ..., x_n$ are assumed linearly independent so that there are no redundant securities. If $\theta = (\theta_1, \theta_2, ..., \theta_n) \in \mathbb{R}^n$ is a non-zero portfolio then its payoff is the vector

$$T(\theta) = \sum_{i=1}^{n} \theta_i x_i.$$

The set of payoffs of all portfolios is referred as the space of *marketed securities* and it is the linear span of the payoffs vectors $x_1, x_2, ..., x_n$ in $\mathbb{R}^m$ which we shall denote it by X, i.e.,

$$X = [x_1, x_2, ...x_n].$$

For any $x, u \in \mathbb{R}^m$ and any real number a the vector $c_u(x, a) = (x - au)^+$ is the *call option* and $p_u(x, a) = (au - x)^+$ is the *put option* of x with respect to the *strike vector* u and *exercise price* a.

Let U be a fixed subspace of $\mathbb{R}^m$ which is called *strike subspace* and the elements of U are the *strike vectors*. Then, the *completion by options* of the subspace X with respect to U is the space $F_U(X)$ which is defined inductively as follows:

- X_1 is the subspace of $\mathbb{R}^m$ generated by $\mathcal{O}_1$, where $\mathcal{O}_1 = \{c_u(x, a) | x \in X, u \in U, a \in \mathbb{R}\}$, denotes the set of call options written on the elements of X,
- X_n is the subspace of $\mathbb{R}^m$ generated by $\mathcal{O}_n$, where $\mathcal{O}_n = \{c_u(x, a) | x \in X_{n-1}, u \in U, a \in \mathbb{R}\}$, denotes the set of call options written on the elements of X_{n-1},
- $F_U(X) = \cup_{n=1}^{\infty} X_n$.

The completion by options $F_U(X)$ of X with respect to U is the vector sublattice of $\mathbb{R}^m$ generated by the subspace $Y = X \cup U$. The details are presented in the next theorem,

Theorem 3. *In the above notation, we have*

(i) $Y \subseteq X_1$,

(ii) $F_U(X)$ *is the sublattice* $S(Y)$ *of* $\mathbb{R}^m$ *generated by* Y, *and*

(iii) *if* $U \subseteq X$, *then* $F_U(X)$ *is the sublattice of* $\mathbb{R}^m$ *generated by* X.

Any set $\{y_1, y_2, \ldots, y_r\}$ of linearly independent positive vectors of $\mathbb{R}^m$ such that $F_U(X)$ is the sublattice of $\mathbb{R}^m$ generated by $\{y_1, y_2, \ldots, y_r\}$ is a *basic set* of the market.

Theorem 4. *Any maximal subset* $\{y_1, y_2, \ldots, y_r\}$ *of linearly independent vectors of* $\mathcal{A}$ *is a basic set of the market, where* $\mathcal{A} = \{x_1^+, x_1^-, \ldots, x_n^+, x_n^-\}$, *if* $U \subseteq X$ *and* $\mathcal{A} = \{x_1^+, x_1^-, \ldots, x_n^+, x_n^-, u_1^+, u_1^-, \ldots, u_d^+, u_d^-\}$, *if* $U \subsetneq X$

The space of marketed securities X is *complete by options* with respect to U if $X = F_U(X)$.

From theorem 1 it follows,

Theorem 5. *The space* X *of marketed securities is complete by options with respect to* U *if and only if* $U \subseteq X$ *and* $card R(\beta) = n$. *In addition, the dimension of* $F_U(X)$ *is equal to the cardinal number of* $R(\beta)$. *Therefore,* $F_U(X) = \mathbb{R}^k$ *if and only if* $card R(\beta) = k$.

Example 2. Suppose that in a security market, the payoff space is $\mathbb{R}^{12}$ and the primitive securities are:

$$x_1 = (1,2,2,-1,1,-2,-1,-3,0,0,0,0)$$
$$x_2 = (0,2,0,0,1,2,0,3,-1,-1,-1,-2)$$
$$x_3 = (1,2,2,0,1,0,0,0,-1,-1,-1,-2)$$

and that the strike subspace is the vector subspace U generated by the vector

$$u = (1,2,2,1,1,2,1,3,-1,-1,-1,-2).$$

Then, a maximal subset of linearly independent vectors of $\{x_1^+, x_1^-, x_2^+, x_2^-, x_3^+, x_3^-, u_1^+, u_1^-\}$ can be calculated by using the following code:

```
>>XX = [max(X,zeros(size(X)));max(-X,zeros(size(X)))];
>>S = rref(XX');
>>[I,J] = find(S);
>>Linearindep = accumarray(I,J,[rank(XX),1],@min)';
>>W = XX(Linearindep,:);
```

where X denotes a matrix whose rows are the vectors x_1, x_2, x_3, u. We can determine the completion by options of X i.e., the space $F_U(X)$, with the `SUBlat` function from [16] by using the following code:

```
>>[VectorSublattice,Positivebasis]=SUBlat(W')
```

The results then are as follows

```
VectorSublattice =
1     2     2     0     1     0     0     0     0     0     0     0
0     2     0     0     1     2     0     3     0     0     0     0
1     2     2     1     1     2     1     3     0     0     0     0
0     0     0     0     0     0     0     0     1     1     1     2
2     0     4     0     0     0     0     0     0     0     0     0

Positivebasis =
0     0     0     0     0     0     0     0     1     1     1     2
0     0     0     1     0     0     1     0     0     0     0     0
0     0     0     0     0     4     0     6     0     0     0     0
4     0     8     0     0     0     0     0     0     0     0     0
0     6     0     0     3     0     0     0     0     0     0     0
```

3.3. Computation of maximal submarkets that replicate any option

We consider a two-period security market X with a finite number m of states and a finite number of primitive securities with payoffs in $\mathbb{R}^m$ and we construct computational methods in order to determine maximal submarkets of X that replicate any option by using the results provided in subsection 2.2. In particular, in the theory of security markets it is a usual practice

to take call and put options with respect to the riskless bond $\mathbf{1} = (1, 1, ..., 1)$. Then, the completion $F_{\mathbf{1}}(X)$ of X by options (see subsection 3.2) is the subspace of $\mathbb{R}^m$ generated by all options written on the elements of $X \cup \{\mathbf{1}\}$. Since the payoff space is $\mathbb{R}^m$, which is a vector lattice, in the case where $\mathbf{1} \in X$ then $F_{\mathbf{1}}(X)$ is exactly the vector sublattice generated by X. If, in addition, X is a vector sublattice of $\mathbb{R}^m$ then $F_{\mathbf{1}}(X) = X$ therefore any option is replicated.

A basic set of marketed securities (i.e., a set of linearly independent and positive vectors) of X always exist and the sublattice of $\mathbb{R}^m$ generated by a basic set of marketed securities is $F_{\mathbf{1}}(X)$. In addition, $F_{\mathbf{1}}(X)$ has a positive basis which is a partition of the unit.

Let us assume that X is generated by a basic set of marketed securities, then from Theorem 1 it is possible to determine a positive basis $\{b_1, b_2, ..., b_\mu\}$ of $F_{\mathbf{1}}(X)$.

The sublattice Z, generated by a basic set of marketed securities, is exactly $F_{\mathbf{1}}(X)$ and $F_{\mathbf{1}}(X)$ has a positive basis which is a partition of the unit, i.e., $\sum_{i=1}^{\mu} b_i = \mathbf{1}$. This is possible since the notion of a positive basis is unique in the sense of positive multiples therefore we are able to extract from the positive basis $\{b_1, b_2, ..., b_\mu\}$ another positive basis $\{d_1, d_2, ..., d_\mu\}$ of $F_{\mathbf{1}}(X)$ which is a partition of the unit. Therefore, let us denote by $\{d_1, d_2, ..., d_\mu\}$ a positive basis of $F_{\mathbf{1}}(X)$ which is a partition of the unit. Then, by Theorem 2, if

$$(\widetilde{d_1}, \widetilde{d_2}, ..., \widetilde{d_r})^T = A^{-1}(z_1, z_2, ..., z_r)^T,$$

where A is the $r \times r$ matrix whose ith column is the vector P_i, for $i = 1, 2, ..., r$ then $\{\widetilde{d_1}, \widetilde{d_2}, ..., \widetilde{d_r}\}$ is a projection basis of $F_{\mathbf{1}}(X)$. The projection basis $\{\widetilde{d_1}, \widetilde{d_2}, ..., \widetilde{d_r}\}$ is the projection basis of X corresponding to the basis $\{d_1, d_2, ..., d_\mu\}$. For $Z = F_{\mathbf{1}}(X)$ proposition 1 takes the following form.

Proposition 2. *Suppose that $\{d_i\}$ is the basis of $F_{\mathbf{1}}(X)$ given by equation (2) of theorem 1 and $\{\widetilde{d_i}\}$ is the projection basis of X corresponding to the basis $\{d_i\}$. Then $\{b_i = \frac{d_i}{\|d_i\|_\infty} | i = 1, 2, ..., \mu\}$ is the positive basis of $F_{\mathbf{1}}(X)$ which is a partition of the unit and $\{\widetilde{b_i} = \frac{\widetilde{d_i}}{\|d_i\|_\infty} | i = 1, 2, ..., n\}$ is the projection basis of X corresponding to the basis $\{b_i\}$ of $F_{\mathbf{1}}(X)$.*

Suppose that Y is a subspace of X, then if $F_{\mathbf{1}}(Y) \subseteq X$ we say that Y is *replicated*. If, in addition, for any subspace Z of X with $Y \subsetneqq Z$ we have that $X \subsetneqq F_{\mathbf{1}}(Z)$ then Y is a *maximal replicated subspace* or a *maximal replicated submarket* of X. Note that, the *replicated kernel of the market*, i.e., the union of all maximal replicated subspaces of the market is the set of all elements x of X so that any option written on x is replicated.

Let $\{b_i \, i = 1, 2, ..., \mu\}$ be a positive basis of $F_{\mathbf{1}}(X)$ which is a partition of the unit and let $\{\widetilde{b_i} \, i = 1, 2, ..., n\}$ be the projection basis of X corresponding to the basis $\{b_i\}$. Recall that, a partition $\delta = \{\sigma_i | i = 1, 2, ..., k\}$ of $\{1, 2, ..., n\}$ is *proper* if for any $r = 1, 2, ..., k$, the vector $w_r = \sum_{i \in \sigma_r} \widetilde{b_i}$ is a binary vector with $\sum_{r=1}^{k} w_r = \mathbf{1}$. If there is no proper partition of $\{1, 2, ..., n\}$ strictly finer than δ, then we say that δ is a *maximal proper partition* of $\{1, 2, ..., n\}$.

The following theorem provides the development of a method in order to determine the set of securities with replicated options by using the theory of positive bases and projection bases.

Theorem 6. *Let $\{b_i, i = 1, 2, ..., \mu\}$ be the positive basis of $F_{\mathbf{1}}(X)$ which is a partition of the unit and let $\{\widetilde{b_i}, i = 1, 2, ..., n\}$ be the projection basis of X corresponding to the basis $\{b_i\}$. If Y is a subspace of X, the following are equivalent:*

(i) Y is a maximal replicated subspace of X,

(ii) there exists a maximal proper partition $\delta = \{\sigma_i | i = 1, 2, ..., k\}$ of $\{1, 2, ..., n\}$ so that Y is the sublattice of $\mathbb{R}^m$ generated by δ.

The set of maximal replicated submarkets of X is nonempty.

4. The computational method

We present the proposed computational method that enables us to determine maximal submarkets that replicate any option. Our numerical method is based on the introduction of the `mrsubspace` function, from [15], that allow us to perform fast testing for a variety of dimensions and subspaces.

4.1. Algorithm for calculating maximal submarkets that replicate any option

Recall that X is the security market generated by a collection $\{x_1, x_2, ..., x_n\}$ of linearly independent vectors (not necessarily positive) of $\mathbb{R}^m$. If $\mathbf{1} \in X$ then it is possible to determine a basic set of marketed securities i.e., a set of linearly independent and positive vectors of X. This is possible through the following easy proposition:

Proposition 3. *If $a = \max\{\|x_i\|_\infty | i = 1, 2, ..., n\}$, then at least one of the two sets of positive vectors of X*

$$\{y_i = a\mathbf{1} - x_i | i = 1, 2, ..., n\}, \ \{z_i = 2a\mathbf{1} - x_i | i = 1, 2, ..., n\},$$

consists of linearly independent vectors.

The main steps of the underlying algorithmic procedure that enables us to determine maximal submarkets that replicate any option are the following:

(1) Use proposition 3 in order to determine a basic set $\{y_1, y_2, ..., y_n\}$ of marketed securities.

(2) Use Equation (1) in order to determine the basic curve β of the vectors y_i.

(3) Determine the range $R(\beta)$ of β.

(4) Use Theorem 1 in order to construct the vector sublattice generated by $y_1, y_2, ..., y_n$, which is exactly the completion by options $F_{\mathbf{1}}(X)$ of X. Then, determine a positive basis $\{d_1, d_2, ..., d_\mu\}$ for $F_{\mathbf{1}}(X)$.

(5) Use Theorem 2 in order to determine a projection basis $\{\widetilde{d_1}, \widetilde{d_2}, ..., \widetilde{d_n}\}$ of X.

(6) Use Proposition 1 in order to determine a positive basis $\{b_1, b_2, ..., b_\mu\}$ of $F_{\mathbf{1}}(X)$ which is a partition of the unit and the corresponding projection basis $\{\widetilde{b_1}, \widetilde{b_2}, ..., \widetilde{b_n}\}$.

(7) Calculate all the possible proper partitions of the set $\{1, 2, ..., n\}$.

(8) Decide which of the proper partitions created in step (7) are maximal proper partitions and determine the corresponding maximal replicated submarkets.

In [15], it is presented the translation followed by the implementation of the aforementioned algorithm in $\mathbb{R}^m$ within a Matlab-based function named `mrsubspace`. Moreover, in the same paper, computational experiments assessing the effectiveness of this function and lead us to the conclusion that the `mrsubspace` function provides an important tool in order to investigate replicated subspaces.

4.2. The computational approach - Code features

We shall discuss the proposed computational method that enables us to determine maximal submarkets that replicate any option. The standard method used currently to determine the maximal replicated submarkets is based on a manual processing. From section 3, it is evident that the required number of verifications for this process can be of significant size even in a relatively low-dimensional space, thus rendering the problem too difficult to solve. Our numerical method is based on the introduction of the `mrsubspace` function, from [15] that allow us to perform fast testing for a variety of dimensions and subspaces.

The structure of the code ensures flexibility, meaning that it is convenient for applications as well as for research and educational purposes. The given security market X, generated by the linearly independent vectors $x_1, x_2, ..., x_n$, must be given under a matrix notation with columns the vectors $x_1, x_2, ..., x_n$. The `mrsubspace` function must be stored in a Matlab-accessible directory and then the input data, i.e., the matrix X, can be typed directly in the Matlab's environment. Under the following command,

```
mrsubspace(X);
```

the program solves the problem of option replication and prints out the maximal proper partitions as well as the corresponding maximal replicated subspaces. If X is a vector sublattice, then $X = F_1(X)$ and any option is replicated. In the case where the initial space X is not a vector sublattice, it is possible to produce the normalized positive basis and the corresponding projection basis with the following code,

```
[Npb,Cprb] = mrsubspace(X)
```

Inside the code there are several explanations that indicate the implemented part of the algorithm. A user proficient in Matlab can easily use the code and modify it if needed. Especially, the user can isolate a part of the code according to his/her special needs to solve different problems like

- Determine a basic set of marketed securities.
- Find the completion $F_1(X)$ of X by options in $\mathbb{R}^m$ or find the vector sublattice generated by a finite collection of linearly independent vectors of $\mathbb{R}^m$.
- Calculate a positive basis and a projection basis for a finite dimensional vector sublattice.

In the last part of the code, entitled Maximal proper partitions - Maximal replicated subspaces, the user can change the way that the `mrsubspace` function understands the values 0 and 1, according to his/her knowledge and needs.

Example 3. Consider the following 5 vectors $x_1, x_2, ..., x_5$ in $\mathbb{R}^{10}$,

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0\,1\,0\,1\,1\,1\,1\,2\,2\,1 \\ 1\,1\,1\,2\,1\,1\,1\,2\,1\,2 \\ 1\,1\,1\,2\,1\,1\,1\,2\,1\,1 \\ 1\,1\,1\,1\,1\,1\,1\,2\,2\,1 \\ 2\,1\,2\,1\,1\,1\,1\,1\,1\,1 \end{bmatrix}$$

and $X = [x_1, x_2, ..., x_5]$ is the marketed space.

Note that $\mathbf{1} = x_5 - x_4 + x_1$. In order to determine the maximal replicated subspaces for the above collection of vectors we use the following simple code:

```
>> X = [0,1,0,1,1,1,1,2,2,1;1,1,1,2,1,1,1,2,1,2;...
1,1,1,2,1,1,1,2,1,1;1,1,1,1,1,1,1,2,2,1;2,1,2,1,1,1,1,1,1,1];
>> [Npb,Cprb] =mrsubspace(X)
```

as a result and after removing irrelevant Matlab output one gets

```
The 1 partition(s) are:
    {1} {2 3} {4} {5}

ReplicatedSubspace =

     1     0     1     0     0     0     0     0     0     0
     0     1     0     1     1     1     1     0     0     0
     0     0     0     0     0     0     0     1     1     0
     0     0     0     0     0     0     0     0     0     1

The 1 partition(s) are:
    {1} {2} {3 4} {5}

ReplicatedSubspace =

     1     0     1     0     0     0     0     0     0     0
     0     1     0     0     1     1     1     0     1     0
     0     0     0     1     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     0     1

Npb =

    1     0     1     0     0     0     0     0     0     0
    0     1     0     0     1     1     1     0     0     0
    0     0     0     1     0     0     0     0     0     0
    0     0     0     0     0     0     0     1     0     0
    0     0     0     0     0     0     0     0     0     1
    0     0     0     0     0     0     0     0     1     0
```

```
Cprb =

     1     0     1     0     0     0     0     0     0     0
     0     1     0     0     1     1     1     0     1     0
     0     0     0     1     0     0     0     0    -1     0
     0     0     0     0     0     0     0     1     1     0
     0     0     0     0     0     0     0     0     0     1
```

Therefore, the marketed space X has two maximal replicated subspaces, $\{1\}$ $\{2\ 3\}$ $\{4\}$ $\{5\}$ and $\{1\}$ $\{2\}$ $\{3\ 4\}$ $\{5\}$ are maximal proper partitions with corresponding maximal replicated subspaces the subspaces

$$Y_1 = [(1,0,1,0,0,0,0,0,0,0),(0,1,0,1,1,1,1,0,0,0),(0,0,0,0,0,0,0,1,1,0),\\ (0,0,0,0,0,0,0,0,0,1)]$$

and

$$Y_2 = [(1,0,1,0,0,0,0,0,0,0),(0,1,0,0,1,1,1,0,1,0),(0,0,0,1,0,0,0,1,0,0),\\ (0,0,0,0,0,0,0,0,0,1)],$$

respectively. The replicated kernel of the market is $Y = Y_1 \cup Y_2$.

5. Conclusions

In this chapter, computational methods for option replication are presented based on vector lattice theory. It is well accepted that the lattice theoretic ideas are one of the most important technical contributions of the large literature on modern mathematical finance. In this chapter, we consider an incomplete market of primitive securities, meaning that some call and put options need not be marketed. Our objective is to describe an efficient method for computing maximal submarkets that replicate any option. Even though, there are several important results on option replication they cannot provide a method for the determination of the replicated options. By using the theory of vector-lattices and positive bases it is provided a procedure in order to determine the set of securities with replicated options. Moreover, we determine those subspaces of the marketed subspace that replicate any option by introducing a Matlab function, namely `mrsubspace`. The results of this work can give us an important tool in order to study the interesting problem of option replication of a two-period security market in which the space of marketed securities is a subspace of $\mathbb{R}^m$. This work is based on a recent work, [15], regarding computational methods for option replication.

Author details

Vasilios N. Katsikis
General Department of Mathematics, Technological Education Institute of Piraeus, 12244 Athens, Greece

6. References

[1] Aliprantis, C.D. & Brown, D.J. (1983). Equilibria in markets with a Riesz space of commodities, *J. Math. Econom.*, 11, pp. 189–207.

[2] Aliprantis, C.D., Brown, D.J. & Burkinshaw, O. (1987a). Edgeworth equilibria, *Econometrica*, 55, pp. 1109–1137.

[3] Aliprantis, C.D., Brown, D.J. & Burkinshaw, O. (1987b). Edgeworth equilibria in production economies, *J. Econom. Theory*, 43, pp. 253–291.

[4] Aliprantis, C.D., Brown, D.J. & Burkinshaw, O. (1990). Existence and optimally of competitive equilibria, *Springer–Verlag*.

[5] Aliprantis, C.D. & Burkinshaw, O. (1991). When is the core equivalence theorem valid?, *Econom. Theory*, 1, pp. 169–182.

[6] Aliprantis, C.D., Tourky, R. & Yannelis, N. C. (2001). A theory of value with non–linear prices. Equilibrium analysis beyond vector lattices, *J. Econom. Theory*, 100, pp. 22–72.

[7] Aliprantis, C.D. & Tourky, R. (2002). Markets that don't replicate any option, *Economics Letter*, 76, pp.443–447.

[8] Baptista A.M. (2003). Spanning with american options, *Journal of Economic Theory*, 110, pp.264-289.

[9] Baptista A.M. (2005). Options and efficiency in multidate security markets, *Mathematical Fi- nance*, 15, No.4, pp.569-587.

[10] Baptista A.M. (2007). On the non-existence of redundant options, *Economic Theory*, 31, pp.205-212.

[11] Katsikis, V.N. (2007). Computational methods in portfolio insurance, *Applied Mathematics and Computation*, 189, pp.9-22.

[12] Katsikis, V.N. (2008). Computational methods in lattice-subspaces of $C[a,b]$ with applications in portfolio insurance, *Applied Mathematics and Computation*, 200, pp.204-219.

[13] Katsikis, V.N. (2009). A Matlab-based rapid method for computing lattice-subspaces and vector sublattices of $\mathbb{R}^n$: Applications in portfolio insurance, *Applied Mathematics and Computation*, 215, pp.961-972.

[14] Katsikis, V.N. (2010). Computational and Mathematical Methods in Portfolio Insurance. A MATLAB-Based Approach., Matlab - Modelling, Programming and Simulations, Emilson Pereira Leite(Ed.), ISBN: 978-953-307-125-1, InTech, Available from: http://www.intechopen.com/articles/show/title/computational-and-mathematical-methods-in-portfolio-insurance-a-matlab-based-approach-

[15] Katsikis, V.N. (2011). Computational methods for option replication, *International Journal of Computer Mathematics*, 88, No. 13, pp. 2752-2769.

[16] Katsikis V.N. & Polyrakis I. (2012). Computation of vector sublattices and minimal lattice-subspaces. Applications in finance, *Applied Mathematics and Computation*, 218, pp.6860–6873.

[17] Mas-Colell, A. (1986). The price equilibrium existence problem in topological vector lattices, *Econometrica*, 54, pp. 1039–1053.

[18] Mas-Colell, A. & Richard, S.F. (1991). A new approach to the existence of equilibrium in vector lattices, *J. Econom. Theory*, 53, pp. 1–11.

[19] Podczeck, K. (1996). Equilibria in vector lattices without ordered preferences or uniform properness, *J. Math. Econom.*, 25, pp. 465–484.

[20] Richard, S.F. (1989). A new approach to production equilibria in vector lattices, *J. Math. Econom.*, 18, pp. 231–247.
[21] Ross, S.A. (1976). Options and efficiency, *Quaterly Journal of Economics*, 90, pp.75-89.
[22] Tourky, R. (1998). A new approach to the limit theorem on the core of an economy in vector lattices, *J. Econom. Theory*, 78, pp. 321–328.
[23] Yannelis, N.C. & Zame, W.R. (1986). Equilibria in Banach lattices without ordered preferences, *J. Math. Econom.*, 15, pp. 85–110.

Chapter 9

Convolution Kernel for Fast CPU/GPU Computation of 2D/3D Isotropic Gradients on a Square/Cubic Lattice

Sébastien Leclaire, Maud El-Hachem and Marcelo Reggio

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46455

1. Introduction

The design of discrete operators or filters for the calculation of gradients is a classical topic in scientific computing. Typical applications are gradient reconstruction in computational fluid dynamics, edge detection in computer graphics and biomedical imaging, and phase boundary definition in the modeling of multiphase flows.

Edge detection, which is widely performed in image analysis, is an operation that requires gradient calculation. Commonly used edge detection methods are *Canny, Prewitt, Roberts* and *Sobel*, which can be found in MATLAB's platform. In this field, edge detection techniques rely on the application of convolution masks to provide a filter or kernel to calculate gradients in two perpendicular directions. A threshold is then applied to obtain an edge shape.

For multiphase flows, an edge or contour corresponds to the interface between the fluids. In this respect, traditional gradient calculation methods based on 1D edge detection are not necessarily suited for the underlying physics, because there is no direction in which the gradients of the phase contours tend to evolve over time. As a result, definition of the geometric progress of the interface requires many gradient estimation computations, as is the case in moving and deforming bubbles or droplets, for example. Although it can still be a viable tool, it is clear that the 1D-based method is becoming less useful for simulating these phenomena, which are not, in general, biased toward any particular direction.

To address this issue, we present an efficient computational method for obtaining discrete isotropic gradients that was previously applied to simulate two-phase flows within the lattice Boltzman framework [1, 2]. This "omnidirectional" approach makes it possible to improve the limitations inherent in handling high density ratios between the phases and to significantly reduce spurious currents at the interface. The method is based on a filter which is generally not split along any direction, and there is no need to make the assumption of a continuous filter to reach isotropy, as done by [3]. We also believe that optimal or maximal isotropy can

only be reached with a discrete filter when the error terms of Taylor's series expansion are isotropic, as explained in detail by [1, 2].

Below, we describe isotropic and anisotropic discretizations that will and will not conserve the isotropic property of the differentiated function respectively. This is followed by a description of how convolution can be used to reduce computer time in the gradient calculation. We then present details of the MATLAB implementation of these ideas, along with speedup comparisons of convolution performed on a single core of an Intel® Core i7-970 processor and on an Nvidia® GeForce GTX 580 GPU using the Jacket plugin for MATLAB developed by AccelerEyes®. The GPU with the Jacket plugin for MATLAB speeds up gradient computation by a factor of up to 138x in the more challenging case. Our study ends with an example of multiphase flow simulation conducted using the lattice Boltzmann method. Finally, we provide useful stencils and weights (or kernels) to yield isotropic gradients in two and three dimensions.

2. Gradient discretization

Let us define the real scalar function $F(x, y) \in \Re$. There are various methods for calculating the derivative of a function, and one way of doing so is to apply a finite difference approximation. If both the horizontal and vertical lattice spacings are h in length, a simple procedure for evaluating the gradient of this function is as follows:

$$\frac{\partial F}{\partial x} \approx \frac{1}{6h^2} \sum_i F(x + c_i^x, y + c_i^y) c_i^x \tag{1}$$

$$\frac{\partial F}{\partial y} \approx \frac{1}{6h^2} \sum_i F(x + c_i^x, y + c_i^y) c_i^y \tag{2}$$

with

$$c^x = [0, h, h, h, 0, -h, -h, -h] \tag{3}$$

$$c^y = [h, h, 0, -h, -h, -h, 0, h] \tag{4}$$

This finite difference discretization is very similar to Prewitt's operator/kernel [4], which is used in image processing. Note that $h = 1$ in this field, and that the application of Prewitt's operator results in a vector that points in the same direction as the finite difference gradient, but with a norm 6 times larger. The application of Prewitt's operator to an image can be computed very quickly using a convolution product. We address the topic of convolution product later in section (3), but first, let us analyze the isotropy property of the previous gradient discretization.

2.1. Anisotropic discretization

As in Ref. [2] and without loss of generality, the function F is expressed using a 2D Taylor series expansion around the zero vector:

$$\begin{aligned}F(x,y) =& F(0,0) + x\frac{\partial F}{\partial x}\bigg|_{(0,0)} + y\frac{\partial F}{\partial y}\bigg|_{(0,0)} + \frac{1}{2}x^2\frac{\partial^2 F}{\partial x^2}\bigg|_{(0,0)} + xy\frac{\partial^2 F}{\partial x \partial y}\bigg|_{(0,0)} \\ &+ \frac{1}{2}y^2\frac{\partial^2 F}{\partial y^2}\bigg|_{(0,0)} + \frac{1}{6}x^3\frac{\partial^3 F}{\partial x^3}\bigg|_{(0,0)} + \frac{1}{2}x^2 y\frac{\partial^3 F}{\partial x^2 \partial y}\bigg|_{(0,0)} \\ &+ \frac{1}{2}xy^2\frac{\partial^3 F}{\partial x \partial y^2}\bigg|_{(0,0)} + \frac{1}{6}y^3\frac{\partial^3 F}{\partial y^3}\bigg|_{(0,0)} + O(x^n y^m)\end{aligned}$$

with $n + m > 3$.

To calculate the gradient in the x direction, the following stencil values can be taken: $F(h,h)$, $F(h,0)$, $F(h,-h)$, $F(-h,-h)$, $F(-h,0)$, and $F(-h,h)$. When these stencil values are combined, as in the case of the gradient approximation in Eqs. (1) and (2), an exact expression for the gradient in the x direction at $(0,0)$ is obtained:

$$\begin{aligned}\frac{\partial F}{\partial x}\bigg|_{(0,0)} =& \frac{1}{6h}\Big(F(h,h) + F(h,0) + F(h,-h) - F(-h,-h) - F(-h,0) - F(-h,h)\Big) \\ &- \frac{h^2}{6}\frac{\partial^3 F}{\partial x^3}\bigg|_{(0,0)} - \frac{h^2}{3}\frac{\partial^3 F}{\partial x^2 \partial y}\bigg|_{(0,0)} + O(h^3)\end{aligned} \tag{5}$$

A similar expression is found for the y direction:

$$\begin{aligned}\frac{\partial F}{\partial y}\bigg|_{(0,0)} =& \frac{1}{6h}\Big(F(-h,h) + F(0,h) + F(h,h) - F(h,-h) - F(0,-h) - F(-h,-h)\Big) \\ &- \frac{h^2}{6}\frac{\partial^3 F}{\partial y^3}\bigg|_{(0,0)} - \frac{h^2}{3}\frac{\partial^3 F}{\partial x \partial y^2}\bigg|_{(0,0)} + O(h^3)\end{aligned} \tag{6}$$

In the gradients of Eqs. (5) and (6), the leading $O(h^2)$ differential operator of the error term can be written in vector form:

$$\vec{E}^{(2)} = -\frac{h^2}{6}\left[\frac{\partial^3}{\partial x^3} + 2\frac{\partial^3}{\partial x^2 \partial y}, \frac{\partial^3}{\partial y^3} + 2\frac{\partial^3}{\partial x \partial y^2}\right] \tag{7}$$

Using the following transformation from a Cartesian to a polar partial derivative operator:

$$\frac{\partial}{\partial x} = \cos(\theta)\frac{\partial}{\partial r} - \frac{1}{r}\sin(\theta)\frac{\partial}{\partial \theta} \tag{8}$$

$$\frac{\partial}{\partial y} = \sin(\theta)\frac{\partial}{\partial r} + \frac{1}{r}\cos(\theta)\frac{\partial}{\partial \theta} \tag{9}$$

And, by supposing that $F = F(r)$ is rotationally invariant, it is possible to show (after a lengthy algebraic manipulation) that the vector that results when the differential operator $\vec{E}^{(2)}$ is applied to $F(r)$ will have a Euclidean norm that is a function of θ and r, except if $F(r)$ is a constant. This result, that is, the norm $||\vec{E}^{(2)}F(r)|| \equiv f(r,\theta)$, is an equation that takes up almost a page, and so is not presented here. Let us note, however, that this expression can easily be obtained using symbolic mathematics software. For a vector function

to be rotationally invariant, it must have an Euclidean norm that is a function of the radius only. Therefore, this gradient approximation is not isotropic to the second order in space, but anisotropic. It is worth noting that the calculated derivatives have a leading error term that is not isotropic, even if the function F is isotropic, i.e. $F = F(r)$ around (0,0). This means that the discrete gradient will not conserve the isotropic property of the differentiated function when the gradient is approximated with this finite difference stencil.

2.2. Isotropic discretization

Taking into consideration the previous anisotropy problem, it is possible to change the weights of the grid points when computing the gradients to make them isotropic, up to the second order in space, by defining gradients in the x and y directions, as follows:

$$\begin{aligned}\left.\frac{\partial F}{\partial x}\right|_{(0,0)} =& \frac{1}{12h}\Big(F(h,h)+4F(h,0)+F(h,-h)-F(-h,-h)-4F(-h,0)-F(-h,h)\Big)\\ &-\frac{h^2}{6}\frac{\partial}{\partial x}\left(\frac{\partial^2 F}{\partial x^2}+\frac{\partial^2 F}{\partial y^2}\right)\Bigg|_{(0,0)}-\frac{h^4}{72}\frac{\partial}{\partial x}\left(\frac{\partial^4 F}{\partial x^4}+2\frac{\partial^4 F}{\partial x^2\partial y^2}+\frac{\partial^4 F}{\partial y^4}\right)\Bigg|_{(0,0)}\\ &-\frac{h^4}{180}\frac{\partial^5 F}{\partial x^5}\Bigg|_{(0,0)}+O(h^5)\end{aligned} \tag{10}$$

$$\begin{aligned}\left.\frac{\partial F}{\partial y}\right|_{(0,0)} =& \frac{1}{12h}\Big(F(-h,h)+4F(0,h)+F(h,h)-F(h,-h)-4F(0,-h)-F(-h,-h)\Big)\\ &-\frac{h^2}{6}\frac{\partial}{\partial y}\left(\frac{\partial^2 F}{\partial x^2}+\frac{\partial^2 F}{\partial y^2}\right)\Bigg|_{(0,0)}-\frac{h^4}{72}\frac{\partial}{\partial y}\left(\frac{\partial^4 F}{\partial y^4}+2\frac{\partial^4 F}{\partial x^2\partial y^2}+\frac{\partial^4 F}{\partial x^4}\right)\Bigg|_{(0,0)}\\ &-\frac{h^4}{180}\frac{\partial^5 F}{\partial y^5}\Bigg|_{(0,0)}+O(h^5)\end{aligned} \tag{11}$$

With this new discretization, the dominant differential operator of the second order error term takes the form:

$$\vec{E}^{(2)} = \left[E_x^{(2)}, E_y^{(2)}\right] = -\frac{h^2}{6}\left[\frac{\partial}{\partial x}\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right), \frac{\partial}{\partial y}\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)\right] \tag{12}$$

If a gradient has a small dependence on direction, this would imply that the dominant error term has only an axial dependence when the function being derived also only depends on the radius. That is, the operator in Eq. (12) applied on $F(r)$ would lead to a function that depends only on the radius:

$$\vec{E}^{(2)}F(r) \equiv \vec{e}_r f(r) \tag{13}$$

with $\vec{e}_r = [\cos(\theta), \sin(\theta)]$ being the unit radial vector. Using the partial derivative operator transformation of Eqs. (8) and (9), and supposing that $F = F(r)$ around (0,0), the

components $E_x^{(2)}F(r)$ and $E_y^{(2)}F(r)$ are:

$$E_x^{(2)}F(r) = \frac{h^2\cos(\theta)}{6r^2}\left(-\frac{\partial F(r)}{\partial r} + r\frac{\partial^2 F(r)}{\partial r^2} + r^2\frac{\partial^3 F(r)}{\partial r^3}\right) \tag{14}$$

$$E_y^{(2)}F(r) = \frac{h^2\sin(\theta)}{6r^2}\left(-\frac{\partial F(r)}{\partial r} + r\frac{\partial^2 F(r)}{\partial r^2} + r^2\frac{\partial^3 F(r)}{\partial r^3}\right) \tag{15}$$

which can be rewritten in the same form as Eq. (13). Similarly, the first differential operator of the fourth order error term in Eqs. (10) and (11) takes the form:

$$\vec{E}_{iso}^{(4)} = \left[E_{x,iso}^{(4)}, E_{y,iso}^{(4)}\right] = -\frac{h^4}{72}\left[\frac{\partial}{\partial x}\left(\frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2\partial y^2} + \frac{\partial^4}{\partial y^4}\right), \frac{\partial}{\partial y}\left(\frac{\partial^4}{\partial y^4} + 2\frac{\partial^4}{\partial x^2\partial y^2} + \frac{\partial^4}{\partial x^4}\right)\right] \tag{16}$$

and the associated components in polar coordinates, when applied to a rotationally invariant function $F(r)$ around (0,0), are given by:

$$E_{x,iso}^{(4)}F(r) = \frac{h^4\cos(\theta)}{72}\left(3\frac{\partial F(r)}{\partial r} - 3r\frac{\partial^2 F(r)}{\partial r^2} + 3r^2\frac{\partial^3 F(r)}{\partial r^3} - 2r^3\frac{\partial^4 F(r)}{\partial r^4} - r^4\frac{\partial^5 F(r)}{\partial r^5}\right) \tag{17}$$

$$E_{y,iso}^{(4)}F(r) = \frac{h^4\sin(\theta)}{72}\left(3\frac{\partial F(r)}{\partial r} - 3r\frac{\partial^2 F(r)}{\partial r^2} + 3r^2\frac{\partial^3 F(r)}{\partial r^3} - 2r^3\frac{\partial^4 F(r)}{\partial r^4} - r^4\frac{\partial^5 F(r)}{\partial r^5}\right) \tag{18}$$

which again meets the rotational invariance requirement, and can be rewritten in the same form as given in Eq. (13). The last differential error operator of the fourth order error term in Eqs. (10) and (11) is:

$$\vec{E}_{ani}^{(4)} = -\frac{h^4}{180}\left[\frac{\partial^5}{\partial x^5}, \frac{\partial^5}{\partial y^5}\right] \tag{19}$$

and can be shown to be anisotropic (i.e. f in Eq. 13 would also be a function of θ). Therefore, the error associated with the anisotropy is lower by two orders when compared to the main second order leading term. It is important to point out that both gradient approximations presented so far are second order approximations in space, but only the latter is a second order approximation in space for the isotropy.

In this work, we only consider the gradient approximation of scalar functions over a square or cubic lattice of unit spacing, so we need to take $h = 1$ from now on. In a more general way, the stencil points with the corresponding weights needed to obtain 2D and 3D isotropic gradients was generalized by [1].

3. Convolution

Here, we present the mathematical and computational aspects of convolution. As finite difference discretization and edge detection kernels are very similar, let us return to the mathematical foundations of these techniques. We often give examples involving 2D images, but we could give the same examples and talk about 2D discrete functions. We don't know the exact coding in the MATLAB and Jacket libraries, as they are under license, but we do have a general idea about function algorithms, which is given in the next section.

3.1. Frequential and spatial filtering

The convolution product of two functions $f(x)$ and $g(x)$, defined by Eq. (20), calculates an average function by sliding g all over f. In common language, we say that the function $g(x)$ acts as a filter of the function $f(x)$.

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x)g(x-s)ds \tag{20}$$

If Eq. (20) defines the convolution of two functions in the space domain, it is also equivalent to the point wise product of these functions in the frequency domain. In mathematics, the convolution theorem for Fourier transforms [5, chap. 4] is formalized by the following equation, where $\mathcal{F}$ corresponds to the Fourier transform applied on a function:

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\} \tag{21}$$

The Fourier transform of a function generates the frequency spectrum of a function. This representation can easily be extended to two dimensions, which is more suitable to image or spatial analysis if the image is a function of two spatial variables. In our case, the output of the Fourier transform will generate a 2D function in the frequency domain from the 2D spatial domain. High frequencies will correspond to information varying rapidly in the original function, while low frequencies correspond to slow variations.

By applying the inverse Fourier Transform to both sides of Eq. (21), we obtain a method for calculating the convolution of two functions:

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\} \tag{22}$$

Therefore, there are two possible ways of convoluting two functions: the Fourier transform pipeline in three steps (Fourier transform, frequency spectrum filtering, and Fourier inverse transform), or the application of a stencil in the spatial domain. Indeed, edge detection kernels acts as high pass filters, accentuating high frequency contributions to the image or to the 2D function in the form of edges and details.

3.2. Discrete circular convolution

Knowing that the properties of the Fourier transform also work for a sampled signal, we present the definition of the 2D discrete Fourier transform (2D DFT), where M and N represent the number of samples in each dimension, x and y are the discrete spatial variables, and u and v are the transform or frequency variables:

$$\mathcal{F}\{u,v\} = \frac{1}{MN} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) e^{-2\pi j(\frac{ux}{N} + \frac{vy}{M})} \tag{23}$$

In addition to having the same properties as the continuous Fourier transform, which are outside the scope of this presentation and can be found in [6], there are two more in 2D DFT that are important: periodicity and separability. Since the discrete signal is sampled from the finite length sequences, its frequency spectrum will be periodic. In the spatial domain, this property allows us to slide the filter from one frontier to the opposite one, and then start again at the frontier where we began. Moreover, its Fourier transform is separable. The 2D DFT

can be processed in two steps: applying 1D DFT on the lines, and then applying 1D DFT on the resulting columns. In the spatial domain, a 2D filter represented by K_1-by-K_2 matrix is separable if it can be obtained by the outer product of two vectors of size K_1 and K_2, knowing that DFT separability does not imply that the matrix kernel is always separable.

As an example, let us take kernelX, the 2D separable kernel in section (4.2):

$$\frac{1}{6}\begin{bmatrix} 3 \\ 0 \\ -3 \end{bmatrix} \otimes \frac{1}{6}\begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix} = \frac{1}{36}\begin{bmatrix} 3 \\ 0 \\ -3 \end{bmatrix}\begin{bmatrix} 1 & 4 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{12} & \frac{1}{3} & \frac{1}{12} \\ 0 & 0 & 0 \\ -\frac{1}{12} & -\frac{1}{3} & -\frac{1}{12} \end{bmatrix} \tag{24}$$

We can easily deduce the main functionality of the 2D kernel, the gradient component in x that approximates the derivative in x, by calculating the difference between the first and third rows in the neighborhood of a point, instead of the first and third columns, because of the rotation of our axis system. The separated 1D filters give us further information about the kernel function: a smoothing mask is applied on the differentiating mask, in order to reduce noise that could be exaggerated by the first filter.

Now, to explain how to apply a circular convolution mask in the spatial domain, we go back to the definition of convolution for discrete functions, presented in 2D and 3D in Eqs. (25) and (26) respectively, where A is the image and B is the mask. Processing a convolution filter consists of computing the scalar product of the filter weights with the input values within a window of the filter dimensions surrounding each of the output values, after flipping the mask in each dimension, as described in [7, chap. 4].

$$(A * B)(x, y) = \sum_i \sum_j A(i, j)B(x - i, y - j) \tag{25}$$

$$(A * B)(x, y, z) = \sum_i \sum_j \sum_k A(i, j, k)B(x - i, y - j, z - k) \tag{26}$$

In circular convolution, the equation is slightly modified to model periodic shifting. Here we consider a square image N by N:

$$(A \circledast B)(x, y) = \sum_i \sum_j A(i, j)B[(x - i) \mod N, (y - j) \mod N] \tag{27}$$

The values on the image we see in the window of the K-by-K filter are distributed on a periodic surface. The 2D filter is rolled up in the opposite direction and will turn in a clockwise direction. For each output value, the stencil is lined up on the input values, the scalar product is applied, and then the stencil rotates to shift to the next position to compute. Therefore, in the spatial domain, an N-by-N image convoluted with a K-by-K filter requires N^2K^2 multiplications and nearly N^2K^2 additions. The K^2 multiplications or additions can be reduced to $2K$ if the filter is separable, resulting in a total complexity of $O(N^2K)$ for spatial convolution with a separable filter.

In the frequency domain, the 2D discrete Fourier transform and its inverse are computed using the divide-and-conquer algorithm of the Fast Fourier Transform (FFT and IFFT), which reduces the complexity from N^3 multiplications and N^3 additions to $2N^2 \log_2(N)$ operations. The same order of complexity is required to compute the FFT of the K-by-K filter: $2K^2 \log_2(K)$.

To those computational complexities, the product cost of two complex numbers, which is six operations, has to be added N^2 times. In the frequency domain, the resulting computational complexity of convoluting an N-by-N image by a K-by-K filter would equal $2N^2 \log_2(N) + K^2 \log_2(K) + 6N^2$, or $O(N^2 \log_2(N))$ if $N \ggg K$. Furthermore, the separability property could be used to implement a 2D FFT by the application of 1D FFT along each direction, in turn, with a global transpose between each of the 1D transform, which computationnal complexity is $O(N \log_2(N))$. To decide whether to use the FFT algorithm or spatial convolution, the two complexity functions should be compared to verify that convolution in the spatial domain performs better than FFT for small kernels only.

4. MATLAB

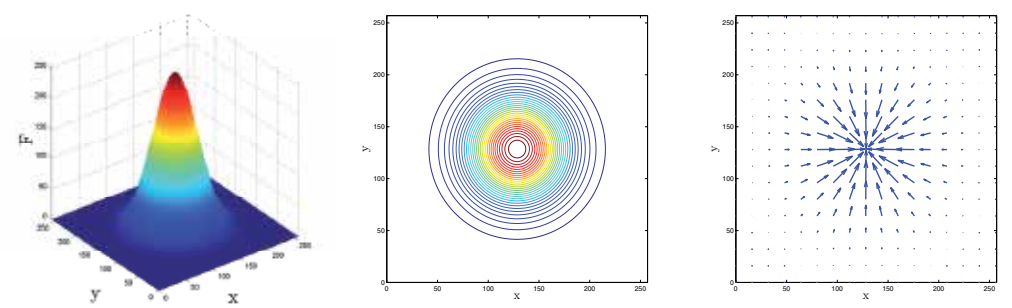
In this section, we present a systematic method that can be applied in MATLAB to calculate the gradient of scalar functions on a square lattice, or in 2D, which is simply the gradient of the images. The 3D case is a straightforward extension of the 2D case. First, let us define a function with which it is possible to test the gradient discretization previously defined in section (2.2):

$$F(x,y) = Ne^{-30\left((\frac{x}{N}-\frac{1}{2})^2+(\frac{y}{N}-\frac{1}{2})^2\right)} \tag{28}$$

For the purpose of this test and for simplicity, we consider only a square image N in width. The exponential function is multiplied by N to make sure that the scaling ratio of the function F is always the same, whatever the value of N. In MATLAB m-code, this function can be defined as:

```
N=256+1;Nx=N;Ny=N;
[X,Y]=ind2sub([Nx Ny],1:Nx*Ny);
X=reshape(X,Nx,Ny)-1;Y=reshape(Y,Nx,Ny)-1;
F=N*exp(-30*(((X)/N-0.5).^2+((Y)/N-0.5).^2));
```

Figures 1(a) and 1(b) show the surface shape and contour of this function.



(a) Surface shape. (b) Contour. (c) Gradient.

Figure 1. Characteristics of the function $F(x,y)$.

4.1. Naive computation of the gradient

When the gradient of a function needs to be computed, it may be tempting to apply Eqs. (10) and (11) directly. This is quite a simple approach, and in MATLAB can be applied as follows:

```
% Stencils and weights

w1=1/3;w2=1/12;
xi=[w1*ones(1,4),w2*ones(1,4)];
d=unique([perms([1 0]);perms([-1 0])],'rows');
d=[d;unique([perms([1 1]);perms([-1 1]);perms([-1 -1])],'rows')];
dxX=(d(d(:,1)~=0,1));dyX=(d(d(:,1)~=0,2));
dxY=(d(d(:,2)~=0,1));dyY=(d(d(:,2)~=0,2));
xiX=(xi(d(:,1)~=0));xiY=(xi(d(:,2)~=0));

% Periodic padding

nPad=max(dxX);
Fpad=padarray(F,[nPad nPad],'circular');

% Naive computation of the gradient

gradFX=0;
for i=1:numel(dxX)
    gradFX=gradFX+xiX(i)*dxX(i).*Fpad(nPad+(1:Nx)+dxX(i),nPad+(1:Ny)+dyX(i));
end

gradFY=0;
for i=1:numel(dyY)
    gradFY=gradFY+xiY(i)*dyY(i).*Fpad(nPad+(1:Nx)+dxY(i),nPad+(1:Ny)+dyY(i));
end
```

However, this is rather a naive and slow implementation for MATLAB. It is far more efficient to evaluate the gradient by using a convolution product, as presented in section (4.2).

4.2. Techniques for calculating the gradient with convolution

Instead of calculating the gradient as previously shown, the use of a convolution product between the functions to differentiate them, and a kernel representing the weights and stencils of the finite difference approximation is more beneficial. For the isotropic discretization of Eqs. (10) and (11), this computation can be performed in MATLAB as follows:

```
% 2D kernel

w1=1/3;w2=1/12;
c01=1*[w2;w1;w2];
kernelX=[-c01,zeros(3,1),c01];
kernelX=-kernelX';
kernelY=kernelX';

% Periodic padding

nPad=(numel(kernelX(:,1))-1)/2;
Fpad=padarray(F,[nPad nPad],'circular');
```

```
% Fast convolution for computing the gradient

gradFX=conv2(Fpad,kernelX,'valid');
gradFY=conv2(Fpad,kernelY,'valid');
```

A useful list of 2D and 3D kernels for calculating isotropic gradients is available in sections (6.1) and (6.2).

4.3. Separable kernel

Sometimes the kernel is separable, which means that, instead of applying an nD convolution, we can apply an 1D convolution n times to obtain the same result with much less computational effort. The previous kernel for computing a 2D isotropic gradient with only the nearest neighbors is, fortunately, separable, and in MATLAB the evaluation takes place as follows:

```
% 1D kernels

kernelXsep10=[3 0 -3]'/6;kernelXsep01=[1 4 1]/6;
kernelYsep10=[1 4 1]'/6;kernelYsep01=[3 0 -3]/6;

% Faster separable convolution for computing the gradient

gradFXconvSep=conv2(kernelXsep10,kernelXsep01,Fpad,'valid');
gradFYconvSep=conv2(kernelYsep10,kernelYsep01,Fpad,'valid');
```

Note that in the work of [1], one of the most important 3D isotropic kernels, which is the one that uses the nearest neighbors only, was not presented. In this work, two variants of this kernel were obtained, one using only 10 nearest neighbors and the other using only 18 nearest neighbors, and their MATLAB form are given in section (6.2). Moreover, the 3D kernel using 18 nearest neighbors has the advantage of being separable, which means that we should expect to be able to rapidly compute a 3D gradient of low isotropic order using the knowledge presented in this chapter. Also note that the higher isotropic order (>2nd) kernels given in sections (6.1) and (6.2) are not separable.

4.4. Accuracy

In this work, the order accuracy is defined by n, the value in h^n in Taylor's series expansion for which the space or the isotropic order is achieved. Therefore, the space and isotropic order may be different. Although this may not have been explicitly stated by [1, 2], all isotropic gradients found in sections (6.1) and (6.2) are of second order accuracy in space, i.e. if the lattice size doubles, the error on the gradient is divided approximately by four. This can be demonstrated numerically, or analytically by means of the expression $E_x^{(2)}F(r)$ and $E_y^{(2)}F(r)$ in polar coordinates, as in Eqs. (14) and (15). These expressions are non zero for every isotropic gradient presented in this work.

The main point about the kernel that we make in this study is that, as its size increases, it becomes possible to set isotropic the higher order error term in the Taylor series expansion at fixed space second order accuracy. It is therefore important to be careful not to confuse

space order accuracy and isotropic order accuracy. We believe that, based on this result, future research could provide other kernels (perhaps of similar size) for which the leading space order accuracy would be the same as the leading isotropic order accuracy. For some applications, finding such kernels could be a significant step forward. In fact, achieving leading higher space order accuracy with equal leading isotropic order accuracy might have greater impact than achieving leading low space order with very high isotropic order accuracy, as is currently the case. However, higher space order gradient discretizations may suffer from another non physical numerical artifacts, known as spurious oscillations.

4.5. Performance

As previously indicated, computation of the gradient by applying convolution is faster than using a simpler, more straightforward method, but a naive one. We present some numerical results in this section that will show that this is indeed true. We also show that using a GPU instead of a CPU significantly reduces computation time.

First, all performance testing consists in evaluating gradients of random 2D and 3D double precision images. Note that we suppose a periodic padding for these images. All these image gradients are computed using MATLAB with the `-singleCompThread` startup option. This is done for benchmarking purposes, because the reference case should be computed using a sequential algorithm, that is, with a single core only. The CPU used in this work is an Intel® Core i7-970 processor, while the GPU is an Nvidia® GeForce GTX 580. All computations on the GPU are performed in MATLAB via the Jacket plugin developed by AccelerEyes®. The version of MATLAB is R2010b, and the Jacket version is 2.0 (build a15607c).

The timing method on the CPU is the usual MATLAB `tic; m-code; toc;` procedure. However, this method is not suited for timing m-code on the GPU. For this, we refer the reader to the method proposed by [8]. In Figures 2-5, the time taken for one simulation "dot" or result "dot" is the average of a hundred simulations.

To test performance, five different algorithms are considered for computing the gradient:

1. MATLAB singlethread naive (section 4.1)
2. MATLAB singlethread convolution (section 4.2) [REFERENCE CASE]
3. MATLAB Jacket naive
4. MATLAB Jacket convolution
5. MATLAB Jacket GFOR + convolution

Note that all results differ with respect to machine accuracy in double precision, and that the padding of the images is computed on the fly to save computer memory. This is because padding is very cheap in terms of computing cost, when compared to the cost of evaluating the gradient.

Case (4), MATLAB Jacket convolution, is the GPU equivalent of reference case (2) with a CPU. Case (3) is the GPU equivalent of case (1) with a CPU. The last case (5), MATLAB Jacket GFOR + convolution, is a special case that is not available on the CPU. To explain this, let us suppose that the user wishes to evaluate the gradient of N different images simultaneously. Using MATLAB, this is not possible without `parfor` , which is only available with the Parallel

Computing Toolbox. Also note that `parfor` is usually used for coarse-grained parallelism, while `gfor` can be used for fine-grained parallelism. Without `parfor`, an `for` loop is required to evaluate the gradients of the images sequentially, and to store the results in one large matrix by subassignment. With Jacket, it is possible to perform these N convolutions in parallel on the GPU using an `gfor` loop. Usually, this significantly reduces computation time, compared to the sequential `for` loop. More details on the functionality and limitations of the `gfor` loop can be found in [8]. In order to be able to compare the `gfor` loop case to the other cases, all performance tests have been conducted with $N = 3$, unless otherwise stated, i.e. it is supposed the user needs to evaluate three image gradients simultaneously at a given time.

In all the figures showing performance behavior, the y-axis is in log scale. Figures 2(a) and 2(b) show the performance speedup with the 2D 2nd and 14th order isotropic gradients as a function of image size. For large images, speedups of $31x$ to $39x$ can be achieved.

Figures 3(a) and 3(b) show the same situation, but with the 3D 2nd and 8th order isotropic gradients. For large images, a speedup of $34x$ to $124x$ can be achieved. These results indicate an important speedup and show the usefulness of Jacket for MATLAB.

Figures 4(a) and 4(b) show the speedup with the 2D and 3D isotropic gradient as function of the isotropy order at a fixed image size. In 2D, images are 992x992, and in 3D they are 110x110x110. For high isotropy, speedups of $49x$ to $124x$ can be achieved.

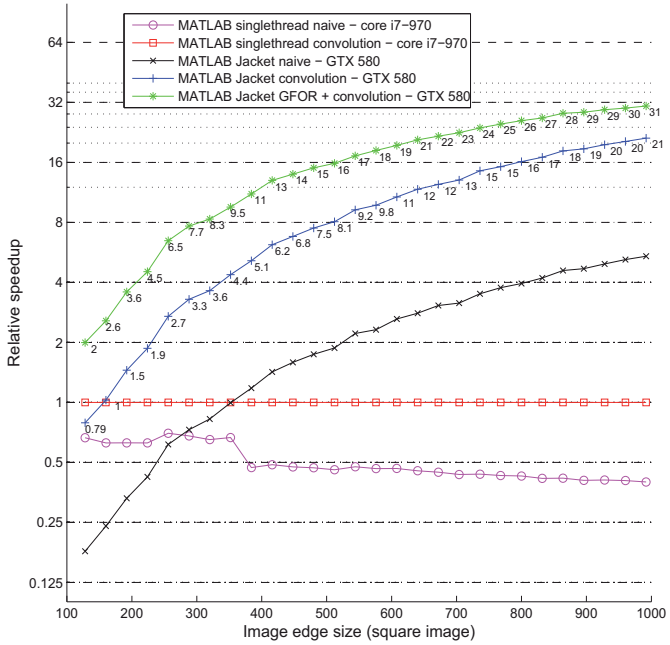
Figure 5 shows the speedup with the 3D 8th order isotropic gradient as a function of N, the number of images to be evaluated simultaneously at a fixed image size of 110x110x110. Speedups of $86x$ to $138x$ can be achieved, depending on the value of N.

For both 2D and 3D cases, as the isotropy order or the number of images to evaluate simultaneously increases, the speedup that can be achieved using the GPU also increases. This is to be expected, since the computational complexity increases. Nevertheless, the chances of obtaining a speedup of $86x$ to $138x$ for the more challenging case were much better than we had anticipated. In fact, the Jacket plugin allowed us to tackle problems that would not have been possible to deal with without using a low level programming language.

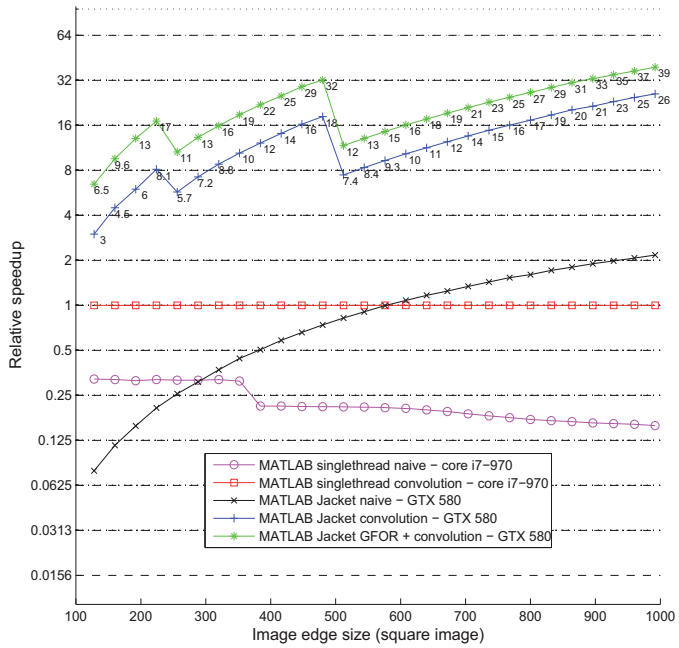
We must remember that the speedups were computed with the reference case, which uses convolution. It is important to note that a speedup of $280x$ may be achieved for the most computationally expensive test (Fig. 5) when comparing the two following cases: Jacket GFOR + convolution, and MATLAB singlethread naive. Thus, depending on how the gradient evaluations are implemented, a huge difference in computational time may be reached. Note that Jacket code does not currently take advantage of the zero entry in the kernel, and that the convolutions are performed in the spatial domain only for 9x9 kernels in 2D and 3x3x3 kernels in 3D. The naive implementation always takes advantage of the zero entry in the kernel, which means that the zero entry could yield an additional speedup, because some kernels are sparse. For example, 2D kernels have 26% to 68% zeros, while in 3D, this number varies from 33% to 84%. The 3x3 kernel contains 33% zeros while the 3x3x3 kernels contain 67% or 33% zeros.

5. Scientific application: lattice Boltzmann method

The lattice Boltzmann method is a computational approach that is mainly used for fluid flow simulation with its roots in the field of cellular automata [9]. This method is particularly useful for solving complex flow systems, such as multiphase flows, in porous media where classical

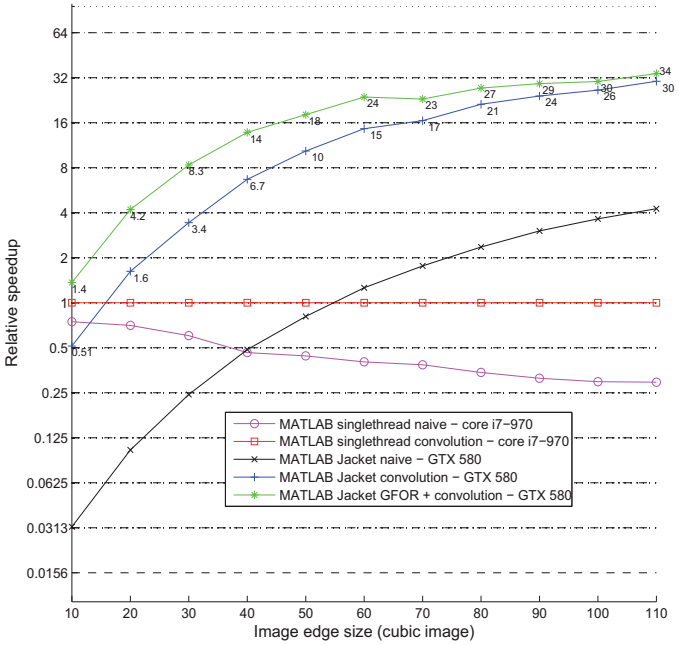


(a) 2nd order isotropic gradient.

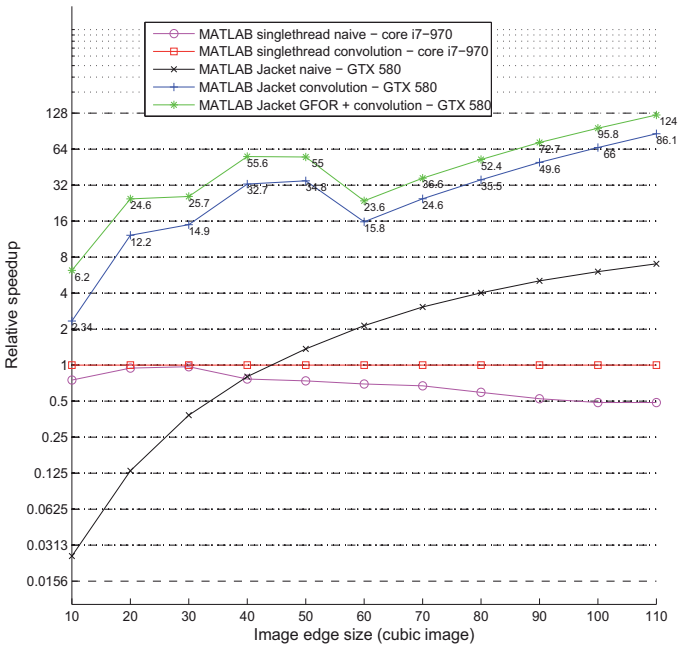


(b) 14th order isotropic gradient.

Figure 2. Speedup as a function of image size (2D isotropic gradient).

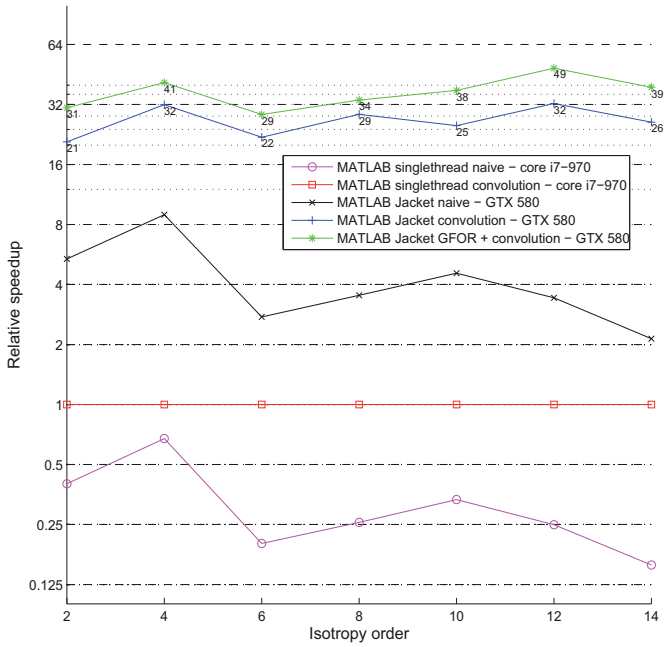


(a) 2nd order isotropic gradient.

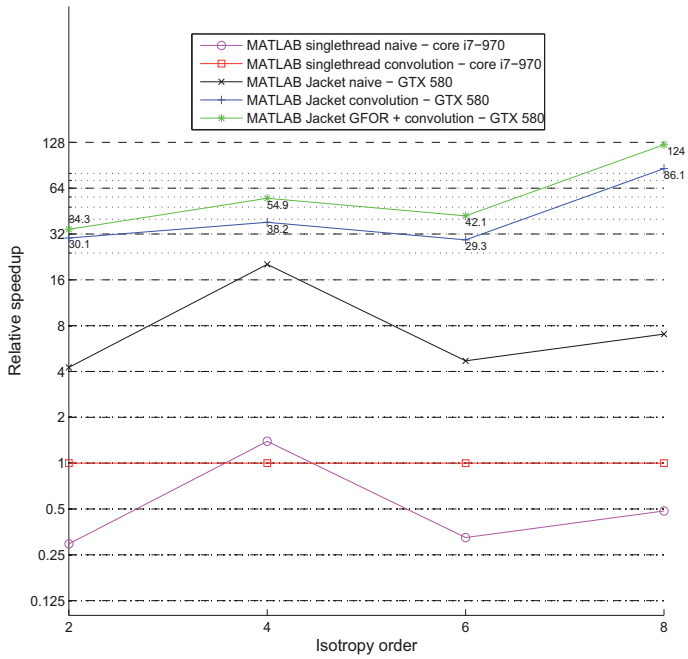


(b) 8th order isotropic gradient.

Figure 3. Speedup as a function of image size (3D isotropic gradient).



(a) Square image (992x992).



(b) Cubic image (110x110x110).

Figure 4. Speedup as a function of the gradient isotropy order.

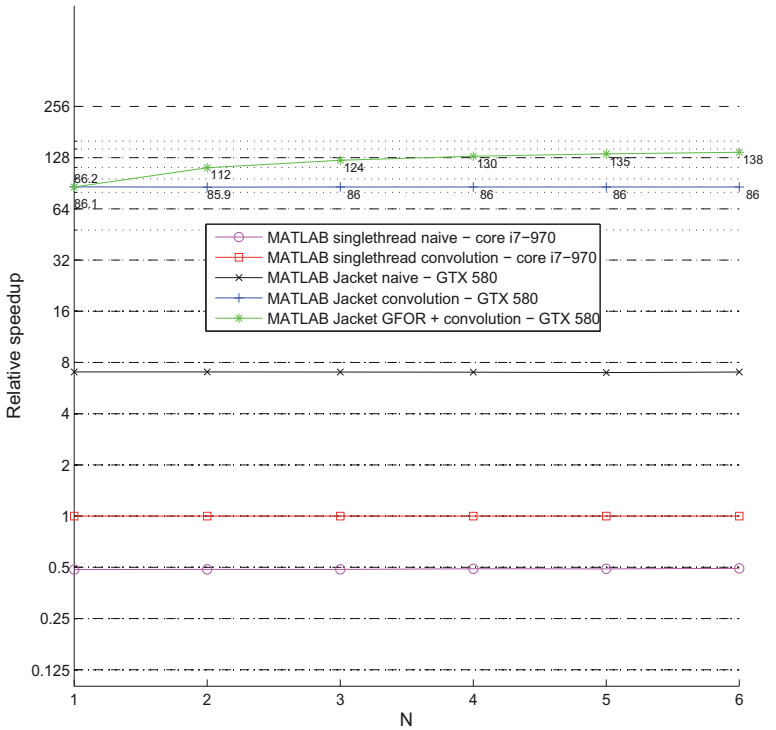


Figure 5. Speedup as a function of the number of cubic images (110x110x110) to be evaluated simultaneously using the 3D 8th order isotropic gradient.

approaches based on the Navier-Stokes equations, like finite volumes or finite elements, encounter some difficulties. The method we use is based on the original Ph.D. thesis of [10]. Since then, there have been several improvements. However, these enhancements are outside the scope of this chapter, and will not be described.

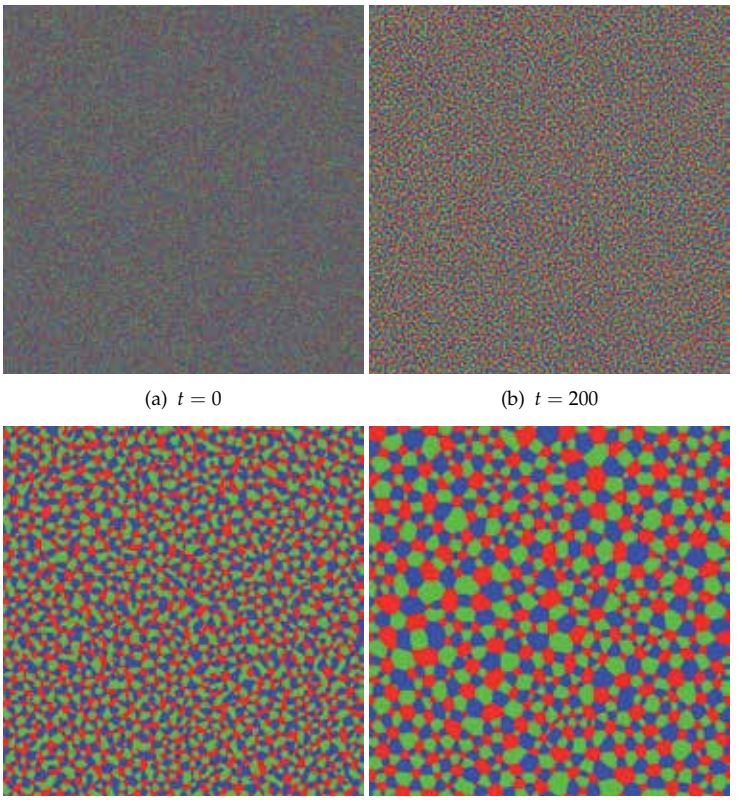
The isotropic gradients we present here are useful for simulating immiscible multiphase flows, where the orientation of the various fluid interfaces has to be computed very frequently. The gradients of the density of each fluid color (phase) define the normal orientation of the interface, and special operators are used to keep the different interfaces between the fluids defined. Moreover, the norm of these gradients serves as a means to introduce a certain amount of surface tension between the fluids.

Suppose we wish to simulate the behavior of three fluids. At a certain point in the algorithm, three image gradients need to be computed, corresponding to the interface normal of each fluid density. Here is where the methods presented in this chapter become useful, a situation described in section (4.5) with $N = 3$. It is important to point out that the isotropy of the gradient helps to reduce some of the computational artifacts that appear at the interface [1, 2]. These are called spurious currents in the lattice Boltzmann community, and are non-physical.

In multiphase flow simulation, calculation of a high order isotropic gradient is the most expensive part of the method, and the use of Jacket, has enabled us to reduce the

computational cost by an astonishing amount. This type of calculation would not have been possible, in a reasonable time, by applying plain MATLAB.

We end this section with a simulation example that shows the spinodal decomposition of a three phase flow. This flow consists of an initial random mixture, where each phase self-agglomerates until a steady state is achieved. Figure 6 shows the spinodal decomposition at various times (in lattice Boltzmann units). Note that this simulation is given for illustration purposes only, and that spinodal decomposition has been quantitatively studied by [11].



(a) $t = 0$ (b) $t = 200$

(c) $t = 2000$ (d) $t = 100000$

Figure 6. Random spinodal decomposition of a three phase flow.

6. Convolution kernel for discrete gradient approximation

In this section, we give several convolution kernels in the form of MATLAB m-code, which is very useful for calculating 2D/3D isotropic gradients on a square or cubic lattice. Most of the weights were taken from Ref. [1].

6.1. Convolution kernel for a 2D isotropic gradient

```
% 2D isotropy up to 2nd order
% This approximation use nearest neighbors only
```

```
w1=1/3;w2=1/12;
c01=1*[w2;w1;w2];
kernel=[-c01,zeros(3,1),c01];

% 2D isotropy up to 2nd order
% This approximation use nearest neighbors only (separable version)

kernelXsep10=[-3 0 3]'/6;kernelXsep01=[-1 -4 -1]/6;
kernelYsep10=[-1 -4 -1]'/6;kernelYsep01=[-3 0 3]/6;

% 2D isotropy up to 4th order

w1=4/15;w2=1/10;w4=1/120;
c01=1*[0;w2;w1;w2;0];c02=2*[0;0;w4;0;0];
kernel=[-c02,-c01,zeros(5,1),c01,c02];

% 2D isotropy up to 6th order

w1=4/21;w2=4/45;w4=1/60;w5=2/315;w8=1/5040;
c01=1*[w5;w2;w1;w2;w5];c02=2*[w8;w5;w4;w5;w8];
kernel=[-c02,-c01,zeros(5,1),c01,c02];

% 2D isotropy up to 8th order

w1=262/1785;w2=93/1190;w4=7/340;w5=6/595;
w8=9/9520;w9=2/5355;w10=1/7140;
c01=1*[w10;w5;w2;w1;w2;w5;w10];
c02=2*[0;w8;w5;w4;w5;w8;0];
c03=3*[0;0;w10;w9;w10;0;0];
kernel=[-c03,-c02,-c01,zeros(7,1),c01,c02,c03];

% 2D isotropy up to 10th order

w1=68/585;w2=68/1001;w4=1/45;w5=62/5005;w8=1/520;
w9=4/4095;w10=2/4095;w13=2/45045;w16=1/480480;
c01=1*[0;w10;w5;w2;w1;w2;w5;w10;0];
c02=2*[0;w13;w8;w5;w4;w5;w8;w13;0];
c03=3*[0;0;w13;w10;w9;w10;w13;0;0];
c04=4*[0;0;0;0;w16;0;0;0;0];
kernel=[-c04,-c03,-c02,-c01,zeros(9,1),c01,c02,c03,c04];

% 2D isotropy up to 12th order

w1=19414/228375;w2=549797/10048500;w4=175729/7917000;w5=50728/3628625;
w8=3029/913500;w9=15181/7536375;w10=221/182700;w13=68/279125;
w16=1139/26796000;w17=68/2968875;w18=17/1425060;w20=17/5742000;
w25_50=1/32657625;w25_34=1/32657625;
c01=1*[0;w17;w10;w5;w2;w1;w2;w5;w10;w17;0];
c02=2*[0;w20;w13;w8;w5;w4;w5;w8;w13;w20;0];
c03=3*[0;w25_34;w18;w13;w10;w9;w10;w13;w18;w25_34;0];
c04=4*[0;0;w25_34;w20;w17;w16;w17;w20;w25_34;0;0];
c05=5*[0;0;0;0;0;w25_50;0;0;0;0;0];
kernel=[-c05,-c04,-c03,-c02,-c01,zeros(11,1),c01,c02,c03,c04,c05];

% 2D isotropy up to 14th order

w1=285860656/3979934595;w2=2113732952/43779280545;w4=940787801/43779280545;
w5=124525000/8755856109;w8=15841927/3979934595;w9=2046152/795986919;
w10=14436304/8755856109;w13=18185828/43779280545;w16=13537939/140093697744;
w17=231568/3979934595;w18=1516472/43779280545;w20=18769/1591973838;
```

```
w25_50=184/315867825;w25_34=464/795986919;w26=1448/4864364505;
w29=148/4864364505;w32=629/400267707840;
c01=1*[w26;w17;w10;w5;w2;w1;w2;w5;w10;w17;w26];
c02=2*[w29;w20;w13;w8;w5;w4;w5;w8;w13;w20;w29];
c03=3*[0;w25_34;w18;w13;w10;w9;w10;w13;w18;w25_34;0];
c04=4*[0;w32;w25_34;w20;w17;w16;w17;w20;w25_34;w32;0];
c05=5*[0;0;0;w29;w26;w25_50;w26;w29;0;0;0];
kernel=[-c05,-c04,-c03,-c02,-c01,zeros(11,1),c01,c02,c03,c04,c05];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Final 2D kernels for gradient approximation %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

kernelX=-kernel';
kernelY=permute(kernelX,[2 1]);
```

6.2. Convolution kernel for a 3D isotropic gradient

```
% 3D isotropy up to 2nd order
% This approximation use only 10 nearest neighbors
% Note that this kernel IS NOT separable

w1=1/6;w2=1/12;w3=0;
c01=1*[w2;w1;w2];c11=1*[w3;w2;w3];
kernel(:,:,1)=[-c11,zeros(3,1),c11];
kernel(:,:,2)=[-c01,zeros(3,1),c01];
kernel(:,:,3)=kernel(:,:,1);

% 3D isotropy up to 2nd order
% This approximation use only 18 nearest neighbors
% Note that this kernel IS separable

w1=2/9;w2=1/18;w3=1/72;
c01=1*[w2;w1;w2];c11=1*[w3;w2;w3];
kernel(:,:,1)=[-c11,zeros(3,1),c11];
kernel(:,:,2)=[-c01,zeros(3,1),c01];
kernel(:,:,3)=kernel(:,:,1);

% 3D isotropy up to 4th order

w1=2/15;w2=1/15;w3=1/60;w4=1/120;
c01=1*[0;w2;w1;w2;0];c02=2*[0;0;w4;0;0];c11=1*[0;w3;w2;w3;0];
c12=2*[0;0;0;0;0];c21=1*[0;0;0;0;0];c22=2*[0;0;0;0;0];
kernel(:,:,1)=[-c22,-c21,zeros(5,1),c21,c22];
kernel(:,:,2)=[-c12,-c11,zeros(5,1),c11,c12];
kernel(:,:,3)=[-c02,-c01,zeros(5,1),c01,c02];
kernel(:,:,4:5)=kernel(:,:,2:-1:1);

% 3D isotropy up to 6th order

w1=4/45;w2=1/21;w3=2/105;w4=5/504;w5=1/315;w6=1/630;w8=1/5040;
c01=1*[w5;w2;w1;w2;w5];c02=2*[w8;w5;w4;w5;w8];c11=1*[w6;w3;w2;w3;w6];
c12=2*[0;w6;w5;w6;0];c21=1*[0;w6;w5;w6;0];c22=2*[0;0;w8;0;0];
kernel(:,:,1)=[-c22,-c21,zeros(5,1),c21,c22];
kernel(:,:,2)=[-c12,-c11,zeros(5,1),c11,c12];
kernel(:,:,3)=[-c02,-c01,zeros(5,1),c01,c02];
kernel(:,:,4:5)=kernel(:,:,2:-1:1);
```

```
% 3D isotropy up to 8th order

w1=352/5355;w2=38/1071;w3=271/14280;w4=139/14280;w5=53/10710;w6=5/2142;
w8=41/85680;w9_221=1/4284;w9_300=1/5355;w10=1/10710;w11=1/42840;
c01=1*[w10;w5;w2;w1;w2;w5;w10];c02=2*[0;w8;w5;w4;w5;w8;0];
c03=3*[0;0;w10;w9_300;w10;0;0];c11=1*[w11;w6;w3;w2;w3;w6;w11];
c12=2*[0;w9_221;w6;w5;w6;w9_221;0];c13=3*[0;0;w11;w10;w11;0;0];
c21=1*[0;w9_221;w6;w5;w6;w9_221;0];c22=2*[0;0;w9_221;w8;w9_221;0;0];
c23=3*[0;0;0;0;0;0;0];c31=1*[0;0;w11;w10;w11;0;0];
c32=2*[0;0;0;0;0;0;0];c33=3*[0;0;0;0;0;0;0];
kernel(:,:,1)=[-c33,-c32,-c31,zeros(7,1),c31,c32,c33];
kernel(:,:,2)=[-c23,-c22,-c21,zeros(7,1),c21,c22,c23];
kernel(:,:,3)=[-c13,-c12,-c11,zeros(7,1),c11,c12,c13];
kernel(:,:,4)=[-c03,-c02,-c01,zeros(7,1),c01,c02,c03];
kernel(:,:,5:7)=kernel(:,:,3:-1:1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Final 3D kernels for gradient approximation %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

kernelX=-permute(kernel,[2 1 3]);
kernelY=permute(kernelX,[2 1 3]);
kernelZ=permute(kernelX,[3 2 1]);
```

6.3. Convolution kernel for 2D anisotropic edge detection

```
% 2D Prewitt kernel

w1=1;w2=1;
c01=1*[w2;w1;w2];
kernel=[-c01,zeros(3,1),c01];

% 2D Sobel kernel

w1=2;w2=1;
c01=1*[w2;w1;w2];
kernel=[-c01,zeros(3,1),c01];

% 2D Scharr kernel

w1=3;w2=10;
c01=1*[w2;w1;w2];
kernel=[-c01,zeros(3,1),c01];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Final 2D kernels for gradient approximation %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

kernelX=-kernel';
kernelY=permute(kernelX,[2 1]);
```

7. Conclusion

In this work, a detailed description of isotropic gradient discretizations and convolution products has been presented. These isotropic gradients are useful, and superior to anisotropic

discretizations. This is especially true in the field of flow simulation, when the lattice Boltzmann method is used. However, high order isotropic gradients are computationally expensive. To address this issue, we combined the convolution product with the Jacket plugin in MATLAB and GPU hardware, which enabled us to achieve high computational speedups (up to 138x), compared to plain MATLAB computations using a CPU. We end this chapter with one final note. While we have provided a useful list of MATLAB m-code defining the kernels needed for evaluating 2D and 3D isotropic gradients, these kernels only lead to second order space accuracy with high isotropic order. The development of kernels that would lead to high order space accuracy combined with high isotropic order could generate significant benefits, particularly for the lattice Boltzmann community addressing multiphase flows. However, the benefits may not be straightforward, because the higher space order gradient discretization may lead to another unwanted numerical artifacts, such as spurious oscillations in regions separating two phases.

Acknowledgments

We extend our special thanks to Pavan Yalamanchili from AccelerEyes for his quick response to our queries and his generous support. We applied the sequence-determines-credit (SDC) approach to our listing of authors [12]. This work was supported by a grant from the NSERC (Natural Sciences and Engineering Research Council of Canada).

Author details

Sébastien Leclaire, Maud El-Hachem and Marcelo Reggio
Department of Mechanical Engineering, École Polytechnique de Montréal, Canada

8. References

[1] M. Sbragaglia, R. Benzi, L. Biferale, S. Succi, K. Sugiyama, and F. Toschi. Generalized lattice boltzmann method with multirange pseudopotential. *Physical Review E*, 75(2):026702, 2007.
[2] Sébastien Leclaire, Marcelo Reggio, and Jean-Yves Trépanier. Isotropic color gradient for simulating very high-density ratios with a two-phase flow lattice boltzmann model. *Computers & Fluids*, 48(1):98–112, 2011.
[3] Sayed Kamaledin Ghiasi Shirazi and Reza Safabakhsh. Omnidirectional edge detection. *Computer Vision and Image Understanding*, 113(4):556 – 564, 2009.
[4] M.K. Ray, D. Mitra, and S. Saha. Simplified novel method for edge detection in digital images. In *Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011 International Conference on*, pages 197–202, july 2011.
[5] T.L. Chow. *Mathematical methods for physicists: a concise introduction*. Cambridge University Press, 2000.
[6] E. Oran Brigham. *The fast Fourier transform and its applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
[7] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.
[8] Accelereyes website, 2012.
[9] Hasslacher B. Frish U and Y. Pomeau. *Phys. Rev. Lett.*, 56(14):1505, 1986.

[10] Andrew K. Gunstensen. *Lattice-Boltzmann Studies of Multiphase Flow Through Porous Media*. PhD thesis, 1992.

[11] F. J. Alexander, S. Chen, and D. W. Grunau. Hydrodynamic spinodal decomposition: Growth kinetics and scaling functions. *Physical Review B*, 48(1):634, 1993.

[12] Teja Tscharntke, Michael E. Hochberg, Tatyana A. Rand, Vincent H. Resh, and Jochen Krauss. Author sequence and credit for contributions in multiauthored publications. *PLoS Biol*, 5(1):e18, 2007.

Section 2

MATLAB General Applications

Chapter 10

MATLAB/Simulink-Based Grid Power Inverter for Renewable Energy Sources Integration

Marian Gaiceanu

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/48489

1. Introduction

The main objective of the chapter is the development of technological knowledge, based on Matlab/Simulink programming language, related to grid connected power systems for energy production by using Renewable Energy Sources (RES), as clean and efficient sources for meeting both the environment requirements and the technical necessities of the grid connected power inverters. Another objective is to promote the knowledge regarding RES; consequently, it is necessary to bring contribution to the development of some technologies that allow the integration of RES in a power inverter with high energy quality and security. By using these energetic systems, the user is not only a consumer, but also a producer of energy. This fact will have a direct impact from technical, economic and social point of view, and it will contribute to the increasing of life quality.

The chapter intends to integrate itself into the general frame of the EU energy policies by imposing the global objectives of reducing the impact upon the environment, and promoting the RES for the energy production. At the same time, the chapter is strategically oriented towards the compatibility with the priority requirements from some European programmes: the wide-spread implementation of the distributed energy sources, of the energy storage technologies and of the grid connected systems.

The chapter strategy follows two directions: the first, is thedevelopment of knowledge (a study and implementation of a high performance grid-power inverter; the fuel cells technology as RES; the control methods; specific modelling and simulation methods); the second focuses upon the applicative research (a real time implementation with dSPACE platform is provided).

The interdisciplinarity of the chapter consists of using specific knowledge from the fields of: energy conversion, power converters, Matlab/Simulink simulation software, real time

implementation based on dSPACE platform, electrotechnics, and advanced control techniques.

2. The grid power converter

The increased power demand, the depletion of the fossil fuel resources and the growth of the environmental pollution have led the world to think seriously of other alternative sources of energy: solar, wind, biogas/biomass, tidal, geothermal, fuel cell, hydrogen energy, gas micro turbines and small hydropower farms.

The number of distributed generation (DG) units, including both renewable and nonrenewable sources, for small rural communities not connected to the grid and for small power resources (up to 1000 kW) connected to the utility network has grown in the last years. There has been an increase in the number of sources that are natural DC sources, for instance fuel cells and photovoltaic arrays, or whose AC frequency is either not constant or is much higher than the grid frequency, for instance micro gas-turbines. These generators necessarily require an DC/AC converter to be connected to the grid. Although some generators can be connected directly to the electric power grid, such as wind power driven asynchronous induction generators, there is a trend to adopt power electronics based interfaces which convert the power firstly to DC and then use an inverter to deliver the power to the 50Hz AC grid.

At the international level**,** SMA Technologies AG (www.sma.de) promotes the innovative technology based on the renewable sources. The following results can be mentioned: the stand-alone or grid connected systems by using either a single type of source (Sunny Boy 5000 Multi-String inverter based on the modular concept, Hydro-Boy and Windy Boy) or combined (Sunny Island) including the interconnection of wind turbines, photovoltaics, micro-hydro and diesel generators. It is well-known that for systems efficiency increasing, the inverter is the answer of the problem. With this respect, Sunways (www.sunways.de) adopted the HERIC concept (from the Fraunhofer Solar and Energetic Systems Institute), by using a tranformerless inverter, obtaining a 97,33% high efficiency of the inverter for low powers (www.ise.fraunhofer.de). The Master-Slave and Team concepts are embedded in SunnyTeam and Fronius inverters in order to increase the efficiency in the partial load conditions. At world level, the implementation of energetic policies (with respect to renewable sources) has been carried out by performing systems based on a single renewable source. There are such examples in countries all over the world: in Europe, Dewind, Vestas, Enercon, Fronius International GmbH, SMA Technologies AG; Renco SpA, Ansaldo Fuel Cells SpA; in North America-Nyserda, Beacon Power, Magnetek Inc., Sustainable Energy Technology, Logan Energy Corp., IdaTech; Australia-Conergy Pty Ltd, Rainbow Power Company Ltd; and Asia-Nitol Solar, Shenzen Xinhonghua Solar-Energy Co Ltd).

In EU, the implementation of the energetic policies is based upon a legal document, Norm 2001/77/EC regarding the promotion of the electrical energy produced from renewable sources on the Energy Single Market. The objectives of the Norm provide that till 2020, a contribution of 20% of the total energy consumption shall be covered by energy produced

from renewable sources. The monitoring of this Norm implementation is managed by the EU Energy General Directorship which presents periodical reports on the European researching and development stages. Considering these reports, under the conditions of implementing the DER concept (Distributed/Descentralized Energy Resources), it is obvious that the futurer research activities will be based upon the hybrid systems (wind-photovoltaic, wind-biomass, wind-diesel generator) having the target of the energetic security by removing the disadvantages of using a single renewable source.

The consolidation of the objectives proposed by Norm 2001/77/EC and the extension in more geographical areas are possible only by using hybrid systems.

The EU gives a great importance to the improvement of the energy efficiency and to the promotion of the renewable sources. Related to the above mentioned issues, the objectives of the EU are to produce at least 20% ofthe gross energy consumption from renewable sources until 2020 (COM, 2006) and to increase the energy efficiency by 20% until 2020 (EREC, 2011). As far as the energy efficiency is concerned, an EU norm aims at a reduction of 9% of the energy losses until 2020 (EREC, 2008)

2.1. A generic topology of the RES integration

In the context of the international scientific studies related to the development of new alternative solutions for electrical energy production by using renewable energy sources (RES), the aim of this chapter is to contribute to these studies by evaluating and working out the possible concepts for stand-alone and grid connected operating interface of these hybrid systems and the efficient and ecologic technologies to ensure an optimal use of the sources (solar energy, wind energy, hydrogen energy by using fuel cells, hydro-energy, biomass) in industry and residential buildings.The battery and the fuel cells are also meant to be reserve sources (which ensure the additional energy requirements of the consumers and the supply ofboth the residential critical loads and the critical loads of the hybrid system-auxiliary circuits for fuel cell start-up and operating), increasing the safety of the system. The fuel cell integration is provided by using a unidirectional DC/DC converter (to obtain regulated high voltage DC), an inverter and a filter in order to accommodate the DC voltage to the required AC voltage (single phase or three phase). The bidirectional DC/DC converter (double arrow, Fig.1) is used in order to charge/discharge the batteries (placed in order to increase the energy supply security and to improve load dynamics). The unidirectional DC-DC converter prevents the negative current going into the fuel cell stack. Due to the negative current, the cell reversal could occur and damage the fuel cell stack. The ripple current seen by the fuel cell stack due to the switching of the boost converter (unidirectional DC/DC converter) has to be low.

2.2. Three-phase versus single phase

Firstly,the problem of choosing the number of phase for the front end converter is a matter of power. In this case, the three-phase line should be used for a 37kVA power converter.

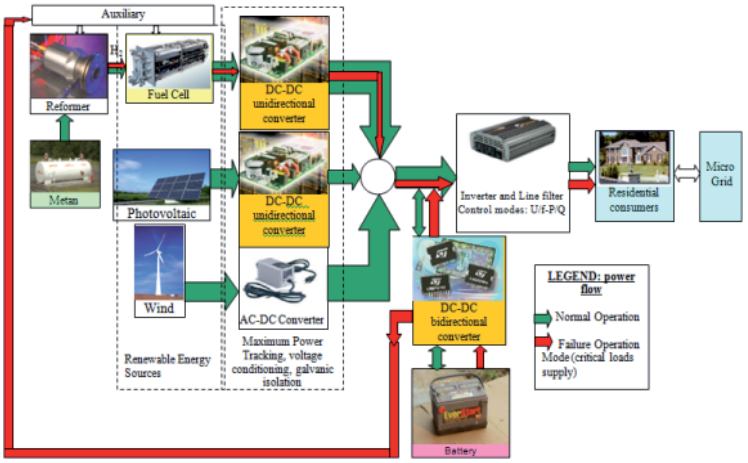


Figure 1. A generic topology of the RES integration

Secondly, in case of using balanced three phase AC loads, the possibility that low frequency components to occur in the fuel cell input current is reduced.

2.3. System description

The proposed way of the efficient integration of RES is illustrated in Fig.1. With this respect only one inverter is used in DC-AC conversion for interfacing the stand-alone or grid-connected consumer (Gaiceanu, *et al.*, 2007b). By its control, the inverter can ensure the efficient operation and the accomplishment of the energy quality requirements related to the harmonics level. The hybrid system can ensure two operation modes: the normal one, and the emergency one (as backup system).

2.3.1. Operation modes

There are four modes of operation:

- ***Direct Supply from Utility***: consists of direct supply of the residential consumers from Utility via the Static Switch;
- ***Precharge Operation***: The DC capacitors in the inverter part of the Power ***Conditioning*** System (PCS) can be precharged from the AC busutility. After the DC capacitors are charged, the inverter can be switched on. As soon as it is running, the inverter by itself will keep the DC capacitors charged to a DC level higher than the No-Load level of the Solid Oxide Fuel Cell (SOFC). During the precharge operation, the residential consumers will still be supplied from Utility;
- ***Normal Operation***: The PCS converts the DC energy from the SOFC into AC and feeds the Utility and the eventual residential consumers.
- ***Island Operation***(failure operation mode): If the Utility goes out of tolerance during normal operation, the PCS will change to island operation. The PCS converts the DC from SOFC and battery and supplies the critical loads.

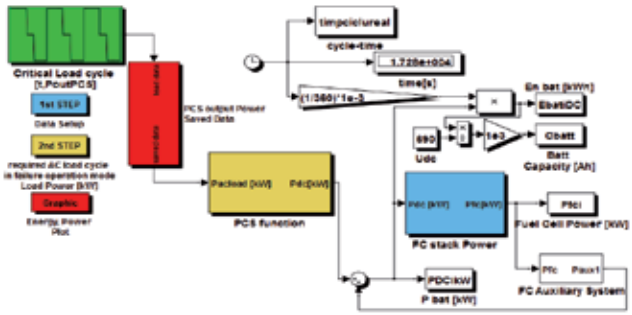


Figure 2. The simplified steady state model of the SOFC Power System for Island Operation Mode

For the sake of simplicity and for chapter length limit reason, only the grid-connected hybrid system with fuel cell generator and battery pack will be investigated.

2.3.1.1 The failure operation mode (Island Operation)

In this operation mode, the power system must assure the power supply for the critical loads (as alarms, auxiliary power systems for fuel cell, reformer and so on, depending on the consumer requirements). In the first stage, the power supply is assured from the battery pack, followed by the SOFC. Therefore, by knowing the critical load power, an adequate Simulink file is designed (Fig.2).A simple and effective energetic model has been considered. This model puts in evidence the inverter power losses at critical load conditions. A repeating table block from the Simulink tool is used in order to implement the load cycle of the residential consummer (critical load cycle, Fig.4). The available acquisition time for the load cycle was at 10 s for each sample, and two days as time interval length. By knowing the critical load power cycle, it is possible to size the required battery pack. Therefore the output inverter power, $P_{out,inv}$, is the same with the critical load power ($P_{acload} = P_{critical\_load}$, Fig.2).

The losses of the Power Conditioning System (PlossPCS=$P_{loss,inv}$) are modelled by using a quadratic function(Metwally, 2005), which is the most used one. The power loss function requires three parameters extracted from the experimental data by the least-squaresmethod.

$$P_{loss,inv} = p_0 + p_1 P_{critical\_load} + p_2 P^2_{critical\_load} \tag{1}$$

The first parameter, p_0, takes into account the load independent losses[W]. The second parameter, p_1, represents the voltage drops in semiconductors as load linear proportional losses. The last term, p_2, includes the magnetic losses [1/W], known as load ohmic losses. The PCS model has been implemented in Simulink (Fig.3) based on the following function:

$$f(u) = p2 * u^2 + p1 * u + p0 \tag{2}$$

in which the coefficients of the approximated function are as follows: p0 = 0.0035*Prated, p1=0.005, p2=0.01/Prated.

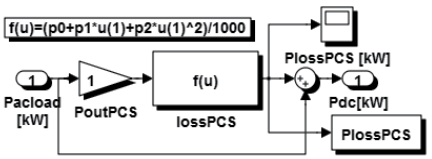


Figure 3. The energetic model of the PCS

In the Fig.3. the energetic component of the PCS block is presented. By knowing the total inverter losses at critical power, $P_{loss,inv}^{tot}$, the corresponding DC power can be obtained:

$$P_{DC} = P_{out,inv} + P_{loss,inv}^{tot} \tag{3}$$

The input and the output inverter powers are related to the inverter efficiency:

$$\eta_{inv} = \frac{P_{out,inv}}{P_{DC}} \tag{4}$$

By taking into account the power requirements of the auxiliary circuits, which are supported only by the battery pack in critical load case during the fuel cell start-up, the corresponding DC power is (Fig.4):

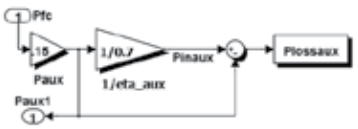


Figure 4. The power losses of the auxiliary power circuits

$$P_{DCi} = P_{DC} + P_{aux} \Rightarrow P_{DCi} = P_{DC} + 0.15 P_{FC}^{critical} \tag{5}$$

The necessary energy of the battery pack is obtained as:

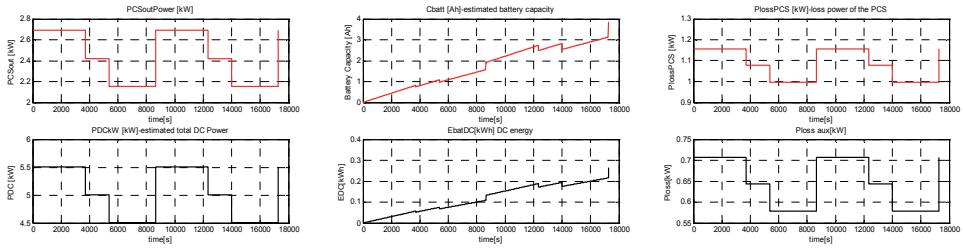
$$W_{batt} = \int_0^t P_{DCi} d\tau \tag{6}$$

The blowers have been considered as main auxiliary loads; the value of $\eta_{aux} = 0.7$ for the equivalent efficiency of the auxiliary power circuits has been considered. In the Fig.4c, the power losses of the auxiliary power circuits have been deducted.

$$\Delta P_{loss,aux} = P_{out,aux}\left(\frac{1}{\eta_{aux}} - 1\right) \tag{7}$$

PCU Matlab/Simulink simulator results

Based on the PCUMatlab/Simulink simulator (Fig.2, Fig. 5a), the required capacity and energy of the battery have been obtained (Fig. 5b)



(a) (b) (c)

Figure 5. (a) The load power PCSoutPower [kW], the estimated total DC power PDC[kW], including the auxiliary power loss; (b) the required capacity and energy of the battery, respectively; (c) the losses of the PCS and the auxiliary power losses.

2.3.1.2 The Normal operation mode

A. The Fuel Cell Power Conditioning System

The fuel cell power conditioning system consists of fuel cell stack and DC power converter.The fuel cell is an electrochemical device which produces DC power directly, without any intermediate stage. It has high power density and zero emission of green house gases. Fuel cell stacks were connected in series/parallel combination to achieve the desired rating. The main issue for the fuel cell power converter design is the fuel cell current ripple reduction. The secondary issue is to maintain a constant DC bus voltage. The former is solved by introducing an internal current loop in the DC/DC power converter control. The latter design requirement is solved by DC voltage control.

A.1 The Fuel cell stack Matlab/Simulink based model

The polarization curve of the SOFC is based on Tafel equation. The output voltage of the SOFC is built taking into account the Nernst instantaneous voltage equation $E_0 + a\ln\left(P_{H_2}P_{O_2}^{1/2}\right)$, the activation overvoltage $b\ln(i)$, the voltage variation due to the mass transport losses $c\ln\left(1-\frac{i}{i_{lim}}\right)$ and the ohmic voltage drop Ri (Candusso D., *et al*. 2002). The first three terms are multiplied by No, number of series cells, in order to obtain the fuel cell stack mathematical model.

The parameters of the Tafel equation are the load current, the temperature and the pressures of the hydrogen and oxygen. The demanded current of the fuel cell system is limited between $\pm I_{fc}^{\text{limit}}$ at a certain hydrogen flow $q_{H_2}^{in}$ value (Padulles, *et al.*, 2000):

$$\frac{0.8q_{H_2}^{in}}{2K_r} \le I_{fc}^{in} \le \frac{0.9q_{H_2}^{in}}{2K_r} \tag{8}$$

The Simulink model of the FC Power System before starting must be initialized (based on *Ifc_init.m* file, Fig.6) from the Fuel Cell data initialization block (Fig.5). In order to obtain the

demanded current between certain limits, an adequate Matlab function has been created(Fcn).

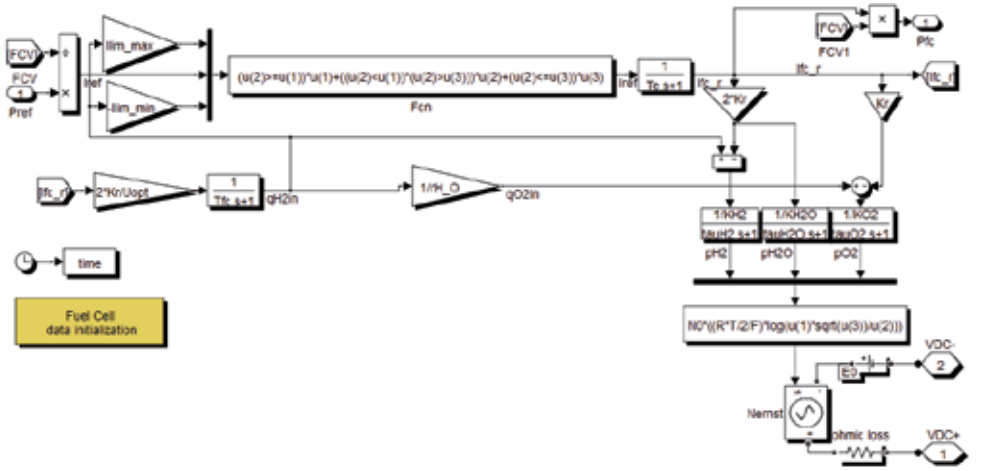


Figure 6. The Simulink model of the solid oxide fuel cell stack

Ifc_init.m

```
prate=80000;  % [W]        Rated power,  Pref=80000;   % [W]        Real power reference

T=1273;    % [K]         Absolute temperature, F=96487;   % [C/mol]      Faraday's constant

R=8314;    % [J/(kmol K)]  Universal gas constant, E0=1.18;   % [V]         Ideal stand potential

N0=384;    %            Number of cells in series inside the stack, Kr=0.995*10^(-5);     %    kmol/(sA),   Constant, Kr = N0/4F

Umax=0.9;  %   Maximum fuel utilization , Umin=0.8;  %   Minimum fuel utilization  Uopt=0.85;  %   Optimal fuel utilization ,

%Value molar constants:

KH2=8.43*10^(-4);     %    kmol/(s atm) - for hydrogen

KH2O=2.81*10^(-4);    %    kmol/(s atm) - for water

KO2=2.52*10^(-3);     %    kmol/(s atm) - for oxygen

tauH2=26.1;          %    s          -  Response time for hydrogen flow

tauH2O=78.3;         %    s          -  Response time for water flow

tauO2=2.91;          %    s          -  Response time for oxygen flow

r=0.126;             %    ohm        -  Ohmic loss

Te=0.8;              %    s          -  Electrical response time

Tfc=5;               %    s          -  Fuel processor response time

rH_O=1.145;          %               -  Ratio of hydrogen to oxygen

PF=1.0;              %               -  Power factor

% demanded current limits of the fuel cell system, Iref, fuel cell

Ilim_max=Umax/2/Kr;

Ilim_min=Umin/2/Kr;
```

Figure 7. SOFC Initial data (Padulles; Zhu)

A.2 Simulation results

By using the implemented Simulink model (Fig.5), the output voltage and the output power have been obtained, as shown in Figure 7.

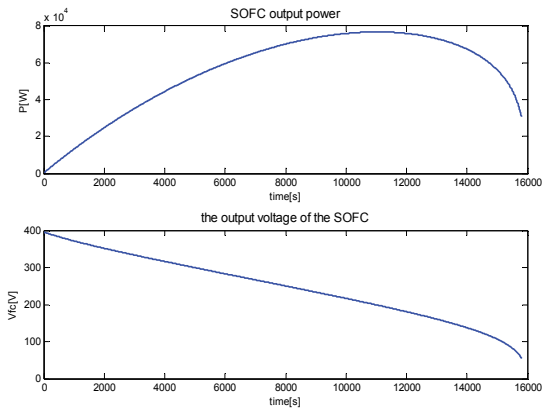


Figure 8. The solid oxide fuel cell characteristics: the power and the output voltage

A.3 Mathematical modeling of the DC-DC power converters for fuel cells and energy storage elements integration: Boost and Buck-Boost power converters

In order to obtain a constant DC voltage, a boost power converter has been taken into consideration (Figure 9), operating in continuous conduction mode (CCM).

The method of the time averaged commutation device is applied to the unitary modeling of the power converters presented in Fig. 9 (Ionescu, 1997).

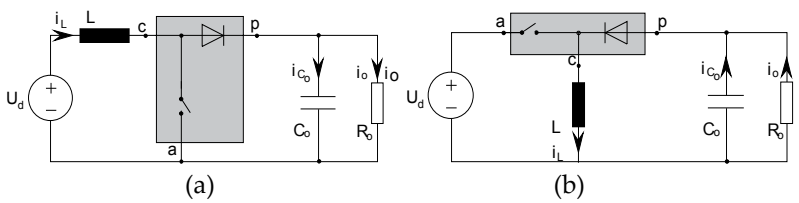


Figure 9. DC-DC non-isolated converters: (a)boost; (b) buck-boost

During the DT_s period, the active device is ON and the passive device is OFF. During the $(1-D)T_s$ period, the active device is OFF and the passive device is ON, while the passive terminal *p* is connected to the common terminal *c*. The duty factor is denoted D and T_s is the switching period. Taking into consideration the above mentioned hypotheses, the following instantaneous currents can be deducted:

$$i_a(t)=\begin{cases} i_a(t) & \text{during } DT_s \\ 0 & \text{during } (1-D)T_s \end{cases}, i_p(t)=\begin{cases} 0 & \text{during } DT_s \\ i_c(t) & \text{during } (1-D)T_s \end{cases} \tag{9}$$

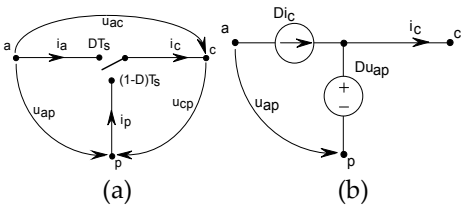


Figure 10. (a) The equivalent three-pole for the commutation device; (b) the equivalent diagram of a time averaged model over a switching period(Ionescu, 1997)

In the similar manner, the specific instantaneous voltages are obtained:

$$u_{cp}(t)=\begin{cases} u_{ap}(t) & \text{during } DT_s \\ 0 & \text{during } (1-D)T_s \end{cases}, u_{ac}(t)=\begin{cases} 0 & \text{during } DT_s \\ u_{ap}(t) & \text{during } (1-D)T_s \end{cases} \tag{10}$$

If averaging is carried out over a period of switching time, equations (9) - (10) will assume the equivalent form of the currents

$$\begin{cases} i_a = D\,i_c \\ i_p = (1-D)\,i_c \end{cases} \tag{11}$$

and of voltages, respectively:

$$\begin{cases} u_{cp} = D\,u_{ap} \\ u_{ac} = (1-D)\,u_{ap} \end{cases} \tag{12}$$

where, for the sake of convenience , values such as i_a are still considered as time averaged values for a period of switching time.

To demonstrate the validity of the time-averaged commutation device model, the mathematical models for DC –DC converters, boost and boost-buck are considered.

A.4 The Boost converter

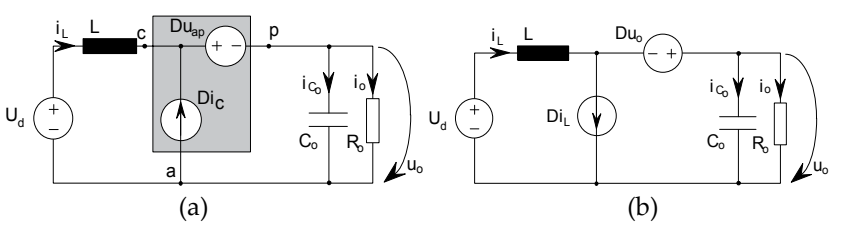


Figure 11. The equivalent structure of the boost converter (Ionescu, 1997)

- From the Fig.11a, the following equivalent relations are obtained:

$$\begin{cases} i_c = -i_L \\ u_{ap} = -u_o \end{cases} \tag{13}$$

- By applying the first Kirchhoff's theorem to the Fig.11b, the first differential equation that characterizes the output voltage dynamic state $\dot{v}_0$ is obtained:

$$(1-D)i_L = C_o\frac{\mathrm{d}u_o}{\mathrm{d}t} + \frac{u_o}{R_o} \text{ or, in the final form } \frac{\mathrm{d}u_o}{\mathrm{d}t} = \frac{1}{C_o}\left[(1-D)i_L - \frac{u_o}{R_o}\right]$$

- By applying the second Kirchhoff's theorem, the second differential equation that characterizes the inductor current dynamic state, $\dot{i}_L$, is obtained:

$$U_d + Du_o = L\frac{\mathrm{d}i_L}{\mathrm{d}t} + u_o \text{ or, in the form } \frac{\mathrm{d}i_L}{\mathrm{d}t} = \frac{1}{L}\left[U_d - (1-D)u_o\right]$$

The commutation mathematical model in state space form will be as following:

$$\begin{bmatrix} \dot{i}_L \\ \dot{u}_0 \end{bmatrix} = \begin{bmatrix} 0 & -(1-D)\frac{1}{L} \\ (1-D)\frac{1}{C_o} & -\frac{1}{RC_o} \end{bmatrix} \begin{bmatrix} i_L \\ u_0 \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} U_d \tag{14}$$

The voltage u_0 is considered controlled output.

- By vanishing the differential terms, the steady-state regime is obtained from the above deducted dynamic state-vector $\dot{x} = \begin{bmatrix} \dot{i}_L & \dot{u}_0 \end{bmatrix}^T$,:

$$\begin{cases} U_0 = U_d\dfrac{1}{1-D} \\ I_o = (1-D)I_L = \dfrac{U_o}{R_o} \end{cases} \tag{15}$$

B. Battery Power Conditioning System

The Battery Power Conditioning Systemconsists of a battery pack and a DC-DC power converter. The NiMH battery produces a variable DC power. The battery pack has as main task to deliver the critical load power (Fig.1).

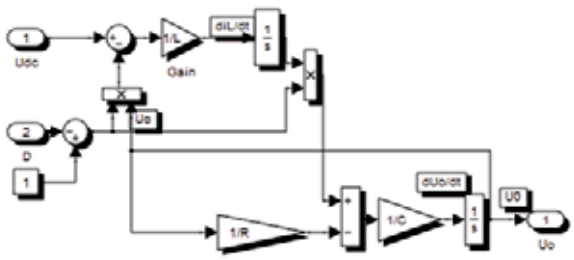


Figure 12. The Simulink diagram of the boost converter

Therefore, individual batteries are connected in series/parallel combination to achieve the desired rating. The Matlab/Simulink battery model from the Mathworks has been used. The main issue for the battery power converter is to charge/discharge battery according to the available flow power. The problem is solved by introducing an internal current loop (Fig.12) in the DC/DC power converter control (Fig.13).

B1. Simulink implementation of the SMC control diagram for DC-DC Boost Power Converter

In (Gulderin Hanifi, 2005) it is shown that the existence condition of the SMC is that the output voltage must be greater than the input one.

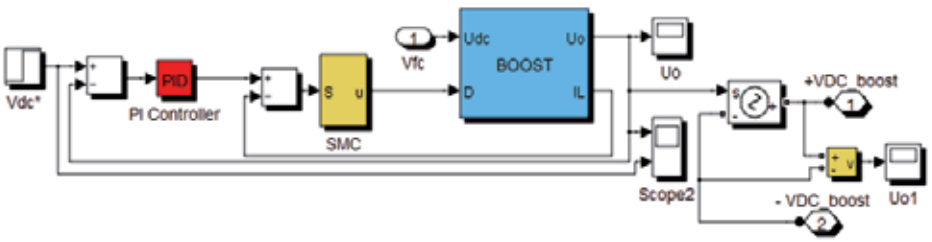


Figure 13. Control of the boost converter: Cascaded DC link voltage loop and current control

The DC-link voltage control is based on the Proportional Integral (PI) controller having k_p=0.00001 and k_i=0.01 as parameters. The circuit parameters of the boost converter are L_{boost}=80*1e-6 [H], C_{boost}=3240*1e-6 [F], R_{boost}=20[Ω].

The current loop is based on the sliding mode control (Fig.14); theMatlab Simulink implementation is shown in Fig.13.

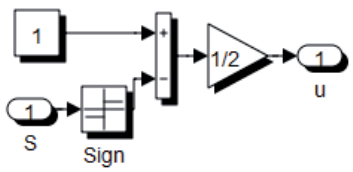


Figure 14. Sliding mode current control

The sliding mode surface S consists of the current error:

$$S = i_L^* - i_L, \tag{16}$$

which vanishes (S=0) in order to force the system to enter the sliding surface. The sliding mode controller has two functions: the control function and the modulator one. Therefore, the output control of the SMC is the duty cycle, D, of the boost power converter.

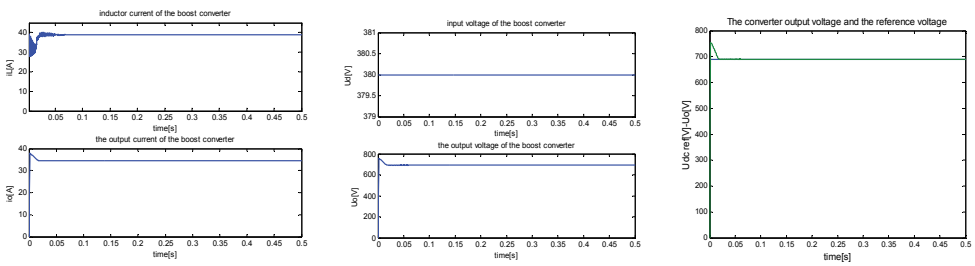
$$D = \frac{1}{2}\left(1 - sign(S)\right) \tag{17}$$

In order to follow the current reference, the output DC voltage must be greater than the following limit:

$$U_0 \geq \frac{3}{2}\sqrt{\left(V_{grid}^{max}\right)^2 + \left(L_{inv}\omega I_{grid}^{max}\right)^2} \tag{18}$$

where the RMS grid voltage is V_{grid}^{max}, the maximal grid current is I_{grid}^{max}, ω is the frequency (rad/s), and L_{inv} is the phase inductance (Candusso D, et al., 2002).

After the simulation, results have confirmed the benefits of SMC control (Fig.15a). The output voltage of the converter reaches and stabilizes at the reference value of 690 V at a time of $2 \cdot 10^{-2}$ s (Fig.15b), a very short time in comparison with other control methods, while the error voltage is zero (Fig.15c).



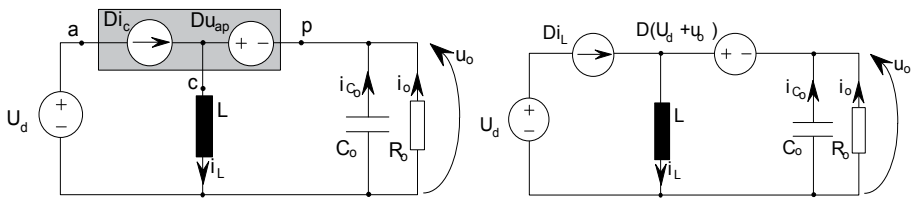
(a) (b) (c)

Figure 15. a) The converter output voltage and the reference voltage; b) Input voltage variation; c) The current and the voltage at converter output

The fundamental purpose of using this converter is to raise the voltage from the fuel cell generator. Thus, the battery pack delivers 380Vdc, being the input voltage of the boost converter; the output voltage must be compatible with that of the three-phase voltage source inverter input, i.e. 690Vdc.

The advantages of this type of control are: stability, robustness and a good dynamics.

B1. The mathematical model of the Buck-Boost power converter



(a) (b)

Figure 16. Equivalent structure of the boost-boost converter (Ionescu, 1997)

- From the Fig.16a, the following equivalent relations are obtained:

$$\begin{cases} i_c = i_L \\ u_{ap} = U_d + u_o \end{cases} \tag{19}$$

- Following the above procedure applied to the boost power converter, the dynamic model of the buck boost converter is deducted :

$$\begin{cases} \dfrac{\mathrm{d}i_L}{\mathrm{d}t} = \dfrac{1}{L}\left[DU_d - (1-D)u_o\right] \\ \dfrac{\mathrm{d}u_o}{\mathrm{d}t} = \dfrac{1}{C_o}\left[(1-D)i_L - \dfrac{u_o}{R_o}\right] \end{cases} \tag{20}$$

and the steady state regime, respectively:

$$\begin{cases} U_0 = U_d \dfrac{D}{1-D} \\ I_o = (1-D)I_L = \dfrac{U_o}{R_o} \end{cases} \tag{21}$$

B.2. The Simulink model of the Buck Boost power converter

Both the buck boost power converter and the current controller have been implemented and simulated in Simulink (Fig.17).

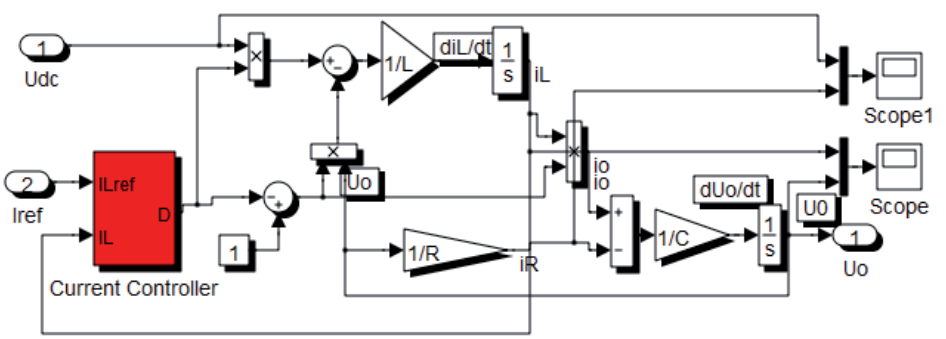


Figure 17. Simulink diagram of the buck-boost converter

B.2.1 The Current Controller

By imposing the inductor current reference (ILref, Fig.18), the current controller will assure the fast reference tracking at the same time with delivering the appropriate duty cycles (D). By introducing an anti-parallel diode for each active power device, a bidirectional buck-boost converter is obtained.

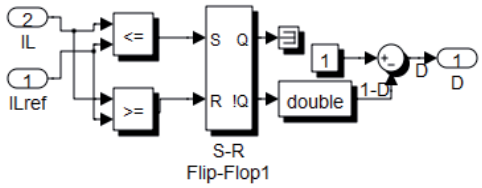
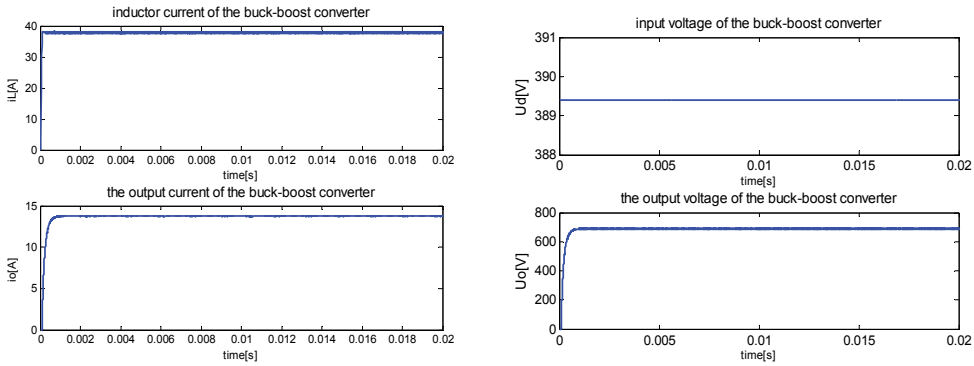


Figure 18. Buck-boost current controller

The buck-boost converter is necessary to connect the battery stack (U_d=U_{dc}) to the power inverter system and it comes into operation when the electrical power demanded by consumers is higher than the electrical power obtained from the fuel cell generator. Another reason for the use of the buck-boost converter is to recharge the batteries from the other available sources. The circuit parameters of the buck boost power converter are L=100*1e-5[H], C=500*1e-8[F], R=50[Ω].



(a) (b)

Figure 19. The simulation results of the buck-boost power converter

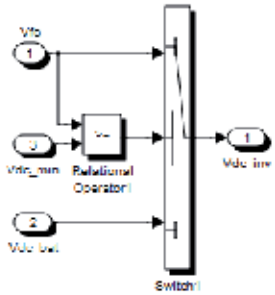


Figure 20. The Power Source Selector

Thanks to the buck boost current controller, the actual inductor current follows the reference current. In the output current a delay of 0.001 s could be found (Fig.19a). The input voltage of the buck-boost converter, U_d, is about 390 Vdc and it is delivered from the battery stack, while the output voltage, U_o, is boosted at 690 Vdc (Fig.19b).

C. Power Source Management (PSM) (Fig.20)

The purpose of the PSM is to assure an adequate DC-link voltage to the power inverter from both power source generators: the solid oxide fuel cell stack and the battery pack.

The final DC link voltage (VDC_inverter, Fig.21) is delivered to the Voltage Source Inverter (VSI) by the Power Source Management block (Fig.20).

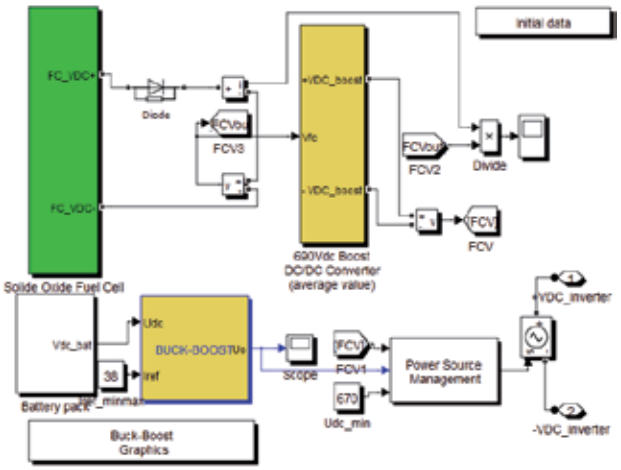


Figure 21. The power sources interconnection

3. Inverter modelling and control

The fundamental types of control can be classified into two categories: current control and voltage control. When the inverter is connected to the network, the network controls the amplitude and frequency of the inverter output and the inverter operates in current control mode. The classical current control can lead to other control methods can be obtained such as active and reactive power control/voltage control. If the network being power injected is not available due to improper network parameters, the inverter will autonomously supply the load; consequently it adequately supplies the alternative voltage in amplitude and frequency and it is not affected by network black outs. In this case, the inverter will control the voltage. The 50 Hz frequency is assured by a phase-locked loop (PLL) control. The grid converter is a full-bridge IGBT transistor-based converter and it normally operates in inverter mode such that the energy is transferred from the hybrid source to the utility grid and/or to the load. When the system is operating in **grid-connected mode**, the PLL tracks the grid voltage to ensure synchronization; but when the system enters in **islanding mode of operation**, the VSI can no longer track the grid characteristics. As seen in Fig. 22, the PLL for the VSI changes the frequency which is sent to the pure integrator for angle calculation by switching between the frequency from the filter and that from another fixed reference. In the islanding mode of operation the VSI needs to have an external frequency reference provided, ω_{fixed} (Fig.22). The PLL for the VSI is the main catalyst for the re-synchronization and re-closure of the system to the Utility once disturbances have passed. The frequency from the filter is used during the grid-connected mode.

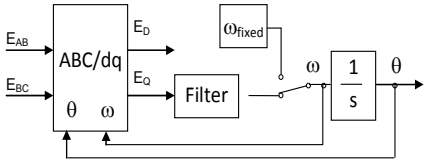


Figure 22. VSI PLL showing switched reference frequency

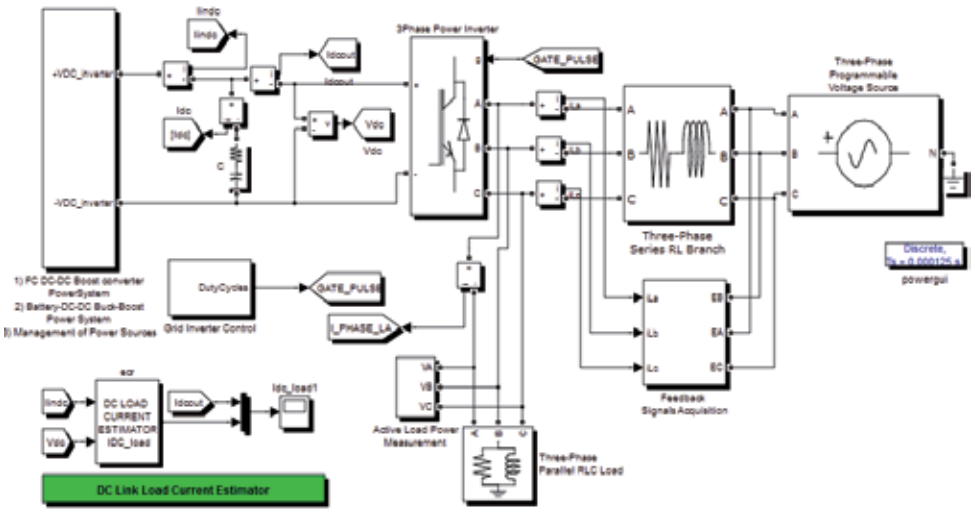


Figure 23. The Simulink model of the Grid Power Inverter for Renewable Energy Sources Integration with DC link Load Current Estimator

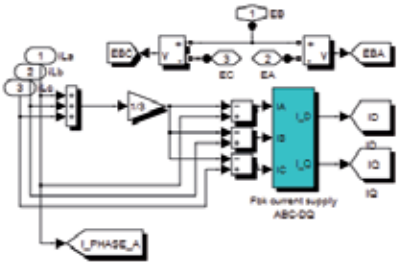
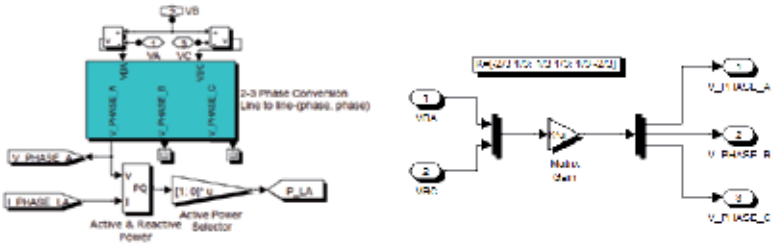


Figure 24. Feedback Signals Acquisition measurement block



(a) (b)

Figure 25. (a)Active Load Power Calculation block; (b) 2/3 phase transformation block

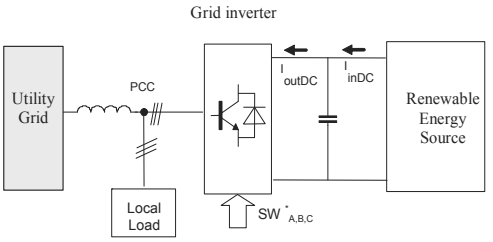


Figure 26. Block diagram of RES, Grid inverter, Local Load and grid interconnection

The Grid Power Inverter for Renewable Energy Sources Integration is of 37kVA and delivers the power to the grid (simulated as three-phase programmable voltage source in Fig. 23) and the necessary power to the consumers (simulated as three-phase parallel RLC load in Fig. 23). There is an adequate boost inductor (three-phase series RL branch, Fig. 23) between the grid and the inverter. In order to calculate the *dq* components of the grid current, (ID, IQ), the Feedback Signals Acquisition block is used (Fig. 24). Through the implemented Simulink blocks (Figs.25a, 25b), the active power of the load is known.

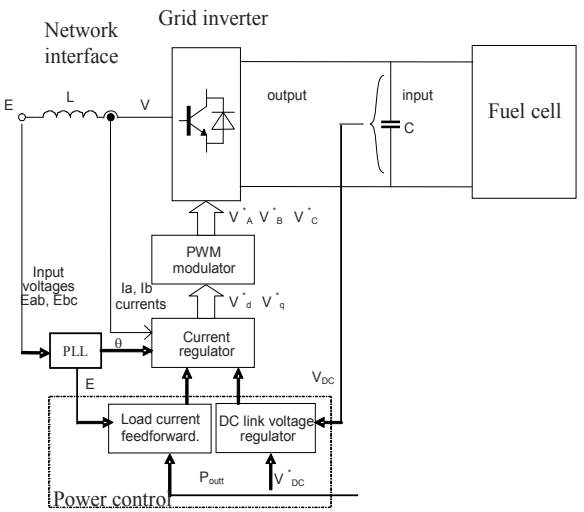


Figure 27. Proposed control system for grid inverter

3.1. The grid inverter control

The grid inverter control block delivers the corresponding duty-cycles to the Power Inverter (Gate_Pulses in Fig.23 or SW^*_{ABC} in Fig. 26). To achieve full control of the utility-grid current, the DC-link voltage must be boosted to a level higher than the amplitude of the grid line-line voltage. The power flow of the grid side inverter is controlled in order to keep the DC-link voltage constant. The structure of the DC/AC converter control system is shown in Fig. 27. The control structure of the power inverter is of vector control type and it uses the power balance concept (Sul and Lipo, 1990). Therefore, the load current feedforward component

was introduced in order to increase the dynamic response of the bus voltage to load changes.

On the basis of the DC voltage reference V^*_{DC}, DC voltage feedback signal (V_{DC}), AC input voltages (E_{ab} and E_{bc}), current feedback signals (I_a, I_b), and the load power signal (got through a load power estimator) (Gaiceanu, 2004a), the Digital Signal Processor-based software operates the control of the power inverter (DC link voltage and current loops) system and generates the firing gate signals to the PWM modulator (Fig.27). The grid connected PWM inverter supplies currents into the utility line by maintaining the system power balance. By controlling the power flow in power conditioning system, the unidirectional DC-link voltage can be kept at a constant value. Using the synchronous rotating frame the active power is controlled independently by the*q*-axis current whereas the reactive power can be controlled by the *d*-axis current.

The control of the grid inverter is based on the minor current loop in a synchronous rotating-frame with a feedforward load current component added in the reference, completed with the DC voltage control loop in a cascaded manner.The outer loop controller consists of two parts: the phase-locked loop (PLL) and the DC link voltage controller. The former, the PLL, is used to extract the fundamental frequency component of the grid voltages and it also generates the corresponding quadrature signals in *d-q* synchronous reference frame, E_d-E_q, which are necessary to calculate the active and reactive power of the grid. The latter monitors the power control loop. The power control of the PWM inverter, is based on the power detection feedforward control loop and the DC-voltage feedback control loop (Fig.27). The main task of the voltage controller is to maintain the DC link voltage to a certain value. Another task is to control the grid converter power flow. The task of the DC link voltage and of the current regulation has been accomplished by means of the Proportional-Integral (PI) controller type, because of its good steady-state and dynamic behavior with the power inverter. It is important to underline that the PI controller performances are parameters sensitive, because of its design procedure, based on the DC bus capacitor and inductor values. However, in these specific applications, the system parameters values are known with reasonable accuracy. The design of the linear control systems can be carried out in either the time or the frequency domain. The relative stability is measured in terms of **gain margin,** and**phase margin.** These are typical frequency-domain specifications and should be used in conjunction with such tools as Bode plot.

The transfer function of the PI controller (Gaiceanu, 2007b) has the form:

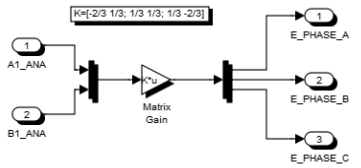
$$C_{PIc}(s) = K_{pc}\left(1 + \frac{1}{T_{ic}s}\right) \tag{22}$$

The calculation of the PI controller coefficients, K_{pc} (proportional gain) and T_{ic} (integral time), is done imposing the phase margin ϕ_{mc} (in radian) and the bandwidth,ω_c, (in radian per second). Imposing these two conditions, the following relations for K_{pc} and T_{ic} are obtained(Gaiceanu, 2007b):

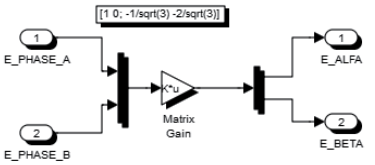
$$\begin{cases} T_{ic} = \dfrac{1}{\omega_c \cdot tan\left(-\dfrac{\pi}{2} - \phi_{mc}\right)} \\ K_{pc} = \dfrac{-T_{ic} \cdot \omega_c^2 \cdot L}{\sqrt{1 + \left(T_{ic} \cdot \omega_c\right)^2}} \end{cases} \tag{23}$$

3.2. The Phase Locked Loop (PLL)

A phase locked loop (PLL) ensures the synchronization of the reference frame with the source phase voltages by maintaining their *d* component at zero (E_d=0) through a PI controller; the grid frequency is delivered by knowing the line-line grid voltages (EBA, EBC), as in Figs.28, 29.



(a) line-line to three phase



(b) (A,B,C)-(alfa, beta)

Figure 28. The transformation of the coordinates

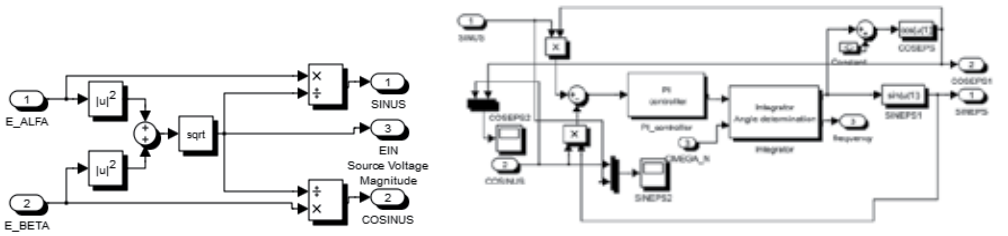


Figure 29. (a) Calculus of the required PLL's input trigonometric functions (b) PLL

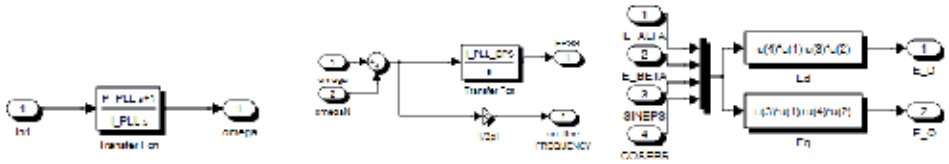


Figure 30. (a) The PI regulator of the PLL (b) The integrator for angle calculation (c) *dq* grid voltage components: ED, EQ

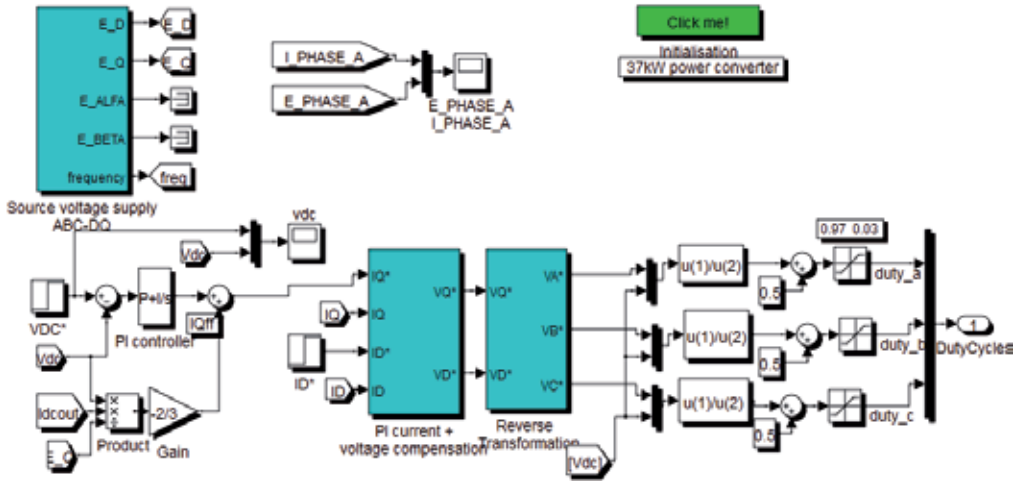


Figure 31. The Simulink structure of the DC/AC converter control system

3.3. The current controllers

By using a decoupling of the nonlinear terms, the cross coupling (due to boost input inductance) between the *d* and *q* axes was compensated. To decouple current loops, the proper utility voltage components have been added (Gaiceanu, 2004b) (Fig 32).

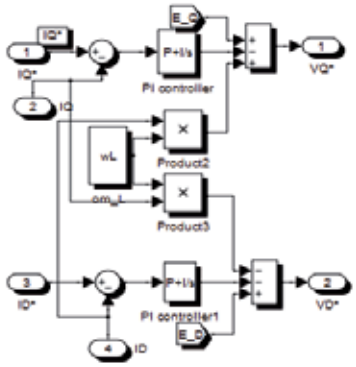
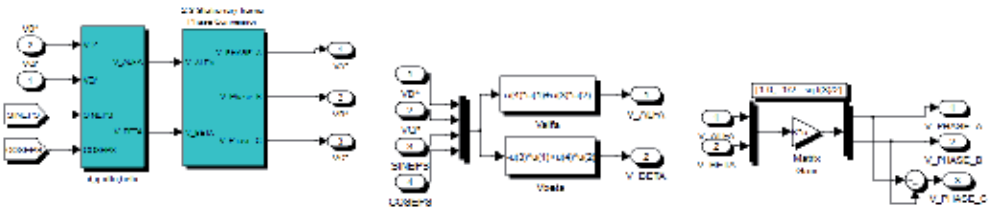


Figure 32. Voltage decoupling control

Fig.33 shows the Simulink implementation of the reverse transformation from synchronous reference frame (d,q) tofixed reference frame (A,B,C) through the (alfa,beta) transformations.



(a) DQ-ABC (b) DQ-(alfa, beta) (c) (alfa,beta)-(A,B,C)

Figure 33. Reverse voltage transformation

4. The DClink current estimator

The load power (P_{load}=P_{out}) is calculated from the load inverter terminals. Another method is to estimate the load power from the DC link, indirectly, through a first or second order DC load current estimator (Gaiceanu, 2004a). The power feedforward control (Uhrin, 1994) allows the calculus of the input current reference based on the generated power, and it satisfies the power balance in a feedforward manner. By using the load feedforward control, the input reference of the current is changed with load, thus it is obtained a better transient response. The increase in the power response of the DC-AC inverter leads to the possibility of reduction the size of the DC link capacitor by maintaining the stability of thesystem.

The block diagram of the second degree estimator is presented in the Fig. 34, where the input needs the measure of the DC link voltage $V_{dc}(p)$ and the calculus of the input ac load current component I_q. The output of the estimator is the estimated DC link load power $\hat{P}_{dcout}(p)$.

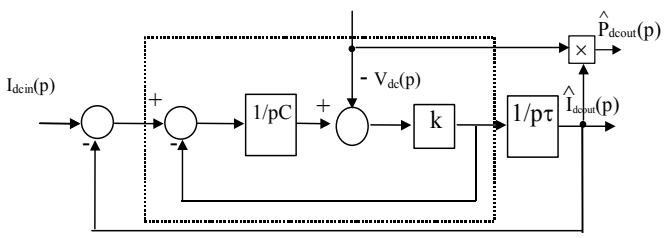


Figure 34. The second order dc link load power estimator

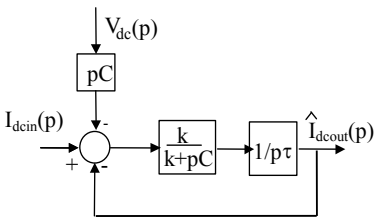


Figure 35. The redrawing estimator

The estimator (Fig. 34), after some manipulations (Fig.35), gets the form presented in the Fig. 36.

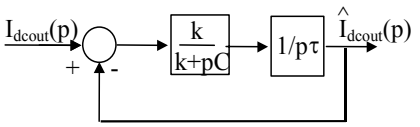


Figure 36. The simplified diagram of the DC load

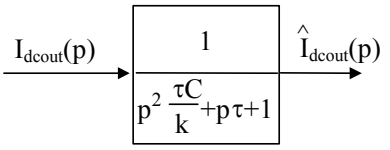


Figure 37. The final second order estimator

Using Laplace transform the DC link voltage equation gets the form

$$pCV_{dc}(p) = I_{dcin}(p) - I_{dcout}(p) \tag{24}$$

or:

$$I_{dcin}(p) - pCV_{dc}(p) = I_{dcout}(p) \tag{25}$$

This means that the block diagram from Fig.35 can be redrawing as in Fig. 36.

4.1. Calculus of the estimator parameters

The problem consists of the calculation of the parameters k and τ such that the error between the estimated DC load current $\hat{I}_{dcout}(p)$and the actual DC load current $I_{dcout}(p)$ to be insignificant. The closed loop transfer function of the estimator, (Fig.37), derived from Fig.36, is given by:

$$G(p) = \frac{\hat{I}_{dcout}(p)}{I_{dcout}(p)} = \frac{1}{p^2\frac{\tau C}{k} + p\tau + 1} \tag{26}$$

Considering a step variation for the $I_{dcout}(p)$, by setting:

$$I_{dcout}(p) = \frac{I_{dcout}}{p} \tag{27}$$

the estimated DC load current gets the form

$$\hat{I}_{dcout}(p) = I_{dcout}\frac{1}{p(p^2\frac{\tau C}{k} + p\tau + 1)} \tag{28}$$

The usual form of the equation (28) is given by

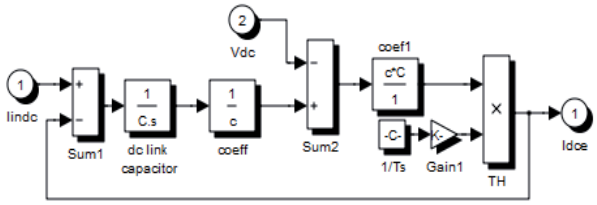


Figure 38. Simulink implementation of the DC link load current estimator

$$\hat{I}_{dcout}(p) = I_{dcout}\frac{1}{p(T_0^2p^2 + 2\xi T_0 p + 1)} \tag{29}$$

where the damping factor is

$$\xi = \frac{\tau \cdot \omega_0}{2} \tag{30}$$

and the pulsation factor

$$\omega_0^2 = \frac{1}{T_0^2} = \frac{\tau}{k \cdot C} \tag{31}$$

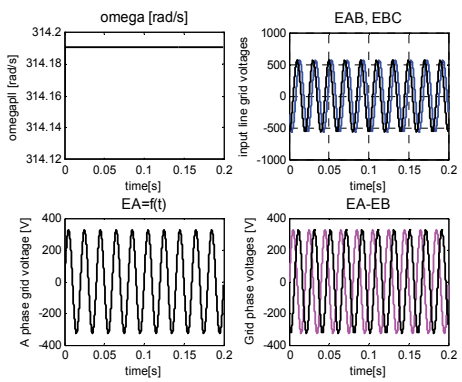


Figure 39. The frequency of the line voltage. The acquisition of the grid line voltages (EAB,EBC) and the phase transformation (EA, EB).

The parameters k and τ are chosen such that the response $\hat{I}_{dcout}(p)$ to have an acceptable overshoot

$$\sigma = e^{-\frac{\pi\xi}{\sqrt{1-\xi^2}}}, \tag{32}$$

a small step time response

$$t_a = \frac{ln(0.05\sqrt{1-\xi^2})}{-\xi \cdot \omega_0}, \tag{33}$$

and a minimum output noise.

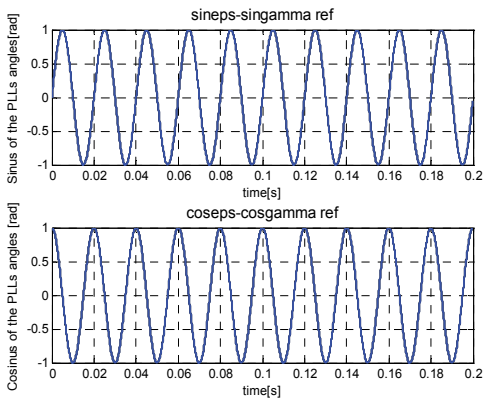


Figure 40. The input and the output signals of the PLL circuit

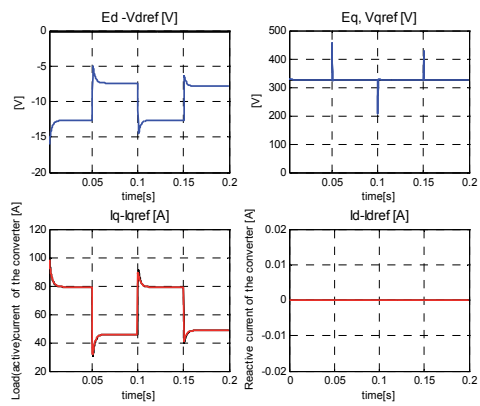


Figure 41. The comparison of the grid voltage and the converter voltage (Ed,Vdref), (Eq, Vqref) and the performances of the current controllers

An advantage of the estimated method is that there is no ripple presence in the feed-forward reference current of the source side. The small reference current ripple is delivered from the output of the DC link voltage controller.

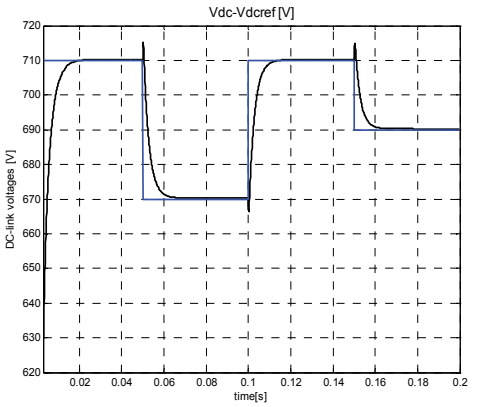


Figure 42. DC link voltage reference, Actual DC link voltage.

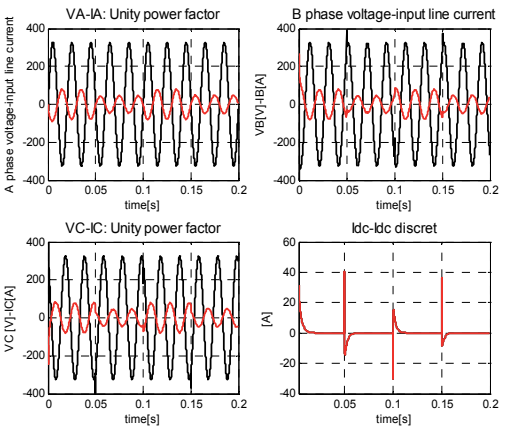


Figure 43. Waveforms showing the unity power factor operation: A, B, C phase grid voltages and the corresponding IA, IB, IC line currents. Simulation results. Idc- the current through the DC link capacitor

4.2. Simulation results

Fig. 41 shows the comparison of the grid voltage and the converter voltage (Ed,Vdref): the Ed component is 0 and the voltage Vdref is calculated as Vdref =om*L_{in}*I_{qN}+E_d in steady state regime (Gaiceanu, 2007a). The voltages Eqref and Vqref have the same value of 326.55[V].

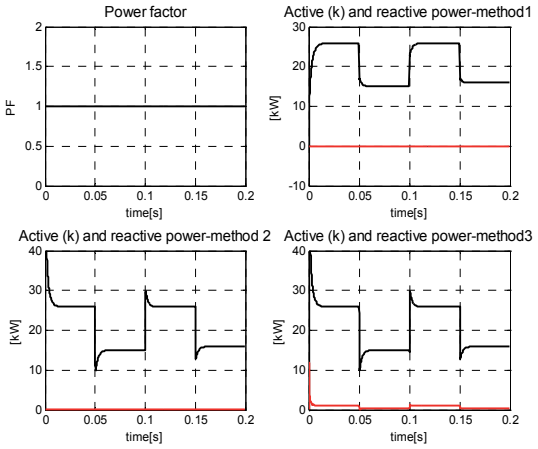


Figure 44. The unitary power factor operation and comparison results of the three methods of active and reactive power deduction

The reference and actual *d* axis current waveforms (Id-Idref) are shown in Fig.41 proving the cancellation of the reactive power.

```
%First Method
% power factor
   fi=atan(Id/Iq);
   PF(i)=cos(fi);
% Active Power [kW]
   Active_power1(i)=Ein*Iqff*cos(fi)/1000;
% Reactive Power
   Reactive_power1(i)=Ein*Iqff*sin(fi)/1000;
% Second Method
  Active_power2(i)=3*(Ed*Id+Eq*Iq)/2/1000;
  Reactive_power2(i)=3*(Eq*Id-Ed*Iq)/2/1000;
% Third Method
 Active_power3(i)=(Valfa*Ialfa+Vbeta*Ibeta)/1000;
 Reactive_power3(i)=(Vbeta*Ialfa-Valfa*Ibeta)/1000;
```

The DC link voltage step response was obtained by using a DC link voltage test generator (Fig.42) under a load current variation between [0.65, 1.15]×I_N (Fig.41), I_N being the rated value of the line current.

The performances of the active current inverter control are shown in Fig.41. The actual active current, I_q, accurately follows the reference I_{qref} (Fig.41). In Fig. 42 the performances of DC-link voltage controllers are shown. The trace of the A phase of the line current is in phase with A phase of the grid voltage, which clearly demonstrates the unity power factor

operation (Fig. 43). Comparative waveforms showing unity power factor operation during regeneration obtained from DC-AC power converter are shown in Fig. 43. For all three methods of active and reactive power deduction (Fig.44) the steady state values are the same, however the first method is more accurate in transient regime (Fig.44) (Gaiceanu, 2004b).

The 2nd degree DC link current estimator was implemented for a 37kVA power inverter. The dynamic performances of the DC load current estimator are presented (Gaiceanu, 2004a).

By an adequate choice of the estimator parameters an acceptable step response can be obtained (Fig.45).

Through simulation (Figs. 45-46) the real and the estimated DC link currents are obtained.

The power semiconductor active devices operate with a switching time T_s=125μs, and a 2μs dead time. The converter specifications are given as follows: Supply voltage (line-to-line): 400V; Main frequency: 50Hz, Line current: 69A, Line inductance: 0.5 mH, DC bus capacitor: 1000 μF, Ambient temperature 40^0C,DC voltage reference: 690V.

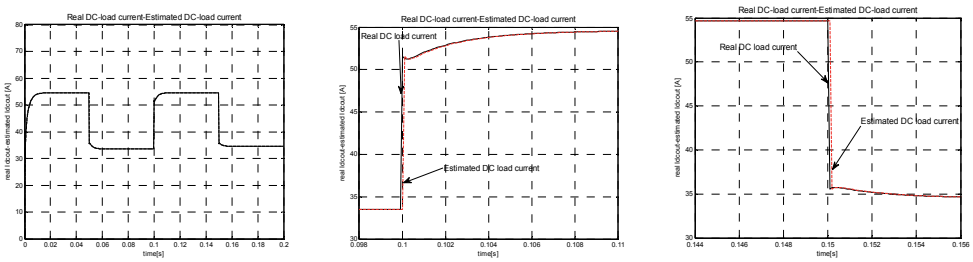


Figure 45. Simulation results. The real DC load current I_{dcout}, the estimated DC link current $\hat{I}_{dcout}$.

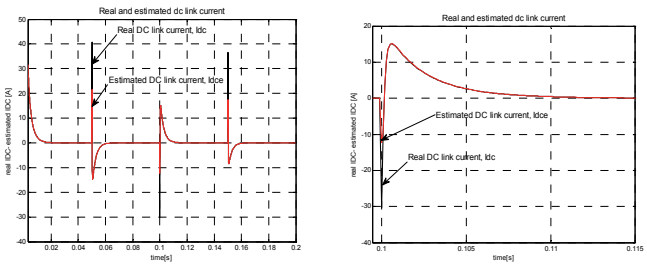


Figure 46. Simulation results. The real DC link current I_{dc}, the estimated DC link current $\hat{I}_{dc}$

5. dSpace implementation

The PI Current Control in Synchronously Reference Frame is shown in Fig.47. The current regulators have two tasks: the error cancelling, and the modulation (the appropriate switching states are provided).

For an adequate tuning of the current regulators, the actual load current, iA, accurately follows its reference i*A (Fig. 49), despite of an inappropriate tuning of the current controllers (Fig.48).

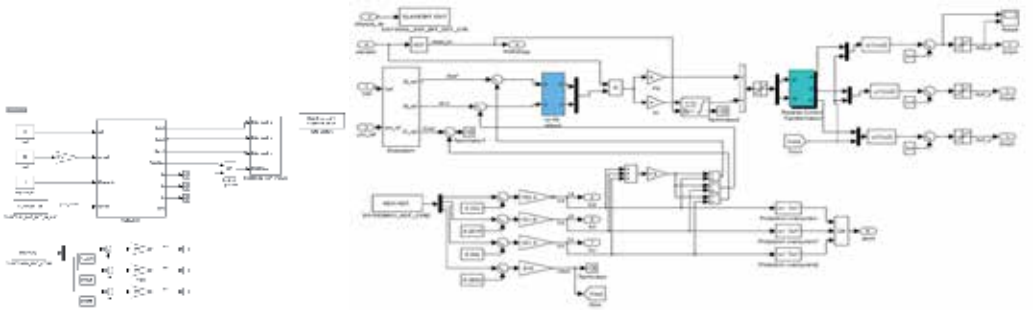


Figure 47. Real time implementation of the current control by using dSpace 1103 platform

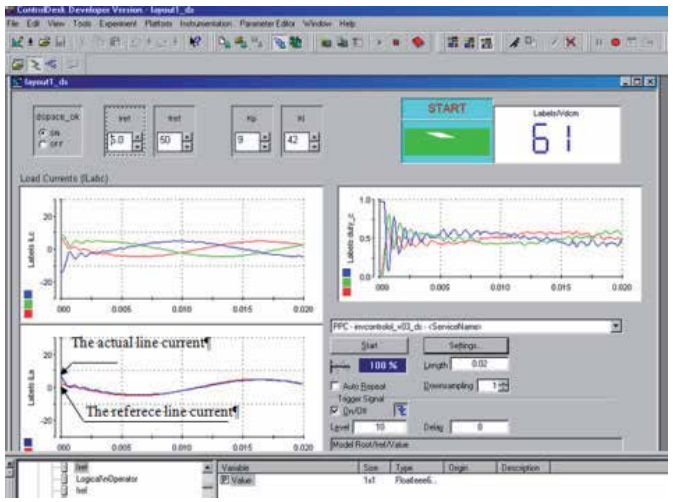


Figure 48. The three phase load currents, the corresponding duty cycles, the actual and the reference line currents for the inaccurate tuning of the current regulator parameters: Kp=9, Ki=42

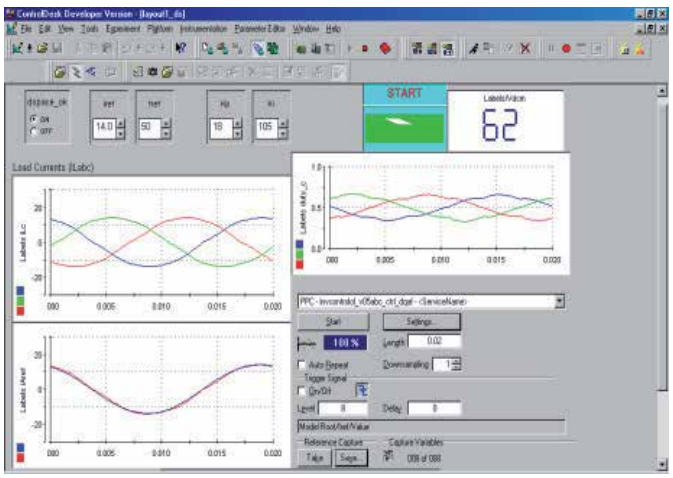


Figure 49. The three phase load currents, the corresponding duty cycles, the actual and the reference line currents for the accurate tuning of the current regulator parameters: Kp=18, Ki=105

6. Conclusions

The main outcomes of the chapter:

- The chapter pursuits to increase the awareness of public regarding the renewable energy technologies through the open access, and to determine the researchers to implement renewable energy projects.
- The chapter will contribute the promotion of RES through the formation of experts so that these experts can later carry out RES projects with outstanding results.

The implicit longer term outcomes are related to:

1. Accurate models for fuel cells power systems.
2. New design of the adequate controllers for integrated systems, which will enable the efficient operation of such power inverters connected to the grid, with high stability in service and power quality.
3. The rapid prototyping through dSpace real time platform can prove very useful in medium and longer term for further modelling/investigation/development of similar systems.

The chapter will also bring contributions to the development of the theoretical knowledge if the following aspects are taken into account: the complexity of the issue, its interdisciplinary, the performance of an experimental model and the necessary theoretical knowledge of the interface solutions for the renewable system, in particular for fuel cells.

Through a proper control sinusoidal input current, a nearly unity power factor (0,998), bi-directional power flow, small (up to 5%) ripple in the DC-link voltage in any operated conditions, disturbance compensation capability, fast control response and high quality balanced three-phase output voltages were obtained. By using the load feed-forward component the input reference of the current is changed with load so that a better transient response is obtained. The proposed control was successfully implemented by the author on quasi direct AC-AC power converter (Gaiceanu M., 2004b) and based on the Matlab/Simulink software the simulation test has been performed for the modified topology of the grid power inverter. The experimental results (Figs. 48, 49) have been obtained by using dSpace platform (Fig.47). The second-degree DC load current estimator for DC-AC power converter system is developed in this chapter. Since the DC-AC power converter control by means of pulse-width modulation (PWM) is based on the power balance concept, its load power should be known. In order to overcome the measuring solution with well-known disadvantages, the load power can be estimated from the DC side by using the DC load current estimator. Thus, it is mandatory to have the information regarding the DC load current. The DC voltage regulation with good dynamic response is achieved even if DC capacitance is substantially reduced. This implies also the good accuracy of the DC link load current estimation.

Author details

Marian Gaiceanu
Dunarea de Jos University of Galati, Romania

7. References

Abou El-Maaty Metwally (2005). Modelling and Simulation of a Photovoltaic Fuel Cell Hybrid System) Kassel, Germany

Candusso D. , Valero I.& Walter A. (2002). Modelling, control and simulation of a fuel cell based power supply system with energy management,IECON 2002 28th Annual Conference , pp.1294-1299

COM(2006) Action Plan for Energy Efficiency: Realising the Potential, Available fromhttp://ec.europa.eu/energy/action_plan_energy_efficiency/doc/com_2006_0545_en.pdf

EREC, Renewable Energy Technology Roadmap (2008), Available from http://www.erec.org/fileadmin/erec_docs/Documents/Publications/Renewable_Energy_Technology_Roadmap.pdf, pp2

EREC, the European Renewable Energy Council (2011). Mapping Renewable Energy Pathways towards 2020, Available from http://www.eufores.org/fileadmin/eufores/Projects/REPAP_2020/EREC-roadmap-V4.pdf

Gaiceanu M (2004b). AC-AC Converter System for AC Drives, *IEE Conference Publication Journal*, British Library, London, Publisher: Institution of Electrical Engineers, Vol. 2, no. 498, Printed in Great Britain by WRIGHTSONS, ISSN 0537-9989, pp 724-729

Gaiceanu M. (2004a). A new load power estimator for quasi-sinusoidal ac-ac converter system," *Proceedings of the 9th International Conference on Optimization of Electrical and Electronic Equipments (OPTIM 2004),* Vol. II: Power Electronics, Electrical Machines & Drives, ISBN 973-635-287-0, Brasov, May 20-21, pp.189-195, 2004

Gaiceanu M. (2007a) Inverter Control for Three-Phase Grid Connected Fuel Cell Power System, The *5th International IEEE Conference CPE 2007*, *Compatibility in Power Electronics Conference*, May 29- June 1, 2007, Gdansk, Poland, Power Electronics, 2007 Compatibility in, Conf Proceedings IEEE Product No.: EX1712, ISBN: 1-4244-1054-1

Gaiceanu, M.& Fetecau G. (2007b). Grid connected Wind turbine-Fuel Cell Power System having Power Quality Issues, EPQU'07 Barcelona, pp.7-13, 2007. ISBN 978-84-690-9441-9

Gulderin Hanifi (2005), Sliding Mode Control of DC-DC boost converter, Journal of Applied Sciences 5 (3): 588-592

Ionescu Fl. et al (1997). Electronica de putere. Modelare si simulare, Editura Tehnica

Padulles J., Ault G.W. &McDonald J.R. (2000). An integrated SOFC plant dynamic model for power systems simulation, J. Power Sources 86 495_500

Sul, S.K., & Lipo T.A. (1990). Design and performance of a high-frequency link induction motor drive operating at unity power factor," *IEEE Trans. Ind. Applicat.*, vol.26, no.3, pp. 434-440, May/June

Uhrin, R.& Profumo F. (1994). Performance comparison of output power estimators used in AC/DC/AC converters, *Industrial Electronics, Control and Instrumentation,* IECON '94., 20th International Conference on, Volume 1, 5-9 Sept. 1994 Page(s):344 - 348 vol.1, 1994

Zhu Y. &Tomsovic K. (2002). Development of models for analyzing the load-following performance of microturbines and fuel cells, Electric Power Systems Research 62 (2002) 1_/11

Chapter 11

Model-Based Simulation of an Intelligent Microprocessor-Based Standalone Solar Tracking System

C.S. Chin

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46458

1. Introduction

Renewable energy resources will be an increasingly important part of power generation in the new millennium. Besides assisting in the reduction of the emission of greenhouse gases, they add the much- needed flexibility to the energy resource mix by decreasing the dependence on fossil fuels [1]. Among the renewable-energy resources, solar energy is the most essential and prerequisite resource of sustainable energy because of its ubiquity, abundance, and sustainability. Regardless of the intermittency of sunlight, solar energy is widely available and completely free of cost. Recently, photovoltaic (PV) system is well recognized and widely utilized to convert the solar energy for electric power applications. It can generate direct current (DC) electricity without environmental impact and emission by way of solar radiation. The DC power is converted to AC power with an inverter, to power local loads or fed back to the utility [2]. Being a semiconductor device, the PV systems are suitable for most operation at lower maintenance costs.

The PV applications could be grouped according to the scheme of interaction with utility grid: grid connected, stand alone, and hybrid. PV systems consist of a PV generator (cell, module, and array), energy storage devices (such as batteries), AC and DC consumers and elements for power conditioning. The most common method uses the PV cells in the grid network. However, to understand the performance and to maximize the efficiency of the irradiation of the PV cells, the standalone PV cells have spurred some interest, especially, in the area of the solar tracker system.

Over the years, test and researchers had proven that development of smart solar tracker maximizes the energy generation. In this competitive world of advanced scientific discoveries, the introductions of automated systems improve existing power generation

methods. Before the introduction of solar tracking methods, fixed solar panels were positioned within a reasonable tilted direction based on the location. The tilt angle depending on whether a slight winter or summer bias is preferred in the system. The PV systems would face "true north" in the northern hemisphere and "true south" in the southern hemisphere. Solar tracking is best achieved when the tilt angle of the tracking PV systems is synchronized with the seasonal changes of the sun's altitude. Several methods of sun tracking systems have been surveyed and evaluated to keep the PV cells perpendicular to the sun beam. An ideal tracker would allow the PV cells to point towards the sun, compensating for both changes in the altitude angle of the sun (throughout the day), latitudinal offset of the sun (during seasonal changes) and changes in azimuth angle. In the light of this, two main types of sun trackers exist: passive (mechanical) and active (electrical) trackers. The detailed literatures review can be found in [3].

One class of the passive solar trackers is the fixed solar panel. It is placed horizontally on the fixed ground and face upwards to the sky. But most of the passive solar trackers are based on manual adjustment of the panel[4], thermal expansion of a shape memory alloy[5]or two bimetallic strips made of aluminum and steel[6]. Usually this kind of tracker is composed of a couple of actuators working against each other, which are, by equal illumination, balanced. By differential illumination of actuators, unbalanced forces are used for orientation of the apparatus in such direction where equal illumination of actuators and balance of forces is restored. Another passive tracking technology is based on the mass imbalance [6] between both ends of the panel. This kind of trackers does not use any kind of electronic control or motor. Two identical cylindrical tubes are filled with a fluid under partial pressure. The sun heats the fluid causing evaporation and transfer from one cylinder to another, which creates the mass imbalance. Passive solar trackers, compared to active trackers, are less complex but works in low efficiency. Although passive trackers are often less expensive, they have not yet been widely accepted by consumers.

On the other hands, major active trackers can be categorized as a microprocessor based, computer-controlled date and time based, auxiliary bifacial solar cell based and a combination of these three systems. In the microprocessor based solar tracker systems [7-11], a controller is connected to DC motors. Once the location is selected, the azimuth elevation range is determined, and the angular steps are calculated. Usually for monitoring the power generation, they also connected this tracking device to a PC by a code written in Assembly or C++ languages. In this solar tracker design, sensors were often used. For example, a photo-resistor [12-13] was put in a dark box with a small holes on the top to detect the illumination, and a light sensor or photosensor called light-dependent resistor (LDR) [11,14] to indicate the intensity of the radiation (that changes its electrical resistance from several thousand Ohms in the dark to only a few hundred Ohms when light falls upon it). The signals were then captured by the microcontroller that provides a signal to the motors to rotate the panel.

Whilst in the auxiliary bifacial solar cell [15] systems, the bifacial solar cell senses and drives the tracker system to the desired position. Auxiliary solar cells (panels) connected directly to a permanent magnet DC motor are fixed to a rotary axle of the tracker and can both sense and provide energy for tracking. In this design, unreliable and expensive components like batteries and driving electronics were completely eliminated. Hence, it is a very simple, reliable solar tracker for space and terrestrial applications. On the other hands, the computer-controlled date and time based system calculates the sun positions with respect to date and time with algorithms and creates signals for the system control via motor to the panel. In the solar tracking system that was designed by [16-17], the required position was calculated in advance and was programmed into Programmable Logic Control (PLC) that in term controls the motor to adjust the panel to maintain position perpendicular to the sun.

In another method that uses the combination of microprocessor with sensor and date/time based system [18-19], the sensors such as pyrheliometers (that measure the direct beam of sun irradiance) and/or light sensors send the signal to the microprocessor. Using the real-time clock (RTC), the tracker computes the position of the sun based on the date/time information of its clock. The data gathered during the day are analyzed, and a new improved set of parameters for the installation errors is computed. These data are used in the next day to compute more accurate positions of the sun, and the cycle continues.

Hence, the main difference between the active trackers is the ability to reduce the pointing error using external sensors, thereby increasing the daily irradiation the solar cells receive and the electric energy that they produced. A comparative study[3] shows that, the power consumption by the tracking device is only 2-3 % of the increased energy. The annual energy available to the two axis tracker was 72% higher than a fixed surface and 30% for single axis East-to-West tracker. However, the two or more axes trackers are more complex and costly to maintain as compared to the single-axis tracker. Furthermore, as Singapore is near to the equinoxes, the sun rises directly in East and set directly West regardless of the latitude, the single axis solar tracker becomes more favorable in this region. Based on the above-mentioned tracking systems, we have suggested that the active based single-axis tracker system is less complex to design and maintain. Additionally, for the system to work solely based on light-sensor tracking technology is not practical due to Singapore's unexpected changes in weather conditions. We have therefore come about with the idea to integrate a time-based tracking technology with the light-sensor tracking technology in the tracker system. With high numbers of high-rise buildings in Singapore, the ideas of having a wall-mounted design for each household usage become an attractive option.

In the new solar tracking system installed with sensor feedback, and real-time clock control was capable of performing both automatic and preset mode of operation. The system's ability to switch between the modes proves to be an important feature. The position and "status" of the sun is detected by two light-dependent resistor (LDR) sensors that are located at the both ends of the surface of the photovoltaic panel. In the automatic modes, the resultant signals from the sensors are fed into an electronic control system that operates a low- speed DC motor to rotate the panel via a speed-reduction system. In this mode, the Sun

is not constantly tracked to prevent energy consumed by the motor, and the system will be in 'sleep' mode when the night falls. In the preset mode, the solar tracker rotates at a pre-determined angle from the sunrise to the sunset. The increment of the angle is determined through the data collected on the day and is analyzed and re-programmed for a better tracking ability on the next day. Whilst in the manual mode, the solar tracker is set to a desired angle by manually increasing or decreasing the angle via the input to the PIC microcontroller. In all modes, a night return algorithm repositioned the panel to its initial home position facing the East (at sunrise).

Besides, the ability to operate as a solar tracker, computer models of the PV panel and the electro-mechanical systems are modeled using MATLAB™/Simulink™ environment. In literature [20], a simple visual C++ program was used to provide an excellent graphic user interface (GUI) and control normal DC power supply output using the computer printer port to exactly simulate solar panel characteristics. As seen in [21-23], MATLAB ™/Simulink ™ was used to model and analyze the PV model characteristics. However, there are only a few attempts [24] to model the entire PV standalone system, including the electro-mechanical subsystem such as DC motor, drive transmission, microcontroller output, battery and charging module. Whether like or not, this is essential as they are parts of the PV systems and the influence on the overall performance such as efficiency and power output can be compromise if the electro-mechanical system is poorly design for the active solar tracker and when it is subjected to external disturbances such as wind and raindrops. With that in mind, the PV standalone system model consists of a PV panel, a servo motor, a battery, a charger, two LDR sensors, external disturbances and the microcontroller are modeled using MATLAB™/Simulink™. It was chosen due to its easy to program language supported by ready toolboxes and graphic block diagrams can be designed for complicated systems simulation. With the completed system model, it is used to determine the power and its efficiency over the fixed solar panel before actual implementation.

In summary, the book chapter presents the modeling and simulation of the solar tracker system consisting of the photovoltaic system under a constant load using MATLAB™ and Simulink™. The chapter is organized as follows. The overview of the electro-mechanical design of the single-axis solar tracker is described in Section 2. This is followed by the description of the proposed MATLAB™ and Simulink™ models in Section 3. The experiments and testing are described in Section 4. Lastly, conclusions are drawn in Section 5.

2. Solar tracker system descriptions

2.1. Mechanical structure

After the solar panels and other components were selected, the overall structural design of the solar tracker as seen in Fig.1 was fabricated. The solar tracker weight 3 kg and has an overall dimension of 340mm x 270mm x 500mm. The compactness of the proposed solar tracker enables it to be mounted on the wall. It consists of the PV panel, the pulley-chain transmission system; the motor and electronics boards support and the vertical pillar with

base plate support. The entire structure was fabricated using the aluminum rods and plates. The pillar holding panel is aligned to the center of the panel for better flexibility during the panel rotation. The tracker is designed to have a single-axis rotation (East to West), and the motor is mounted in such a way that the tracker systems have only a single-axis freedom of rotation. The fixture to hold the sensors are then assembled and aligned at both ends of the PV panel to sense the sun irradiance.

The PV panel frame support (as seen in Fig. 1) has a support rod that runs across the PV panel width. The pulley on the rod supported by two bearings is directly connected to the motor shaft via the pulley transmission system. The two mechanical stoppers at each ends were incorporated to limit the rotation of the panel. As shown in Fig. 1, the components were arranged along the vertical pillar mounted on the base plate support.

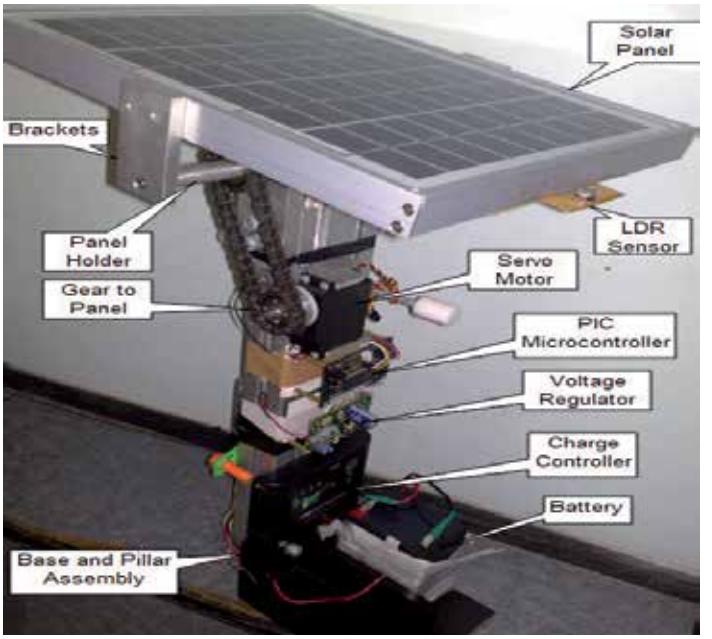


Figure 1. Actual solar tracker design

2.2. Electrical system

The overall mechanical and electrical subsystems were integrated into the solar tracker system as shown in Fig. 2. The detailed design of the circuitry could be found in [24]. The block diagram of the solar tracker system consists of mostly electrical components. The solar tracker consists of the PV cells, the charge controller and the lead-acid battery. Other subsystems such as the LDR sensors, the voltage regulator, and the microcontroller-PIC18F4520 target board were also used. The LDR sensors sense the sunlight intensity and send the signal to the microcontroller to rotate the PV panel via the servo motor. The electrical energy is then stored in the lead-acid battery that is later used to power the respective components.

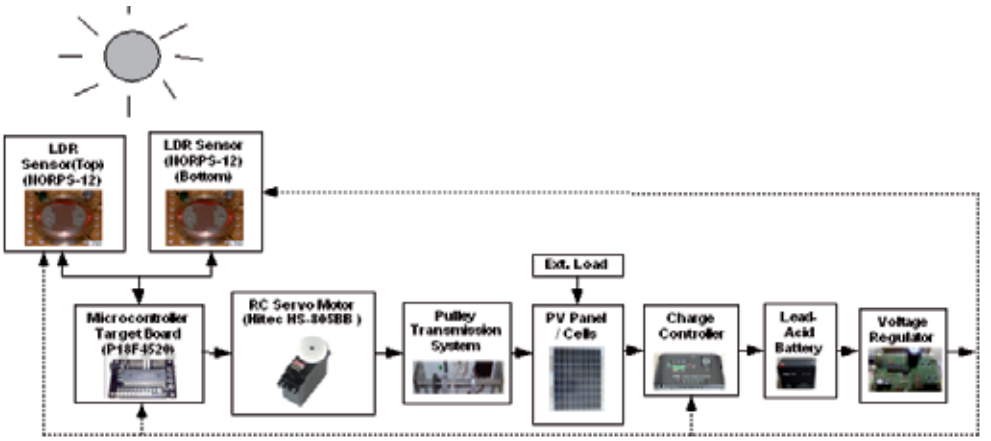


Figure 2. Schematic diagram of the standalone solar tracker system

The PV cells are a device that helps to convert the solar energy into electrical energy. The solar panel selected is capable of generating 10W power. As per the vendor specification, it weighs about 1.3kg with a dimension of 341mm x 269 mm x 28mm. Charge controller was supplied together with the solar panel's unit. It requires 12V supply and is capable of handling a maximum of 5A. The charge controller prevents the over-charging of the battery.

The LDR sensors (NORPS-12) are basically resistors that vary their resistance according to the sunlight intensity when exposed to irradiance. The output of the sensor circuit is an analogue voltage that is used as an input to the PIC microcontroller. To determine the value of resistor R, various values of different resistors were examined to finalize an appropriate resistor. The desired resistor value should provide a voltage that covers the sunny and cloudy conditions. The following resistor values as shown in Table 1 were tested. From the test result, it was found that varying the value of resistors in the voltage divider circuit helps to improve the sensitivity of the output. The resistor of 100Ω was found to be suitable to differentiate between the sunny and cloudy day.

Fixed Resistor (Ω)	V_{out} on sunny day	V_{out} on cloudy day	ΔV_{out}
50	2.14	0.82	1.32
100	***3.95***	***0.90***	***3.05***
200	4.56	1.35	3.21
500	4.78	1.41	3.37
1000	5.02	1.9	3.12

Table 1. Recorded Voltage variation at different resistor values

The driving mechanism includes the servo motor and the pulley system. The servo motor was controlled using the microcontroller. The controller uses the PWM (Pulse Width Modulation) signal to drive the servo motor at a controlled speed correspond to a maximum voltage of 6V. The PWM wave is a continuous square wave signal that changes between 0V and 6V. The duration or width of the pulse determines the angle of the shaft's rotation. A

voltage regulator circuit was used to bring the supply voltage down to a level suitable for use in the microcontroller, the charge controller and the LDR sensors. The microcontroller target board in the system was used to control the servo motor. It receives the signals from the LDR sensors. The analogue voltage is converted into digital signal (logic 1 or 0) for processing. The processor was a PIC18F4520 from Microchip Inc. The PICKIT2 programmer was then used to interface MPLAB Integrated Development Environment to the target board. To program the microcontroller target board, MPLAB C18 Compiler software that runs on the Microsoft Window as a 32-bit application was used. The C program was then compiled into assembly language before downloading into the target board.

2.3. Modes of operation

There are three modes of operation in the solar tracker. They are namely: automatic, preset and manual mode. In the automatic mode, the PIC microcontroller rotates the PV panel to balance the light intensity at both LDR sensors. In the case when both sensors receive a low voltage due to cloudy conditions, the PV panel is programmed to wait for 15minutes and automatically switched to preset mode (using internal real-time clock). In this mode, the PV panel is programmed to rotate 2° towards west in every 15 minutes. If the extreme position towards the west is sensed (at sunset), the night return algorithm repositioned the panel to its initial home position facing the East (at sunrise). In the manual mode, it allows the panel to rotate to the desired angle by manually increasing or decreasing the angle via the input to the PIC microcontroller. Once the PV panel is positioned to the desired angle, it switches back to the automatic mode.

In summary, the operation modes for the control of solar tracker and the features in these user options are shown below.

- When the SW1 switch is pressed, it rotates the panel to the home position and waits for the user to select the automatic mode (SW2), the preset mode (SW3) or manual mode (SW4).
- When the SW2 switch is pressed, it starts the tracking in “automatic mode”. In this mode, the rotation of the PV panel depends on the LDR sensors.
- When the SW3 switch is pressed, it starts the tracking in “preset mode”. In this case, it rotates the panel in a pre-determined angle till the sunset.
- When the SW4 switch is pressed; the panel is allowed to rotate manually to a desired position. Once it is positioned, it switches back to the automatic mode.

3. Standalone solar tracker system modeling

In this section, the main aim is to simulate the single axis solar tracking system during the automatic mode using MATLAB™ /Simulink™ (see Fig. 3). To obtain a more realistic model, the solar tracker is subjected to external disturbances such as wind and raindrop. It should be similar to the real prototype made in section 2 such that comparisons could be made fairly. All the data for building the simulation models were obtained from either the

components' datasheets or the experiments conducted. The simulation run was performed in every second of the entire 10 hours or 36000 seconds of experimental setup. The rapid accelerator mode has been chosen to reduce the simulation time to about 180-seconds real time (one-second second simulation time is equivalent to 180-seconds real time). The rapid accelerator mode gives the best speed improvement compared to normal mode when simulation execution time exceeds the time required for code generation. For this reason, the rapid accelerator mode generally performs better than normal mode when simulation execution times are several minutes or more. The ODE45 solver type of variable step size was used throughout the simulations. Solar tracker model developed in Simulink could detect the sun irradiance to produce the required current. The simulation model is implemented in such a way that when the sun irradiance falls on the sensors, the servo motor moves the PV panel in an incremental way until the sunset.

The PV tracking panel with two LDR sensors, namely: V_LDR_B and V_LDR_T are the bottom and top voltage outputs based on the corresponding sun irradiance data. The irradiance from the sun model was obtained by dividing the power obtained from the tracker by the surface area of the PV cells. The outputs were used as inputs to the microcontroller. The servo motor rotates the panel at an angle based on the microcontroller PWM signal. This process repeats again until the sunset. During the process, the PV panel generates direct current that keeps the 12V battery charged. The battery gets charged or discharged depending on the state of the charger. The external load was modeled by a pure resistor to simulate the loading on the motor shaft. The wind and rain droplet were modeled as additional current load (that is by deducting the actual current generated from the solar cells) on the solar tracker. The efficiency of the proposed solar tracking system over the fixed panel can be compared. With the model, it is used to determine the types of PV systems that could be successfully combined to give a required level of efficiency before actual implementation.

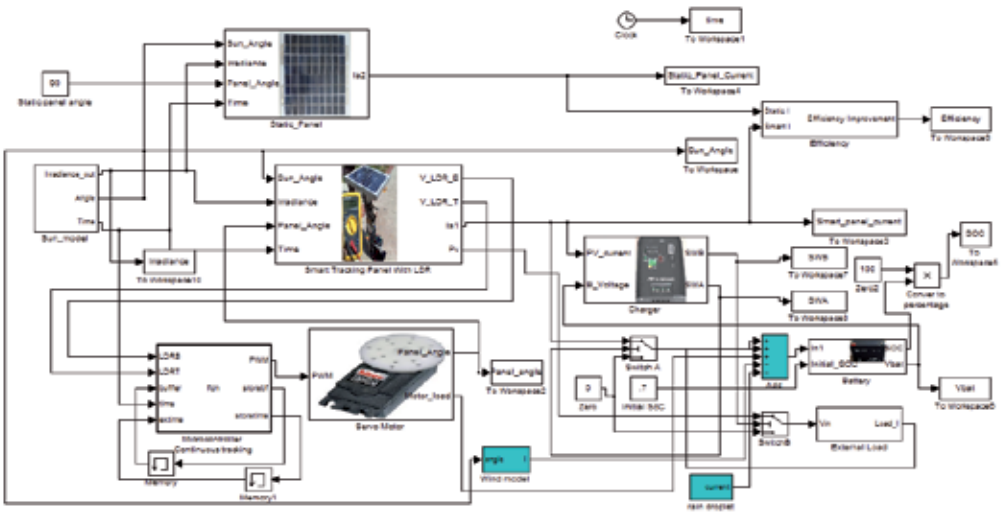


Figure 3. Overall tracking system block diagrams in Simulink

3.1. PV panel model

The simplest equivalent circuit of a solar cell is a current source in parallel with a diode as shown in Fig. 4. The current source represents the current generated by the PV cell due to the photons received by it, and is constant under constant sun irradiance and temperature. During darkness, the solar cell is not an active device; it works as a diode. It produces neither a current nor a voltage. However, if it is connected to an external supply (large voltage) it generates a saturation current or dark current. The key parameters for a PV cell are short circuit current (*Isc* or the current from the solar cell when the voltage across the cell is zero), open circuit voltage (*Voc*) and sun irradiance value. Usually these values are given by the manufacturer in the data sheet.

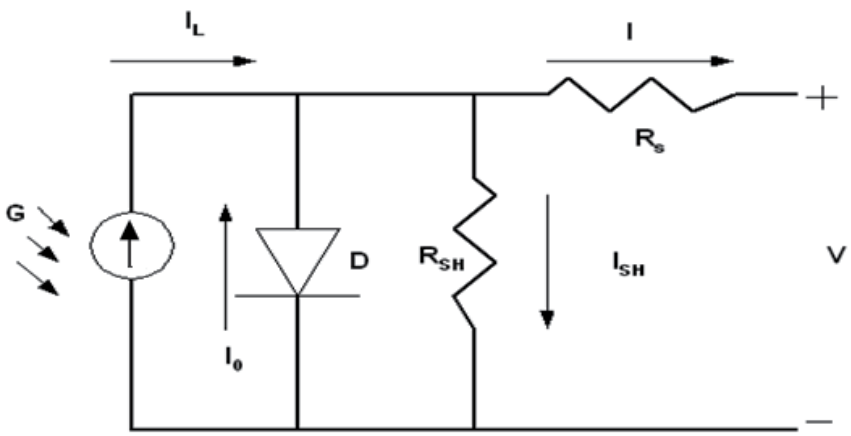


Figure 4. Circuit diagram of a PV cell model

Normally a single PV cell produces a rather small voltage that have less practical use. The real PV panel always uses many cells to generate a large voltage. For example the Kamtex-10W PV module used for our project comprises of 36 cells to generate a large enough voltage to charge a 12 volt battery. The data sheet for Kamtex-10W is given in Table 2.

Parameter	Value
Maximum Power (P_{max})	10W
Voltage at P_{max}	17.6V
Current at P_{max}	0.57A
Short-circuit current (I_{sc})	0.6A
Open-circuit voltage (V_{oc})	21.6V
Temperature coefficient of open-circuit voltage	-(10+/-1)mV/degree Celsius
Temperature coefficient of short-circuit current	(0.065+/-0.015)%/degree Celsius
Nominal Operating Cell Temperature (NOCT)	30+/-3degree Celsius

Table 2. Electrical characteristics of PV Module-Kamtex (KMX-10W)

The following parameters were used in the calculation of the net current of a PV cell.

- Saturation current of the diode, I_0.
- Net current from the PV panel I.
- Light-generated current inside the cell I_L.
- Series resistance R_s, which is internal resistance of the PV panel;
- Shunt resistance R_{sh}, in parallel with the diode, R_{sh} is very large unless many PV modules are connected in a large system;
- Diode quality factor, n;

In an ideal cell R_s is 0 and R_{sh} is infinite. The net current of the PV cells is the difference between the output current [25-26] from the PV cells and the diode current is given by.

$$I = I_L - I_o\left[e^{\frac{q(V+IR_s)}{nkT}} - 1\right] \tag{1}$$

where V is the voltage across the PV cell, k is the Boltzmann's constant (=1.381×10^{-23} J/K), T is the junction temperature in Kelvin, q is the electron charge (=1.602×10^{-19}C), n is the diode ideality factor (≈1.62).

$$I_L = I_L(T_1) + K_0(T_{ref} - T_1) \tag{2}$$

$$I_L(T_1) = I_{sc}(T_1).G(T_{ref}) \tag{3}$$

$$K_0 = \frac{I_{sc}(T_2) - I_{sc}(T_1)}{T_2 - T_1} \tag{4}$$

Here, T_{ref} equals to 298K is the reference temperature of the PV cell, G (T_{ref}) equals to 1000W/m^2, K_0 is the temperature coefficient are used. Then,

$$I_0 = I_0(T_{ref}) \times \left(\frac{T}{T_{ref}}\right)^{\frac{3}{n}} e^{-\frac{qV_g}{nk\left(\frac{1}{T_{ref}} - \frac{1}{T_1}\right)}} \tag{5}$$

$$I_0(T_{ref}) = \frac{I_{sc}(T_1)}{e^{\frac{qV_{oc}(T_1)}{nkT_1}} - 1}; \quad V_0(T_1) = \frac{nkT_1}{q}\ln\left(\frac{I_L}{I_0}\right) \tag{6}$$

where V_g is the band gap energy (=1.12eV). It is the energy needed to break a bond in the crystal, V_{oc} is the open circuit voltage corresponds to the photocurrent I_L. The resistance within each cell in the connection between cells is the series resistance, R_s

$$R_s = \frac{dV}{dI_{Voc}} - \frac{1}{X_v} \tag{7}$$

$$X_v = I_0(T_1)\frac{q}{nkT_1}e^{\frac{qV_{oc}(T_1)}{nkT_1}} - \frac{1}{X_v} \tag{8}$$

By Newton's method,

$$x_{n+1} = x_n - f(x_n)/f'(x_n) \tag{9}$$

where $f'(x)$ is the derivative of the function $f(x)=0$, x_n is a present value, and x_{n+1} is the next value. So,

$$f(I) = I_L - I - I_o\left[e^{\frac{q(V+IR_s)}{nkT}} - 1\right] = 0 \tag{10}$$

Then using Newton's equation:

$$I_{n+1} = I_n - \frac{I_L - I_n - I_o\left[e^{\frac{q(V+IR_s)}{nkT}} - 1\right]}{-1 - I_o qR_s.e^{\frac{q(V+IR_s)}{nkT}}} \tag{11}$$

By using MATLAB™, the above function can be computed numerically to obtain the net output current from the PV cells.

The fixed PV panel was modeled as shown in Fig.5 using the equations as seen in (11). The output voltage of the battery is 12V. The temperature (T_{aC}) obtained during the experiment and the sun irradiance data (G) that represents the intensity (or the power of sunlight falling per unit area) were used. The solar irradiance data were taken hourly, and the averages in each hour were then tabulated. The PV Cells Array block diagram computes the net current from the PV cells using the embedded MATLAB function. The plot for the actual and simulated net output current is shown in Fig. 6. The deviation in the plot may due to the averaging done during each hour and the power consumed before noon was actually higher than expected. This is reasonable as the sunlight is stronger during the day.

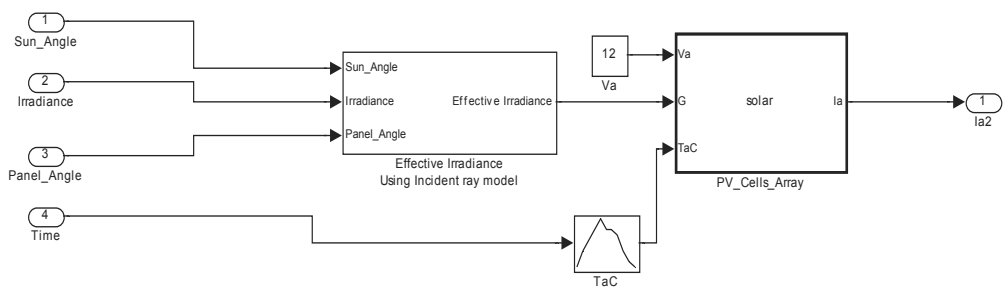


Figure 5. Fixed PV panel model

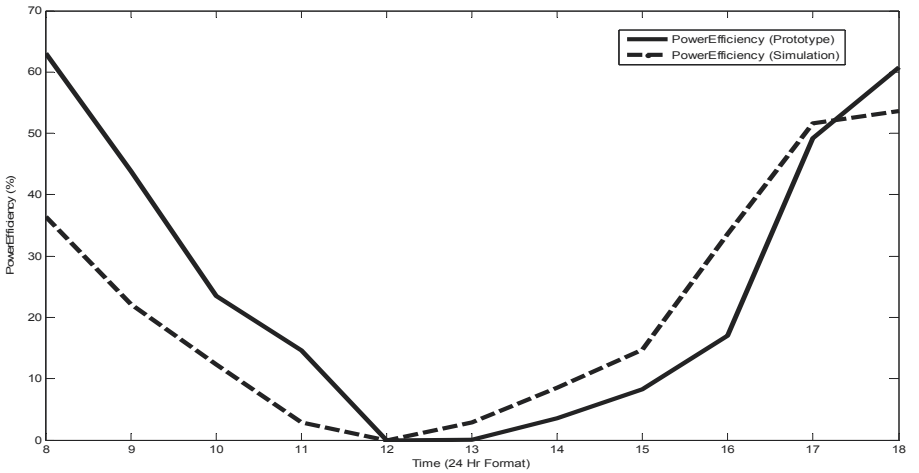


Figure 6. Actual and simulated net output current value of fixed panel

To simulate the sun irradiance at different PV panel's angle, the effective irradiance was used. Details of the effective irradiance block diagram can be seen in Fig. 7. The block diagram defined the angle between sun's incident ray and PV panel. For a static panel, it is always parallel to the ground that is at 90 degrees (0 degree for sunrise and 180 degrees for sun set). A simple program was written to obtain the relationship of the effective sun irradiance when the difference between the sun angle and panel angle is more than +/-90 degrees. To limit the angle to 90 degrees, the cosine trigonometric function was introduced in the model to create the zero sun irradiance when such a situation occurs.

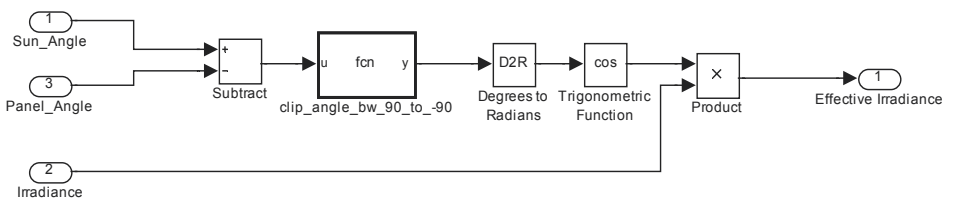


Figure 7. Effective sun irradiance model

3.1.1. Smart tracker PV panel

The smart tracker panel was installed with two LDR sensors. Assuming both sensors are placed in parallel with the PV panel, the effective irradiance is similar. As the results, the smart tracker is unable to perform the proposed sun tracking algorithm. To circumvent this, the top and bottom sensors were positioned at 45 degrees and 135 degrees respectively as seen in Fig. 8. When the sunlight falls onto the PV panel, the LDR sensors generate different voltages (that is V_LDR_B and V_LDR_T according to the changes in the sun irradiance) to move the PV panel.

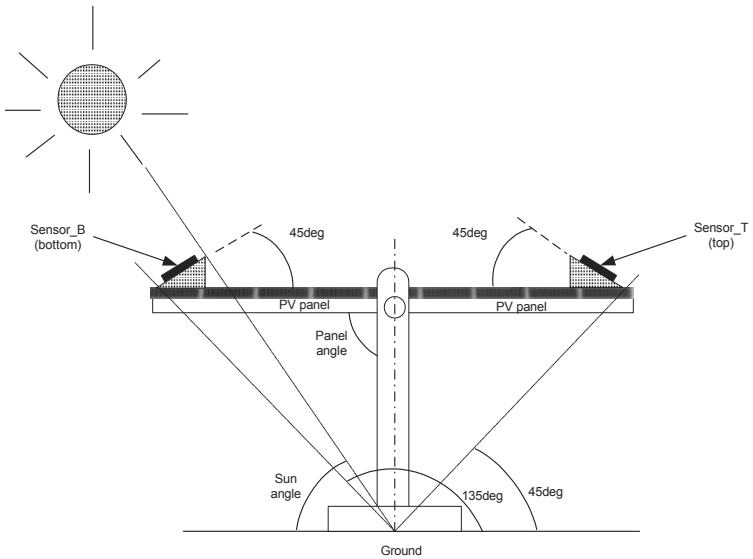


Figure 8. PV panel and LDR sensor angle position

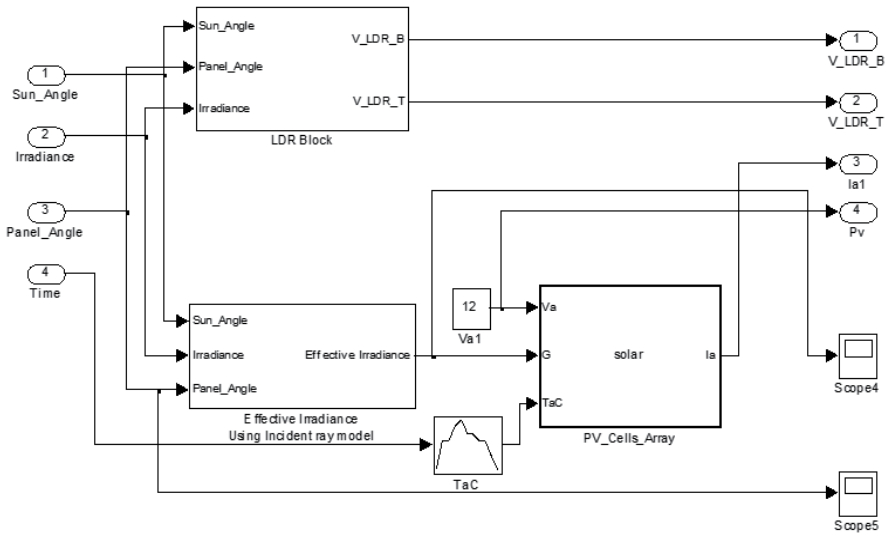


Figure 9. Smart tracker PV panel Model

3.2. Sun model

At each time instant, the actual sun irradiance data obtained from the experiment was used. In the sun model, the sun is assumed to travel from the 0 degree (sunrise) to 180 degrees (sunset) from 7am to 5pm. During these 10 hours, the PV panel rotates 180 degrees. As shown in Fig. 10, the initial sun's angle is at 30 degrees and with the angle changes at 15 degrees per hour or 0.004147 per simulated time in second; the corresponding sun angle (with respect to the base) is obtained.

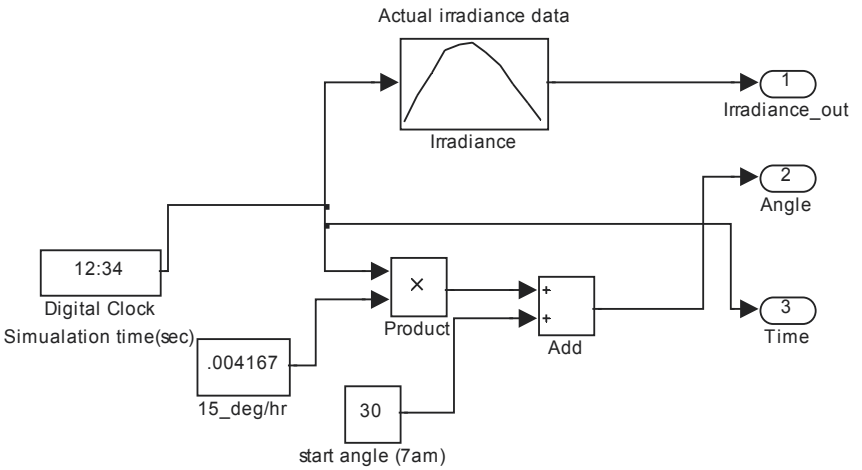


Figure 10. Sun's model

3.3. LDR sensor model

The LDR sensor is a variable resistor that changes the resistance according to the intensity of incident ray illuminated onto it. As the intensity of sunlight changes, the resistance and the voltage of LDR sensors change. The output voltage across the resistor (resistance value of 100Ω) is converted into digital signal at the input of the microcontroller. Based on the TTL input, the servo motor rotates clockwise (CW) or anticlockwise (CCW). In the LDR sensor model as shown in Fig. 11, the difference between the panel angle and the assumed sun position was calculated. The angles were limited to +/- 90 degrees. When 90 degrees is reached, the LDR sensor output a zero irradiance that corresponds to a certain voltage as shown in Table 1. Recalled, V_LDR_B and V_LDR_T are the voltage output from the bottom and top sensors.

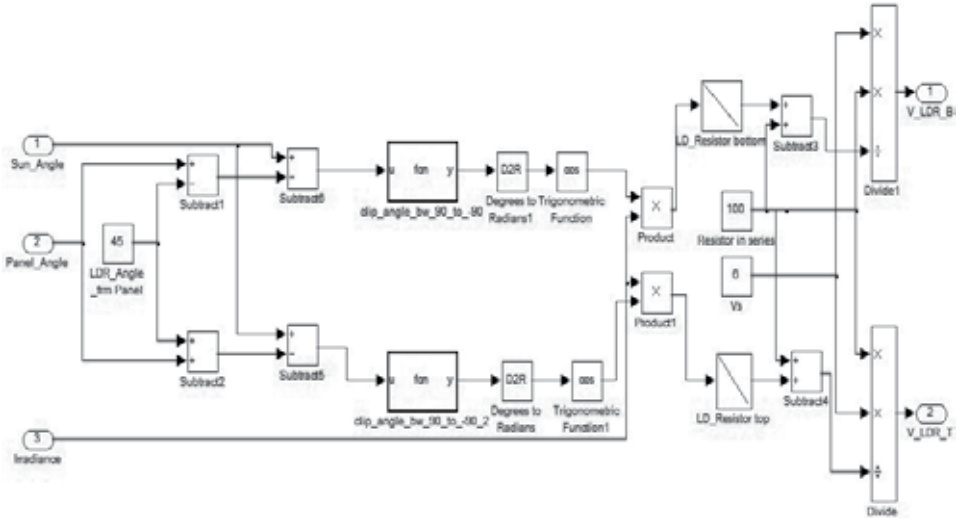


Figure 11. LDR sensor model

3.4. Microcontroller model

As shown in Fig. 12, the microcontroller model is modeled using the embedded MATLAB™ function. The inputs to this function are LDR_B and LDR_T, a real-time clock and initial buffer value of 1.5. One of the inputs (named Extime) is used to compare the current time with the previous time when the PWM value changes. The microcontroller generates output duration of 1.5ms to rotate the PV panel if the voltage difference of the LDR sensors is less than 0.07V and are both less than 0.75V (very low irradiance). If LDR sensor's voltages are both greater than 0.75V but the voltage difference is less than 0.07V, the PV panel remains in the current position. When the LDR sensor values is greater than 0.07V, the motor turns the PV panel by adjusting its PWM value until the sensors' voltages are equal. The delay time of 0.7 seconds and increment steps were found using the trial-and-error method during the simulation.

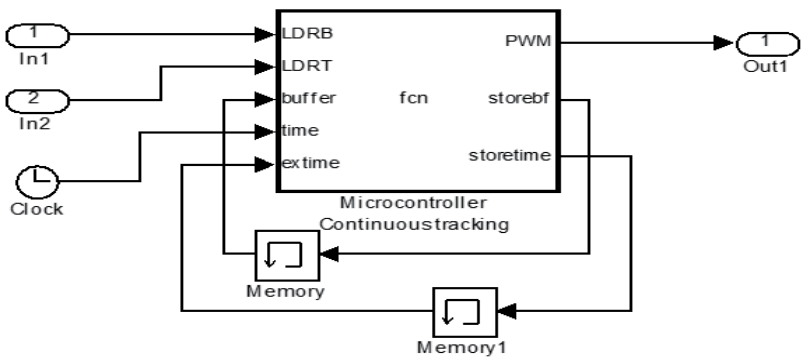


Figure 12. Microcontroller model

3.5. Charger model

A charger is essential to protect the battery from over-charged and fully drained. In an ideal case, the battery charge remains between 20% and 100% based on a PSpice photovoltaic model [27]. This charger model (see Fig. 13) includes two switches namely: Switch A and B to control the battery voltage flow. Switch A remains deactivated for any value of battery charge above 12.95V (100% charged). Switch B deactivates once the battery charge drops to 11.6V (20% charge left). The truth table for the charge control is given in Table 3.

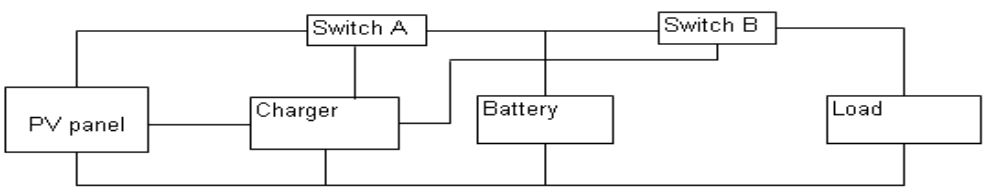


Figure 13. Block diagram of charge control

The Switch A is connected to the PV module on one side and the battery on the other side. When the net output current from the PV panel is positive, it begins to charge the battery until the maximum charge of 12.95V is reached. When there is no current from PV panel, the switch remains activated, the current required is obtained from the battery. After the battery is fully charged, the switch disconnects. The switch is activated again if the battery voltage drops below to 12.2 V.

The Switch B is connected when the battery is in charging mode (greater than 12.25V). When the battery voltage drops below 11.6 V, the battery is draining, and hence it is important to cut off the loads in order not to damage the motor. The conditions for the switching are given in Table 3 and the corresponding values of Switch A and Switch B (previous states and current states) are given by X, Y, M and N respectively. The value '1' represents the closed switch and '0' represents the open switch.

Voltage range of battery	X	Y
V>=12.95	0	0
V>=12.95	1	0
12.2<V<12.95	0	0
12.2<V<12.95	1	1
V<=12.2	0	1
V<=12.2	1	1

Voltage range of battery	M	N
V<=11.6	0	0
V<=11.6	1	0
11.6<V<=12.25	0	0
11.6<V<=12.25	1	1
V>=12.25	0	1
V>=12.25	1	1

Table 3. Truth table used for discharging (left) and charging (right) condition

The Simulink model of the charger can be seen in Fig. 14. The Compare block diagrams were used to compare the various conditions as shown in Table 3. The logic gate after the comparisons serves as the truth table for operating the Switch A and B.

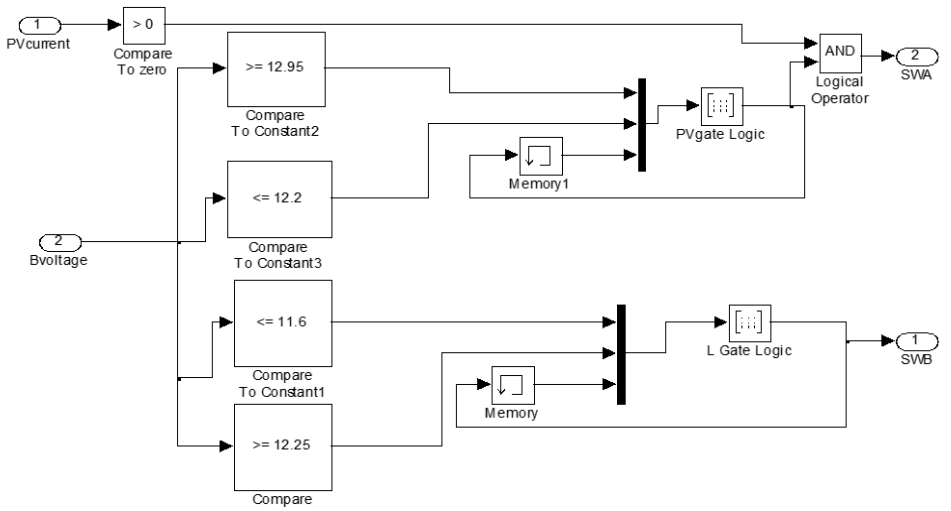


Figure 14. Charger model in Simulink

3.6. Battery model

The lead-acid battery model was implemented based on a PSpice model [27] for a lead acid battery. It has two modes of operation – charging and discharging modes. When the current to the battery is positive (negative), the battery is in the charging (discharging) mode. The following parameters were used for modeling the battery.

- SOC_I is the initial state of charge,
- SOC (%) is the available charge.
- SOC_m is the maximum state of charge.
- n_s is the number of 2V cells in series.
- D (h^{-1}) is the self-discharge rate of battery.
- K_b (no unit) is the charging and discharging battery efficiency.

As SOC varies linearly with V_{ocb} (open circuit voltage of the battery), the relationship between open circuit battery voltage and state of charge can be determined using the Table 4.

Voltage	State of Charge (%)
12.63	100
12.54	90
12.45	80
12.39	75
12.27	60
12.18	50
11.97	25
11.76	Completely discharged

Table 4. State of charge with respect to Voc

The terminal voltage for the battery is given by:

$$V_{bat} = V_1 + I_{bat} R_1 \tag{12}$$

Here V_1 and R_1 both depend on the mode of battery operation and have different equations. Battery current; I_{bat} is positive when battery is in charge (ch) mode and negative when in discharge (dch) mode.

In charging mode, we can write the resistance and voltage as follows:

$$R_1 = R_{ch} = \left(0.758 + \frac{0.139}{[1.06 - SOC(t)]n_s}\right)\frac{1}{SOC_m} \tag{13}$$

$$V_1 = V_{ch} = [2 + 0.148 \cdot SOC(t)]n_s \tag{14}$$

where SOC (t) represents the current state of charge (%), SOC (t) is defined by a set of equations later. In discharging mode, the resistance and voltage are written as follows:

$$R_1 = R_{dch} = \left(0.19 + \frac{0.1037}{[SOC(t) - 0.14]n_s}\right)\frac{1}{SOC_m} \tag{15}$$

$$V_1 = V_{dch} = [1.926 + 0.124 \cdot SOC(t)]n_s \tag{16}$$

To estimate the value of SOC (t), the following equations have been used to describe them in the PSpice model[27].

$$SOC(t + dt) = SOC(t)\left(1 - \frac{D\ dt}{3600}\right) + \frac{K_b\left(V_{bat} I_{bat} - R_1 I_{bat}{}^2\right)\ dt}{3600} \tag{17}$$

In equation (17), the time is assumed in seconds, so some terms must be divided by 3600 such that SOC is in Wh. By substituting V_{bat} as a function of V_1, the value of SOC (t) can be determined as shown.

$$SOC(t) = SOC(t-1) + \frac{1}{3600}\int_{t-1}^{t}\left[\frac{K_b V_1 I_{bat}}{SOC_m} - SOC(t-1)D\right] dt \tag{18}$$

The Simulink model in Fig. 15 is used to model the charging and discharging conditions during the process. The left-hand side of the Fig. 15 shows the respective functions used. There are namely: V_{ch}, V_{dch}, R_{ch} and R_{dch}. The input to the battery is the net current output from the PV panel and the output is the battery voltage, V_{bat}. In order to obtain 12V, six 2V cells denoted by n_s were used. The maximum state of charge, SOC_m was set to 84. The discharge rate, D and the efficiency, K_b are set as 1.5×10^{-5} and 0.8 respectively. As the battery is in charging or discharging mode, it allows only one value for R_1 and V_1 to the equations.

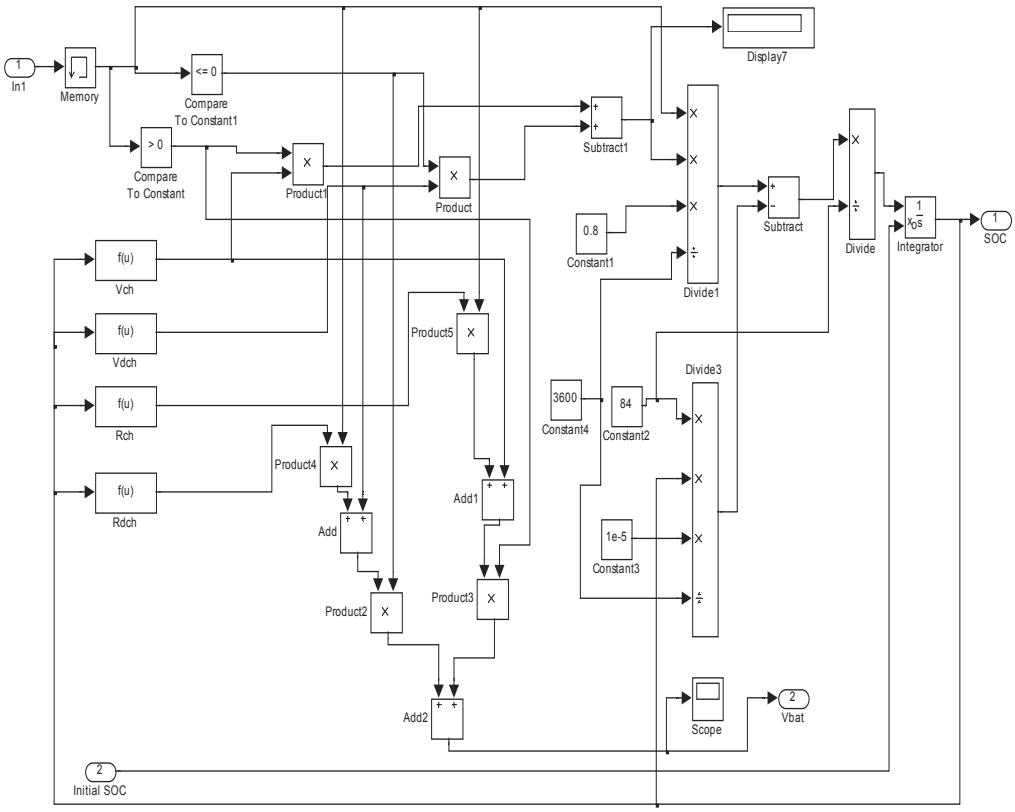


Figure 15. Battery model in Simulink

3.7. PWM servo motor model

The solar panel is designed to drive the PV panel in a small angle, between 0 to 180 degrees at a low speed. PWM is used to control the motor. The PWM is a continuous square wave with a period of 20ms. With the PWM signal, the output shaft of the servo motor changes the angular position of the PV panel. The following parameters are used for the servo motor.

- Moment of inertia (J) = 0.01 kg.m^2/s^2;
- Damping ratio (b) = 0.1 Nms;
- Electromotive force constant (K_t) = 0.01Nm/Amp;
- Back electromotive force constant (K_e) = 0.01s.s/rad;
- Electric resistance (R_m) = 1 ohm;
- Electric inductance (L_m) = 0.5H;
- Input Voltage (V_m);
- Output angle (θ);

For the PWM servo motor, the transfer function between the output rotational angle and the input voltage is written as follows.

$$\frac{\theta(s)}{V_m(s)} = \frac{15}{(s+1)(s+1)+14} = \frac{15}{s^2+2s+15} \tag{19}$$

Based on this transfer function, the servo motor model can be modeled in Simulink (as seen in Fig. 16) using the look-up tables. The look-up table uses the PWM as an input to rotate the motor to a pre-determined angle. When pulse width changes from 1.25ms to 1.75ms, the panel angle changes from the 0 degree to 180 degrees in a linear manner. The second look-up table on the feedback path provides the actual pulse width results. The actual and the desired pulse-width are then compared to obtain the error signal for the Proportional-Integral-Derivative (PID) controller (using the controller gains: $K_p = 60, K_I = 30, K_D = 3$) to drive the motor to the desired angle. The embedded MATLAB™ function block is used to deactivate the motor load when it is not turning. An external load was added to show whether the motor is able to drive the PV panel. The weight could vary due to the modeling error. In this case, the external load is modeled as a pure resistance value (≈40Ω).

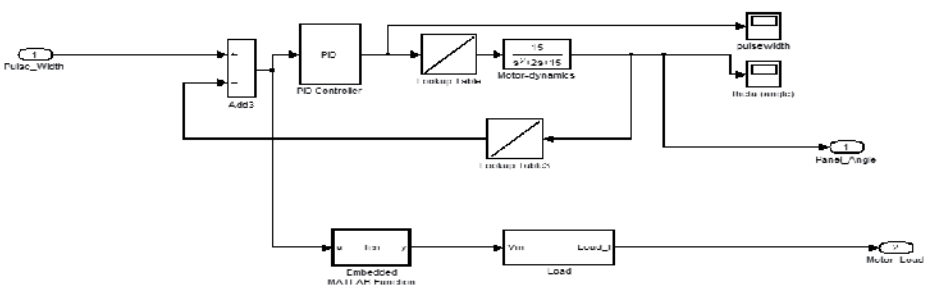


Figure 16. PWM Servo motor model in Simulink

3.8. Wind model

Tracking the sun irradiance in an ideal condition would involve no disturbances on the system. But in practice, factors such as external force like wind and rain (as seen in Fig. 17) moves the tracker away from its position. It is therefore vital for the solar tracker to be able to track the sun throughout the daytime. Two types of disturbances are considered: wind and raindrop as depicted in Fig. 17 schematically.

The drag force due to the wind comprises of two main forms: (a) friction drag is present when the panels are near to horizontal position; (b) Pressure drag is prominent when the tracker is positioned near to vertical position. As a preliminary model, the drag coefficient is modeled as a flat plate and the area of the solar tracker varies with angle, θ. The seventh power law [28], as shown below can estimate the air velocity over the tracker.

$$\frac{u}{U_o} = \left(\frac{y}{d}\right)^{1/7} \tag{20}$$

where u is air velocity at height y; U_0 is air velocity at boundary layer; y is height above surface; d is height of boundary layer (taken from ground).

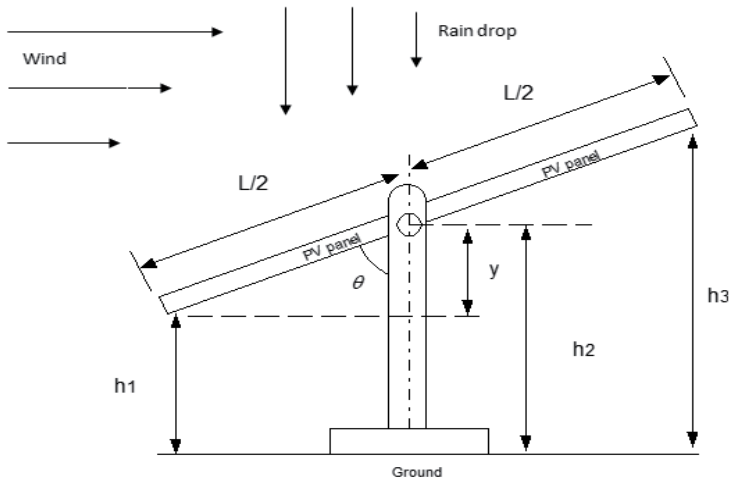


Figure 17. Schematic diagram of disturbances acting on solar tracker

Taking small changes on both sides,

$$du = \frac{U_o}{d^{1/7}}(dy)^{1/7} \tag{21}$$

For a certain velocity at any height above the ground, the drag force acting on a tilted plate is written as:

$$F_D = \frac{1}{2} C_d \rho_a A_a U^2 \tag{22}$$

Using (21) and (22), the wind drag force must be integrated over the area of the tilted solar tracker facing the wind. The resulting moment or torque about the central axis becomes:

$$dM = \frac{1}{2} C_d \rho_a A_a \cos\theta \left(\frac{U_o}{d^{1/7}}\right)^2 \left[\left(\int_{h_2}^{h_1} \left(y - h_2\right)^{7/2} dy\right)^{1/7} - \left(\int_{h_3}^{h_2} (h_2 - y)^{7/2} dy\right)^{1/7}\right]^2 \tag{23}$$

where h_1 and h_3 are heights of each side of the solar tracker; h_2 is height of centre axis of rotation as shown in Fig. 17. This moment differential about the central axis, due to the wind force, can be modeled using MATLAB/Simulink as a continual force that consumes the current obtained from the irradiance.

The Simulink model is shown in Fig.18. Equation (23) was modeled in Simulink. The integral terms in (23) are modeled as a summation in the difference between the lower vertical height (y-h_2) acting in counter-clockwise direction and its upper vertical height (h_2-y) acting in the clockwise direction. The values for the parameters as shown in Fig. 18 are as follows: h_1=0.415m, h_2=0.500m, h_3=0.585m, L= 0.34m, ρ_a = 1.165 kg/m^3 (at outdoor temperature of 30°C), A = 0.092m^2, U_o=0.8m/s (around 1.56 knots) , C_d=1.0 and d=1m.

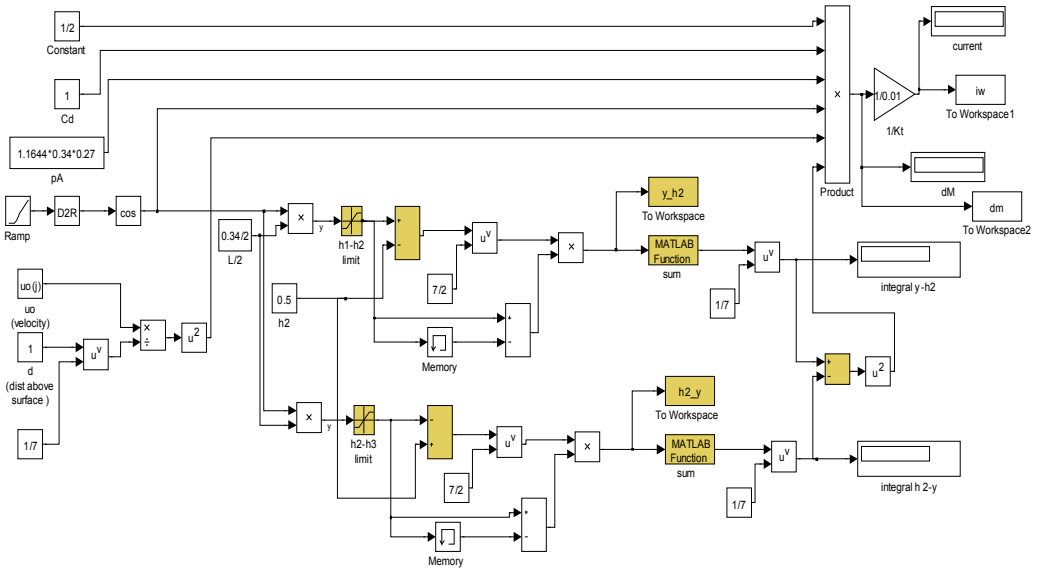


Figure 18. Simulink model for wind force

As shown in Fig. 19, the increases in the wind speed causes the current consumption (or moment) acting on the tilted solar tracker to increases. It could be observed that the wind loading is cyclic in nature as the moment acting on the solar tracker changes as time increases. As an increase in the torque required implies an increase in the current (as shown in the relationship K_t=0.01Nm/A) needed to drive the tilted solar tracker, this causes an increase in the current drawn from the battery. For a wind speed of 0.8m/s (or 1.56 knots), the estimated moment acting on the tilted solar tracker is estimated to be around 0.0022Nm and the corresponding current consumed is computed to be approximately 0.22A.

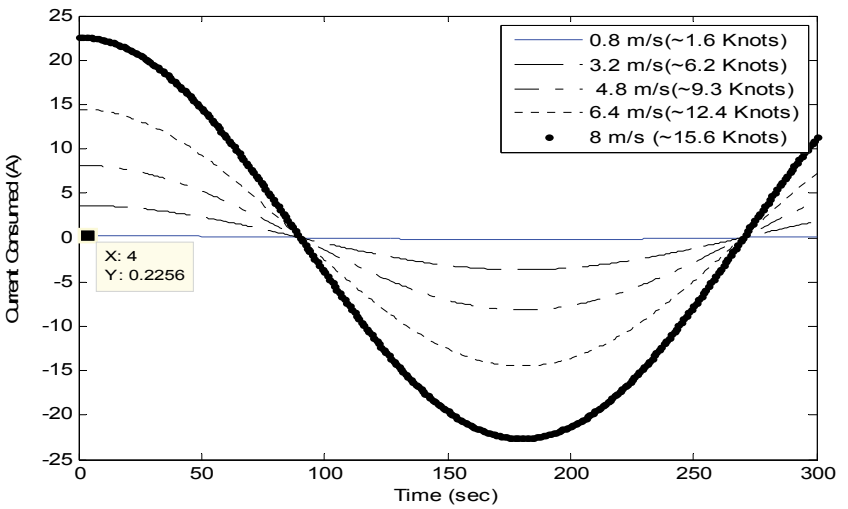


Figure 19. Current consumed due to wind force over time

3.9. Raindrop

Raindrop is driven along by the gravitational force. The movements can be evaluated by considering the equation of motion of the particles in horizontal along-wind, cross-wind and vertical directions. In this paper, we considered the raindrop that is not driven by the wind. Instead, we consider the mass of rain droplet (m) acting on the surface of the solar tracker. As the volume of the rain droplet is spherical in shape, the mass can be computed by multiplying it by the water density. The total force acting on the surface of the solar tracker can be computed as:

$$F_R = \frac{4}{3}\pi r_r^3 \rho_w g \tag{24}$$

where ρ_w is the water density, r_r is the radius of the raindrop and g is the gravitational constant.

In the simulation, the total force, F_r acting on the surface of the solar tracker could be easily modeled. The point force acting on the surface is translated into the torque acting about the central axis by multiplying the value by $L/2$ (that is half of the width of the solar panel). The radius of the rain droplet r_r is assumed to be small, that is 0.25mm instead of large raindrop. The current consumed as a result of the rain droplet is around 0.011 mA.

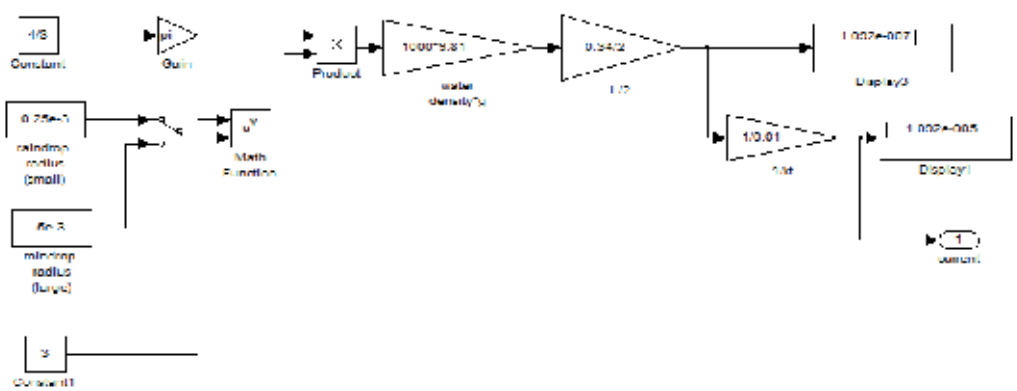


Figure 20. Rain droplet model on surface of solar tracker

4. Experimental and simulation results

In order to validate the proposed modeling, it was necessary to compare the experiment results for the fixed panel with the smart solar tracker system. To obtain this data, simple experiments were performed. The experiment setup for both fixed, and tracker system can be seen in Fig. 21. The setups were installed on building roof top that was 40m above the ground. The temperatures during the experiment were recorded using the Type-K thermocouple sensor. The open-circuit voltage and the current readings were recorded using a multi-meter connected to the solar cells. The climatic condition considered for

experimental was sunny during the entire test period. The average temperature recorded was around 30˚C and the local wind speed was broadcasted to be around 0.8m/s (or 1.6knots) during the tests.

Figure 21. Experiment Setup for fixed (left) and smart tracker (right) system

The simulation results of the solar tracker using the Simulink are shown in Fig. 22. As observed from the plots, the solar tracker is able to follow the sun angle. In the plot, the smart solar tracker produces a higher current output as compared to the static PV panel. The other part of the graph shows the state of the charge during the simulation. As seen in the battery voltage, the Switch A was deactivated when the maximum allowable voltage of 12.95V reached. The switch B was always connected during the entire duration (in charging mode).

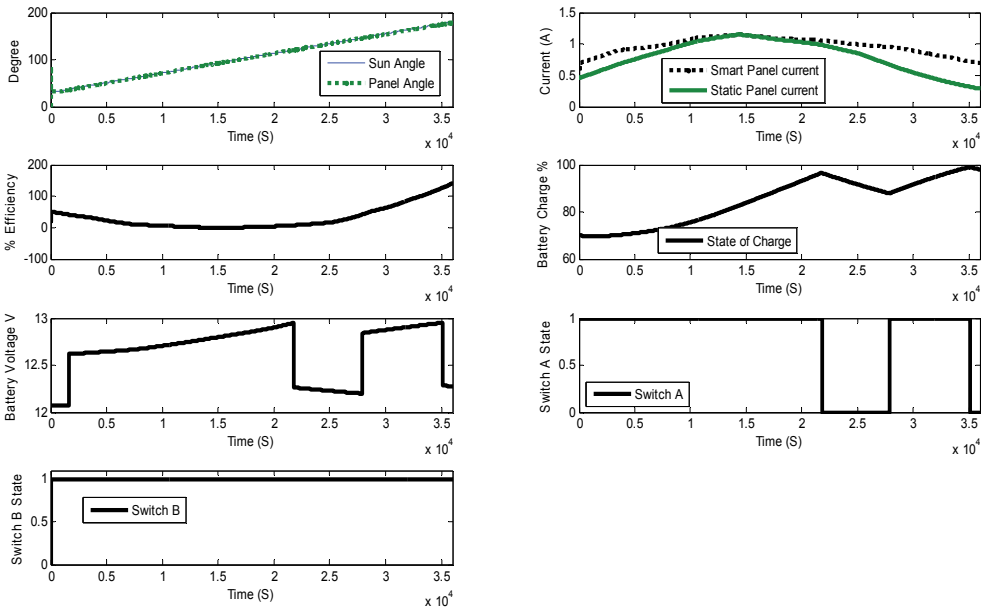


Figure 22. Simulation results of the standalone smart solar tracker

In Fig. 23, the simulated power output (or current) over the day was quite close to the actual results obtained. There was some slight deviation during the noon and evening period due to the modeling of the actual irradiance obtained from the experiment. Furthermore, the current consumed during the actual test was different as the wind loading and other disturbances were not modeled in the simulation.

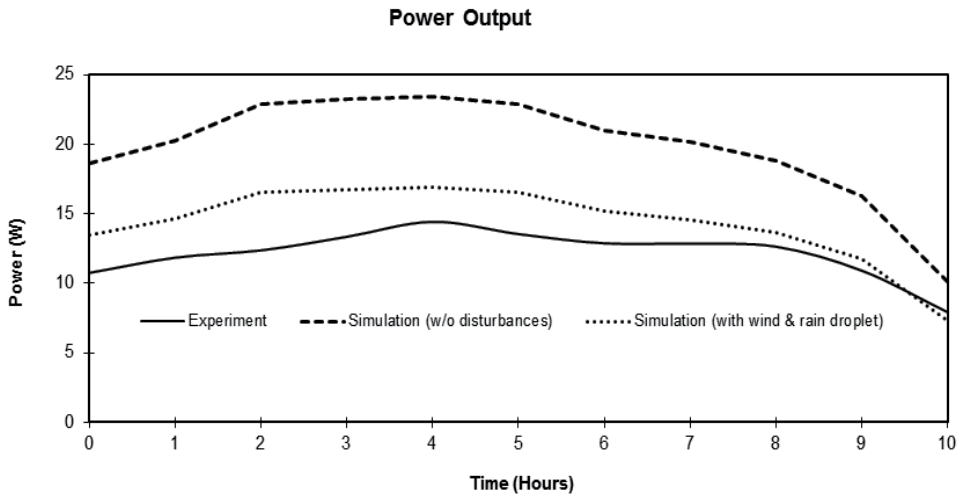


Figure 23. Simulated and actual power generated by smart solar tracker

The output power of the smart tracker was compared with the fixed panel design in order to determine the efficiency of the solar tracker system. As expected the overall power (or the efficiency) generated by the tracking panel is higher than that of the static panel. However, with the disturbances acting on the simulated models (see Fig. 23), the simulated curve exhibits closer behavior to the actual response obtained from experiment. Here, the efficiency refers to the ratio of difference between the sum of smart tracker and the fixed panel power to the sum of fixed panel power over the period of interest. The efficiency (obtain from experiment results) is around 20%. As compared to various the solar trackers as seen in [3], the average efficiency is around 12-15% and hence, the proposed design is slightly higher and comparable to the existing design. The solar tracker was efficient during most of the day except during the noon time where the sun irradiance was found to be the highest. The fixed panel that lies horizontally on the ground is therefore, quite comparable to the solar tracker system and hence zero efficiency was obtained. This explained why it is common to use a fixed panel in a sunny day where it experienced the maximum sun irradiance during the noon time.

For the cost and benefit of the proposed solar tracker system, it has some special features such as the initial expenditure on the equipment is usually high but there is no fuel cost involved, and the maintenance cost is low. The accumulated data [29] show that at present a PV system is competitive where small amounts of energy are required at a place that is far away from an electric grid or any other source of energy. For the economic evaluation of a system, the parameters that are usually considered are the life-cycle cost (LCC), payback

period (PP) and rate of return (RR). LCC is the sum of all the costs of a system over its lifetime, expressed in today's money. In case of the analysis of a PV system, the lifetime of the modules is usually taken as 20 years. The calculations were made on the basis of the approach described in [29-30]. Our estimates of the electricity cost matched with those reported in [29] for the PV, diesel generator and grid-extension systems. Our analysis shows that the cost of electricity from a PV system is approximately equal to that from a diesel generator and cheaper than a grid extension. This PV system may be used for domestic applications, especially in remote areas. Keeping in view the environmental impact and economic assessments of the designed PV system it is evident to the listed data that the PV system even at present is a competitive choice for small power requirements. With the expectation that the cost of a solar module will reduce from around $ 3/Wp to somewhere below $2/Wp in the near future (see one of the quarterly report from Solarbuzz.com), a PV system may as well become an economic and more attractive option for higher loads.

5. Conclusions

The design, modeling and testing of an active single axis solar tracker were presented. In the proposed design and operation of the solar tracker system, the sun was not constantly tracked based on the simulated irradiation. This helps to prevent unnecessary energy to be consumed by the devices and the system stops moving when the night falls. Hence, the proposed control structure provides the flexibility to accommodate different weather conditions, and also different user preference. The completed MATLAB™/Simulink™ model of the solar tracker with external disturbances is first used to provide a computer-aided design tool to determine the efficiency over the fixed solar panel, net current output, power generated and the types of PV systems that can be combined to give a required level of efficiency before actual implementation. The experimental results show a similar behavior in the power, the efficiency and the current output over the fixed solar panel when compared them with the simulation results. The external disturbances such as wind loading and raindrop model provide an insight to the impact of the current consumption on the model of the solar tracker before actual implementation.

Author details

C.S. Chin
Faculty of Science Agriculture and Engineerin,Newcastle University,
Newcastle upon Tyne , United Kingdom

6. References

[1] Lopes L A C, Lienhardt A M, (2003) A Simplified Nonlinear Power Source for Simulating PV Panels. IEEE 34th Annual Conference on Power Electronics Specialist: 1729- 1734.

[2] Kroposki B, DeBlasio R (2000) Technologies for the New Millennium: Photovoltaics as a Distributed Resource. IEEE Power Engineering Society Summer Meeting: 1798 – 1801.

[3] Mousazadeh H, Keyhani A, Javadi A, Mobli H, Abrinia K, Sharifi A (2009) A Review of Principle and Sun-Tracking Methods for Maximizing Solar Systems Output. Renewable and Sustainable Energy Reviews 13:1800-1818.
[4] Mwithiga G, Kigo S N (2006) Performance of a Solar Dryer with Limited Sun Tracking Capability. Journal of Food Engineering 74: 247-252.
[5] Poulek V (1994) Testing the New Solar Tracker with Shape Memory Alloy Actors. IEEE Photovoltaic Specialists Conference 1: 1131–1133.
[6] Clifford M J, Eastwood D (2004) Design of a Novel Passive Solar Tracker. Solar Energy 77: 269-280.
[7] Bingol O, Altintas A O (2006) Microcontroller Based Solar-Tracking System and Its Implementation. Journal of Engineering Sciences 12:243-248.
[8] Koyuncu B, Balasubramanian K (1991) A Microprocessor Controlled Automatic Sun Tracker. IEEE Transactions on Consumer Electronics 37: 913-917.
[9] Zeroual A, Raoufi M, Ankrim M, Wilkinson A J (1998) Design and Construction of a Closed Loop Sun-Tracker with Microprocessor Management. Solar Energy 19: 263-274.
[10] Jinayim T, Arunrungrasmi S, Tanitteerapan T, Mungkung N (2007) Highly Efficient Low Power Consumption Tracking Solar Cells for White LED-Based Lighting System. International Journal of Electrical Computer and Systems Engineering 1: 1307-5179.
[11] Hatfield P (2006) Low Cost Solar Tracker, Bachelor of Electrical Engineering Thesis, Department of Electrical and Computer Engineering, Curtin University of Technology.
[12] Rumala S (1986) A Shadow Method for Automatic Tracking. Solar Energy 37: 245-247.
[13] Palavras I, Bakos G C (2006) Development of a Low-Cost Dish Solar Concentrator and its Application in Zeolite Desorption. Renewable Energy 3: 2422-2431.
[14] Kalogirou S A (1996) Design and Construction of a One-Axis Sun-Tracking. Solar Energy 57: 465-469.
[15] Poulek V, Libra M (2000) A Very Simple Solar Tracker for Space and Terrestrial Applications. Solar Energy Materials and Solar Cells 60: 99-103.
[16] Mohamad A (2004) Efficiency Improvements of Photo-Voltaic Panels using a Sun Tracking System. Applied Energy 79: 345-354.
[17] Khader M, Badran O, Abdallah S (2008) Evaluating Multi-Axes Sun-Tracking System at Different Modes of Operation in Jordan. Renewable and Sustainable Energy Reviews 12: 864-873.
[18] Roth P, Georgiev A, Boudinov H (2004) Design and Construction of a System for Sun Tracking. Renewable Energy 29: 393-402.
[19] Hession P J, Bonwick W J (1984) Experience with a Sun Tracker. Solar Energy 32: 3-11.
[20] Kulkarni S S, Thean C Y, Kong A W (2003) A Novel PC Based Solar Electric Panel Simulator. The Fifth International Conference on Power Electronics and Drive Systems: 848 – 852.
[21] Yu T C, Chien T S (2009) Analysis and Simulation of Characteristics and Maximum Power Point Tracking for Photovoltaic Systems. International Conference on Power Electronics and Drive Systems: 1339-1344.
[22] Tsai H L (2010) Insolation-Oriented Model of Photovoltaic Module using Matlab/Simulink. Solar Energy 84: 1318-1326.

[23] Tsai H L, Tu C S, Su Y J (2008) Development of Generalized Photovoltaic Model Using MATLAB/Simulink, Proceedings of the World Congress on Engineering and Computer Science, San Francisco, USA:1-6.
[24] Chin C S, Babu A, McBride W (2011). Design, Modeling and Testing of a Standalone Single-Axis Active Solar Tracker using MATLAB/Simulink. Renewable Energy 36(11): 3075-3090.
[25] Enrique J M, Duran E, Sidrach-de-Cardona M, Andu J M (2007) Theoretical Assessment of the Maximum Power Point Tracking Efficiency of Photovoltaic Facilities. Solar Energy 81: 31–38.
[26] Villalva M G, Gazoli J R, Filho E R (2009) Comprehensive Approach to Modeling and Simulation of Photovoltaic Arrays. IEEE Transactions on Power Electronics 5: 1198–1208.
[27] Luis C, Silvestre S (2002) Modelling Photovoltaic Systems Using PSpice, John Wiley and Sons, Chichester.
[28] White F M (1999) Fluid Mechanics, fourth ed. McGraw-Hill.
[29] Karimov K S, Saqib M A, Akhter P, Ahmed M M, Chattha J A, Yousafzai S A (2005) A Simple Photo-Voltaic Tracking System. Solar Energy Materials and Solar Cells 87: 49-59.
[30] Markvart T (2000) Solar Electricity, Wiley, New York.

Chapter 12

Micro-Robot Management

Wael A. Al-Tabey

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/39240

1. Introduction

Micro-Robots have been and continue to be essential components of medical field. Micro-Robots are used in medical field to surgical applications. Knowledge of the kinematics, dynamics and trajectory planning of these Micro-Robots is most important for their design and control. MATLAB is a modern tool that has transformed the mathematical calculations methods because MATLAB not only provides numerical calculations but also facilitates analytical calculations using the computer. The present textbook uses MATLAB as a tool to solve problems from micro-robots. The intent is to show the convenience of MATLAB for micro-robots analysis. Using example problems the MATLAB syntax will be demonstrated. MATLAB is very useful in the process of deriving solutions for any problem in micro-robots. The chapter includes a most problem of micro-robots for surgical applications that are being solved using MATLAB. The programs are available at the end of this chapter.

Robots are widely used in medical field for getting minimally invasive surgery efficiently and accurately. Minimally invasive surgery is an innovative approach that allows reducing patient trauma, postoperative pain and recovery time [1]. The kinematic and dynamic analysis of the robot, for any applications whether in medical field or another, are very important. The direct aim is to properly select the workspace and the actuator size. Frumento [1] designed a minimally invasive robot for heart surgery. They concentrated mainly on the kinematic analysis and the workspace of the robot. Tsai and Hsu [3] investigated a parallel surgical robot having six degrees of freedom. They studied the kinematics only, to obtain the workspace of surgical robot and control it using Fuzzy Logic control. Miller and Christensen [4] analyzed the dynamics of the Multi-rigid-body robot using Newton's second law of motion and used the results to design the controller. Featherstone and Orin [5] investigated the robot dynamics and used Newton-Euler technique to obtain the equation and algorithms of robot motion. Wang, et al [6] designed the dimensional synthesis of 6-DOF Micro-surgery manipulator (Micro Hand) .They studied the kinematics and the workspace of the manipulator they considered by an Optimal design

method. Alici and Shirinzadeh [7] obtained the singularity loci of parallel manipulators exemplified by a 3-DOF spherical parallel manipulator. They utilized the inverse kinematics, and Jacobian matrices to obtain the velocity equation of the actuator and the end effector. The determinants of the manipulator Jacobian matrices were evaluated for a specified set of geometric parameters. Finally, the singularity loci of the manipulator in Cartesian space were generated. Ben-Horin, et al [8] analyzed the kinematics and dynamics of parallel robot consisting of three plenary actuated links [6-DOF parallel manipulator]. They investigated by direct kinematic analysis and obtained the dynamics equations and algorithms using Newton's second law and presented the system performance. Bonnifait and Garcia [9] studied the 6-DOF dynamic localization of an outdoor mobile robot. They obtained the robot dynamics equations and algorithms using the numerical method (Taylor method) and used the azimuth and elevation angles of known landmarks, collected by a rotating linear camera to obtain the simulation results of the robot. They finally presented and analyzed the results of real experiments, performed with an outdoor mobile robot. Abdellatif and Heimann [10] investigated the inverse dynamics equations of 6-DOF fully parallel manipulators using the Lagrangian formalism. With respect to a proposed set of generalized coordinates and velocities they obtained the final form with respect to the robot's active coordinates. Attention was paid to the transformation of the sub chains dynamics. Finally, a systematic study of the resulting computational effort was presented and discussed with respect to results of other methods and approaches of other researches. Zhu, et al [11] investigated the kinematic and dynamic modeling for a newly developed parallel robot with the proposed Tau configuration and used the inverse kinematics method to obtain the kinematic equations of end effector of 3-DOF parallel robot. The dynamic modeling of 3-DOF parallel robot was derived by analytical solutions, which were verified by both numerical simulation and actual experiments. This analytical approach enabled the real-time control of this parallel robot with high positioning accuracy. Gouliaev and Zavrazhina [12] investigated the dynamic and kinematic control of the spatial movements of a flexible multi-link manipulator. They focused on the dynamics of a flexible multi-link manipulator by Euler-Bernoulli method, so that each element was in a compound motion. A technique for the numerical construction of solutions for an essentially non-linear hybrid-type system of constituent equations was proposed. They used the linear control method. Eliodoro and Serna [13] investigated the inverse dynamics of flexible robots. They focused on the new and general technique for solving the inverse dynamics of flexible robots. The proposed method finds the joint torques that must be applied by the actuators to obtain a specified end-effector trajectory. The inverse dynamics of flexible robots are derived by using the Euler-Bernoulli beam theory and Lagrange's equations. The finite element method was utilized to discretize space variables. They finally established the global dynamic equations of the robot. Kinematic constraints were introduced in the dynamic equations by means of a penalty formulation. The system performance was drawn for different tip trajectories.

Martins et al [14] examined an adaptive controller to guide a unicycle-like mobile robot during trajectory tracking. They concentrated, mainly on the kinematic analysis to design

the adaptive controller using Lyapunov theory. The kinematic equations of mobile robot were found by inverse kinematics method to obtain the desired values of the linear and angular velocities, which represented the input to adaptive controller. Valero et al [15] investigated a trajectory planner for 3-DOF industrial robots that have to operate in workspaces with obstacles. They found the workspace modeling analytically using the differential equations of all joints angle and solved them in Cartesian coordinates system. The trajectory planner for 3-DOF industrial robots they found by workspace model using the finite element method for joints and end effector. Integrating all elements, the function of the trajectory planner for 3-DOF industrial robots was obtained. Finally they minimized the objective function of the trajectory planner to attain the minimum time. Geng [16] investigated the dynamics and trajectory planning of a planar flipping robot with two feet and one-leg robot. Geng made a dynamic analysis to obtain the trajectory planning using Lagrange's formulation. Geng presented the simulation results for joints kinetic variables with time. Alessandro and Vanni [17] investigated a technique for optimal trajectory planning of robot manipulators to minimize the jerk. They concentrated on the trajectory planning of robot manipulators. In order to get the optimal trajectory, an objective function composed of two terms was minimized. The first term was proportional to the total execution time, and the second was proportional to the integral of the squared jerk. Finally, they minimized the time of trajectory planning to minimize energy (or actuator effort) and jerk of manipulator joints. They represented the Simulation results for joints kinetic variables and jerk against the minimized time. Pires et al [18] tested a manipulator trajectory planning with multiple objectives and obstacle avoidance (MOEA).It is a non-trivial optimization problem. They concentrated on the trajectory planning of 2-DOF robot manipulators using MOEA method and simulated for several ranges of joint angle. Finally they gave the simulation results for joints kinetic variables with minimized time. Chettibi et al [19] studied the problem of minimum cost trajectory planning for 2-DOF industrial manipulators. They studied the optimal control via direct parameter optimization of joint positions then they concentrated on the trajectory planning of 2-DOF industrial manipulators for six ranges of joint angle. Finally, they represented the simulation results for joints kinetic variables, jerk and torque with minimized time, to obtain the optimum parameters of joint positions and minimum cost trajectory planning. Alessandro and Vanni [20] investigated a method for smooth trajectory planning of robot manipulators. In order to ensure that the resulting trajectory is smooth enough they used the same method used by them in reference. They also presented in their work a new method to obtain the smooth jerk by composing the overall trajectory with respect to other trajectory optimization techniques.

1.1. Introductory remarks on robots for medical field applications

Robots in medicine are recent entry, beginning as generic instrumental aides and aiming at specialized duties once technology sophistication enables effective settings. Several classifications are used, mainly, dressing taxonomy by means of the expected accomplishments [1]:

1. Patients and disabled aid: bed automation, walking assistants, delivery servants, etc.
2. Laboratory support: clinical testers, radiation therapies assistants, etc.
3. Soundness care: pace-makers, health-monitors, drugs dosing up suppliers, etc.
4. Surgery help: surgeon's servants, remote effectors, autonomous actors, etc.

Moreover, roughly speaking, the example taxonomy distinguishes extra corporeal fixtures, mainly derived out of conventional technologies, from in-body active devices, generally requiring invasive actions, thus, critically dependent on micro-mechanics and nanotechnologies. With focus on surgery robots, four kinds of tasks are, generally, considered:

1. Organ inspection: cerebral probing, laparoscopic monitoring, etc.
2. Organ nursing or repair: internal anastomosis, obstruction relief, etc.
3. Organ removal: cysts excision, lymph node dissection, etc.
4. (Artificial) organ implant: prosthesis insertion, etc.

Surgical robotic systems are commonly classified according to the degree of direct control the surgeon has over the machine [2].Under this classification there are three principal types of robots Table 1:

1. Autonomous: performs a preoperative plan without any immediate control from the surgeon.
2. Surgical Assist Device: surgeon and robot share control.
3. Teleoperator: function of the robot completely controlled by the surgeon.

Type of system	**Examples**	**Function(s)**	**Clinical Discipline(s)**
Autonomous	Robodoc	Percutaneous renal needle placement	Orthopedic surgery
	PAKY-RCM	Prostatectomy	Urology
	ProBot	Hip surgery	Urology
Surgical Assist	Caspar	Voice controlled telescope	Orthopedic surgery
	AESOP	Stereotactic	Multiple
	NeuroMate	Neurosurgery	Neurosurgery
Teleoperator	Acrobat	Knee arthroplasty	Orthopedic surgery
	PUMA (Programmable Universal Machine for Assembly)	Multiple	Multiple
	Da Vinci	Multiple	Multiple
	Zeus	Multiple	Multiple
	Neurobot	Multiple	Neurosurgery

Table 1. Classification of robotic surgical systems

The systems in study at this present work are Zeus and Da Vinci robotic arms which are usually attached to a patient-side tower structure and consists of two to three arms that control the operative instruments and a separate arm that controls the video endoscope as shown in the Figure 1.

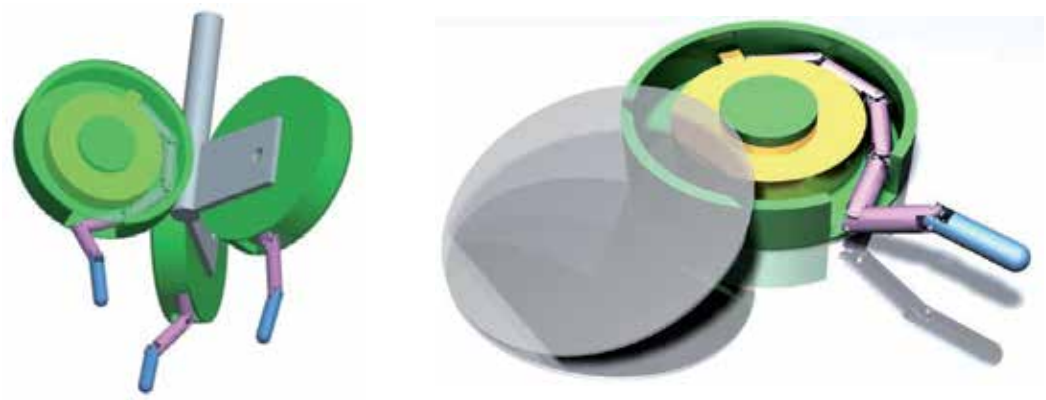


Figure 1. Zeus and Da Vinci robotic arms

Both the Zeus and Da Vinci systems enhance dexterity in several ways. Internal software filters out the natural tremor of a surgeon's hand, which becomes particularly evident under high magnification and problematic when attempting fine maneuvers in very small fields. In addition, the system can scale movements such that large movements of the control grips can be transformed into smaller movements inside the patient. Finally the system group has 7-DOF. The Surgery arm has 6-DOF plus 1-DOF for the tool actuation. Arrangement, the DOF ''a'' and ''b'' are respectively the last DOF at the carrier and the 6-DOF robot joints has Roll-Yaw-Pitch motions as shown in the Figures .2, 3 and 4.

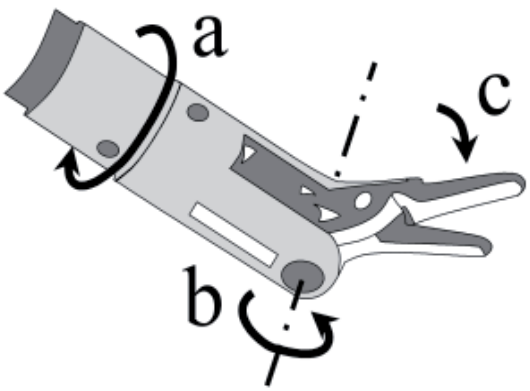


Figure 2. The ZEUS® surgery tools

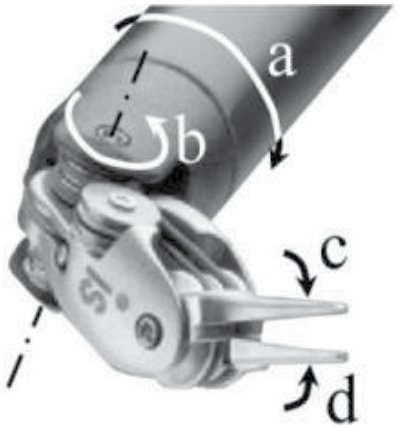


Figure 3. Da Vinci® surgery tools

Figure 4. Roll-Yaw-Pitch motions

2. The kinematic model

2.1. The forward kinematics

The forward kinematics problem is concerned with the relationship between the individual joints of the robot manipulator and the position and orientation of the tool or end effector. Stated more formally, the forward kinematics problem is to determine the position and orientation of the end effector, given the values for the joint variables of the robot. The joint variables are the angles between the links in the case of revolute or rotational joints, and the link extension in the case of prismatic or sliding joints.

The solution is always unique: one given joint position vector always corresponds to only one single end effector pose. The FK problem is not difficult to solve, even for a completely arbitrary kinematic structure.

There are different methods for a forward kinematic analysis: like using straightforward geometry and using transformation matrices.

2.1.1. The Denavit-Hartenberg convention

The D-H modeling rules:

1. Find and identify all joint axes: Z_0 to Z_{n-1}.
2. Establish the base frame. Set base origin anywhere on the Z_0 axis. Choose X_0 and Y_0 conveniently and to form a right hand frame.
3. Locate the origin O_i where the common normal to Z_{i-1} and Z_i intersects Z_i.
   If Z_i intersects Z_{i-1} locate O_i at this intersection.
   If Z_{i-1} and Z_i are parallel, locate O_i at Joint (i+1).
4. Establish X_i along the common normal between Z_{i-1} and Z_i through O_i, or in the direction normal to the plane Z_{i-1} - Z_i if these axes intersect. See Figure 5.
5. Establish Y_i to form a right hand system
6. Repeat Steps 3 to 5 for i= 1 : n-1.
7. Establish the end effector (n) frame: $O_nX_nY_nZ_n$. Assuming the nth joint is revolute.
   Set K_n = a along the direction Z_{n-1} and establish the origin O_n conveniently along Z_n, at center of tool tip. Set j_n=o in the direction of tool closure (opening) and set i_n= n, such that n=oxa.
   If the tool is not a simple gripper, set X_n and Y_n conveniently to form a right hand frame.
8. Create a table of "Link" parameters: See Figure 5.
   Joint Angle θ_i : angle between X_{i-1} and X_i about Z_i.
   Link Offset d_i : distance from X_{i-1} and X_i along Z_i.
   Link Twist α_i : angle between Z_i and Z_{i+1} about X_i.
   Link length a_i : distance between Z_i and Z_{i+1} along X_i.
9. Form HTM matrices A_1, A_2 … An from the information contained in each row of the LP table by substituting θ, d, α and a into the general model.
10. Build forward kinematic solution: $T_1^n = A_1 * A_2 * \ldots * A_n$.

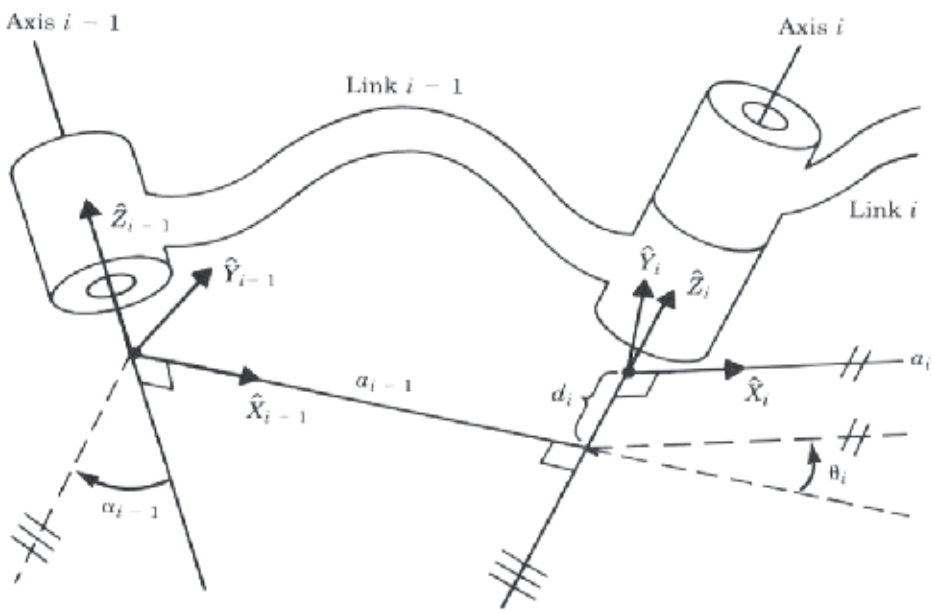


Figure 5. Construction of the link frame

2.2. The kinematics of Zeus and Da Vinci robotic arms

The geometrical model of the surgical robot in this study has 6 degrees of freedom (DOF) and an extra one for tool action. The end-effector has 3 rotations (Roll, Pitch and Yaw) as shown in Figure 6 and the frame assignment of 6-DOF surgical manipulator is represented in Figure 7.

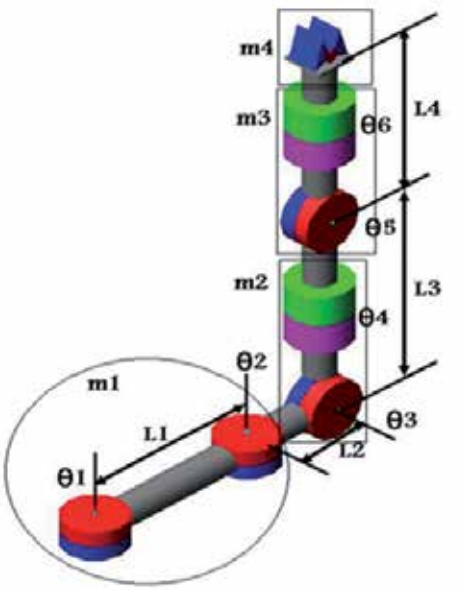


Figure 6. The geometrical model of surgical manipulator

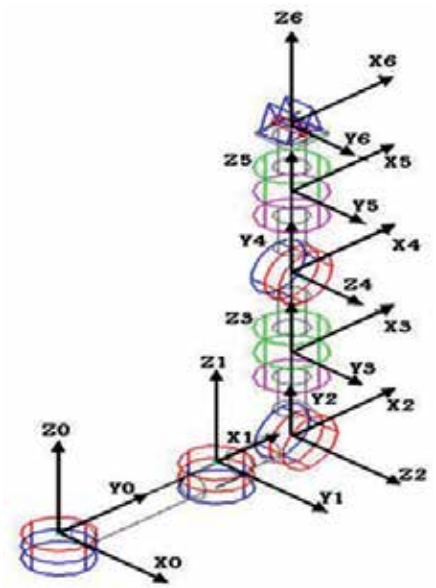


Figure 7. The frame assignment of 6-DOF surgical manipulator

The 6-DOF manipulator kinematic parameters are derived using Denavit Hartemberg formulation shown in Table 2.

Link #	θ_j	d_j	a_j	α_j	T
1	θ_1	0	L_1	0	${}^0T_1$
2	θ_2	0	L_2	90	${}^1T_2$
3	θ_3	0	0	-90	${}^2T_3$
4	θ_4	L_3	0	90	${}^3T_4$
5	θ_5	0	0	-90	${}^4T_5$
6	θ_6	L_4	0	0	${}^5T_6$

Table 2. Full mobility robotic tool: geometry and link parameters

The flowchart representing the sequence of generating the MATLAB CODE of the link transformations matrix is shown in Figure 8.

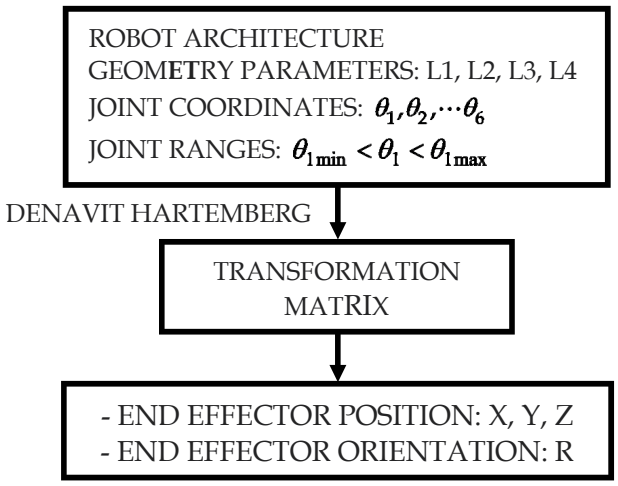


Figure 8. The flowchart of the link transformations matrix

The general form of the Homogeneous Transformation Matrix is:

$$ {}^{j-1}T_j = \begin{bmatrix} \cos\theta_j & -\sin\theta_j\cos\alpha_j & \sin\theta_j\sin\alpha_j & a_j\cos\theta_j \\ \sin\theta_j & \cos\theta_j\cos\alpha_j & -\cos\theta_j\sin\alpha_j & a_j\sin\theta_j \\ 0 & \sin\alpha_j & \cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

The link transformations matrix can be given as:

$$ {}^0T_1 = \begin{bmatrix} C_1 & -S_1 & 0 & L_1C_1 \\ S_1 & C_1 & 0 & L_1S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^1T_2 = \begin{bmatrix} C_2 & 0 & S_2 & L_2C_2 \\ S_2 & 0 & -C_2 & L_2S_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^2T_3 = \begin{bmatrix} C_3 & 0 & -S_3 & 0 \\ S_3 & 0 & C_3 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}T_4 = \begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & L_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{4}T_5 = \begin{bmatrix} C_5 & 0 & -S_5 & 0 \\ S_5 & 0 & C_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{5}T_6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & L_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The kinematics equations of the end effectors are manipulated using MATLAB symbolic Toolbox were as follows:

$$ \begin{bmatrix} P_X \\ \\ P_Y \\ \\ P_Z \\ \\ Z \end{bmatrix} = \begin{bmatrix} \begin{array}{c} ((-(C_1*C_2-S_1*S_2)*C_3*C_4-(-C_1*S_2-S_1*C_2)*S_4)*S_5+ \\ (C_1*C_2-S_1*S_2)*S_3*C_5)*L_4-(C_1*C_2-S_1*S_2)*S_3*L_3+ \\ C_1*L_2*C_2-S_1*L_2*S_2+L_1*C_1 \\ \\ ((-(C_1*S_2+S_1*C_2)*C_3*C_4-(C_1*C_2-S_1*S_2)*S_4)*S_5+ \\ (C_1*S_2+S_1*C_2)*S_3*C_5)*L_4-(C_1*S_2+S_1*C_2)*S_3*L_3+ \\ S_1*L2*C_2+C_1*L_2*S_2+L_1*S_1 \\ \\ (-S_3*C_4*S_5-C_3*C_5)*L_4+C_3*L_3 \\ \\ 1 \end{array} \end{bmatrix} $$

2.3. The robot workspace

The workspace of a robot can be defined as the set of points that are reachable by the manipulator (with fixed base). Roughly speaking the workspace is the volume of space which the end effector of the robot can reach. Both shape and total volume are important. Workspace is also called work volume or work envelope.

The workspace depends on the characteristics of the manipulator; physical configurations, size, number of axes, the robot mounted position (overhead gantry, wall-mounted, floor mounted, on tracks, etc), limits of arm and joint configurations. The addition of an end effector can move or offset the entire work volume.

The kinematics design of a manipulator can tailor the workspace to some extent to the operational requirements of the robot.

Some robots will have unusable spaces such as dead zones, singular poses, and wrist-wrap poses inside of the boundaries of their reach. Elbow manipulators tend to have a wider volume of workspace.

1. Dexterous workspace: This is the volume of space which the end-effector of the manipulator can reach with all orientations.
2. Reachable workspace: This is the volume of space which the end-effector of the manipulator can reach with at least one orientation.

The dexterous workspace is obviously a subset of the reachable workspace.

2.3.1. The workspace calculation

The workspace may be found mathematically by writing equations that define the robot's links and joints and including their limitations, such as ranges of motions for each joint. Alternatively, the workspace may be found empirically, by moving each joint through each range of motion and combining all the space in can reach and subtracting what it cannot reach.

The workspace of the surgical manipulator can be represented by solving the inverse kinematic equations and taking into consideration all the physical limits of the joints. Figure 9 represents the flowchart showing the sequence of generating the three dimensional workspace of the robot and end-effector to be manipulated using the MATLAB symbolic Toolbox.

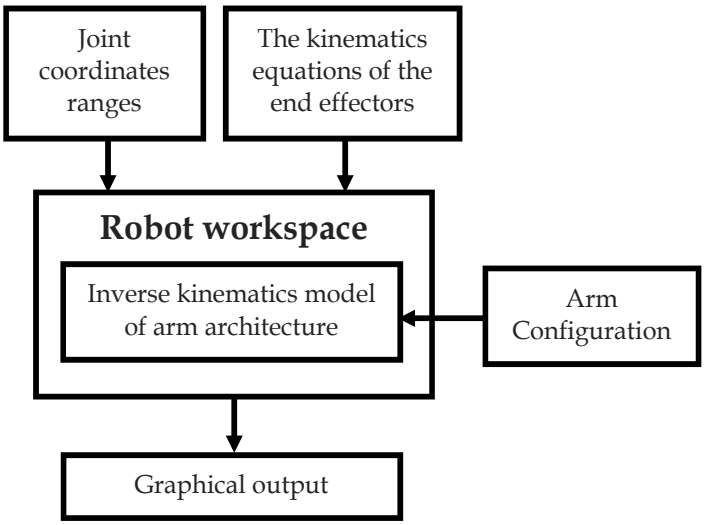


Figure 9. The flowchart of The Robot workspace

Table 3 represent the physical limits of the six joints while Figure10 represent the workspace of end-effector respectively.

Link #	1	2	3	4	5	6
θ_i (degree)	-180°	-90º	0	-180°	-90º	-180°
θ_f (degree)	180°	90º	180°	180°	90º	180°
L (mm)	0	50	15	36	0	39.5

Table 3. Joint coordinates ranges

The workspace of the end-effector depends on the physical limits of the six joints angle, i.e. if the angle range of the robot joints is changes the workspace of the robot changes. So it is important to take into account the accuracy in determining the angle range of the robot joints to get the required workspace which covers the work area.

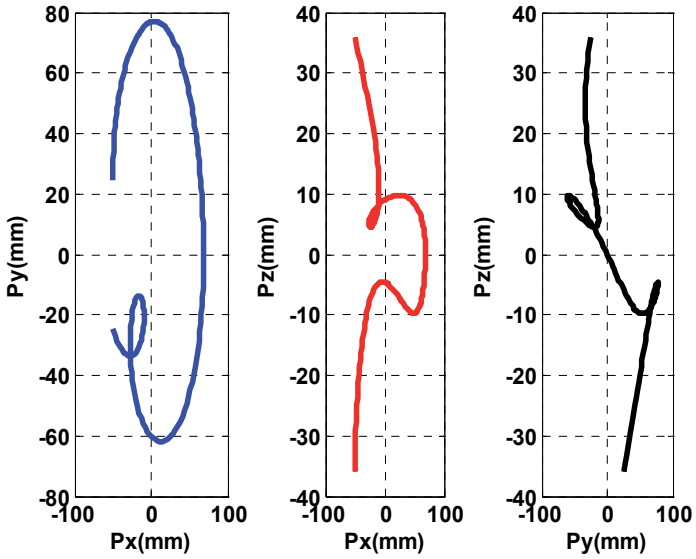


Figure 10. The work space of end effector

2.4. The robot Jacobian

The Jacobian is a representation of the geometry of the elements of mechanism in time. It allows the conversion of differential motions or velocities of individual joints to differential motions or velocities of pints of interest. It also relates the individual joint motion to overall mechanism motions. Jacobian is time related; since the values of θ_i vary in time, and the magnitude of the elements of Jacobian vary in time as well.

2.4.1. The differential motions and velocities equations

Differential motions are small movements of robot parts that can be used to derive velocity relationships between different parts of the robot. To find these relations the following steps, are to be considered:

1. Frames relative to a fixed frame.
2. Robot joint relative to a fixed frame.
3. Jacobian matrix.
4. Robot velocity relationship.

2.4.2. The Jacobian equations

Suppose we have a set of equations Y_i in terms of variables X_j

$$Y_i = f_i(X_1, X_2, X_3 \cdots, X_j) \tag{2}$$

The differential change in Y_i for a differential change in X_j is:

$$
\begin{aligned}
\delta Y_1 &= \frac{\partial f_1}{\partial X_1}\delta X_1 + \frac{\partial f_1}{\partial X_2}\delta X_2 + \cdots\cdots\cdots\cdots + \frac{\partial f_1}{\partial X_i}\delta X_j \\
\delta Y_2 &= \frac{\partial f_2}{\partial X_1}\delta X_1 + \frac{\partial f_2}{\partial X_2}\delta X_2 + \cdots\cdots\cdots\cdots + \frac{\partial f_2}{\partial X_i}\delta X_j \\
\vdots \quad & \quad \vdots \qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots \\
\vdots \quad & \quad \vdots \qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots \\
\delta Y_i &= \frac{\partial f_i}{\partial X_1}\delta X_1 + \frac{\partial f_i}{\partial X_2}\delta X_2 + \cdots\cdots\cdots\cdots + \frac{\partial f_i}{\partial X_i}\delta X_j
\end{aligned}
\tag{3}
$$

2.4.3. The Jacobian matrix

$$
\left[\delta Y_i\right] = \left[\frac{\partial f_i}{\partial X_i}\right]\left[\delta X_j\right] \equiv \left[D\right] = \left[J\right]\left[D_\theta\right] \tag{4}
$$

2.4.4. The Jacobian relations

The Kinematics equations of the end effectors which are used to calculate the Jacobian matrix of the robot are:

$$
P_X = f_1 = ((-(C_1*C_2-S_1*S_2)*C_3*C_4-(-C_1*S_2-S_1*C_2)*S_4)*S_5+(C_1*C_2-S_1*S_2)*S_3*C_5)*L_4-(C_1*C_2-S_1*S_2)*S_3*L_3+C_1*L_2*C_2-S_1*L_2*S_2+L_1*C_1
$$

$$
P_Y = f_2 = ((-(C_1*S_2+S_1*C_2)*C_3*C_4-(C_1*C_2-S_1*S_2)*S_4)*S_5+(C_1*S_2+S_1*C_2)*S_3*C_5)*L_4-(C_1*S_2+S_1*C_2)*S_3*L_3+S_1*L2*C_2+C_1*L_2*S_2+L_1*S_1
$$

$$
P_Z = f_3 = (-S_3*C_4*S_5-C_3*C_5)*L_4+C_3*L_3
$$

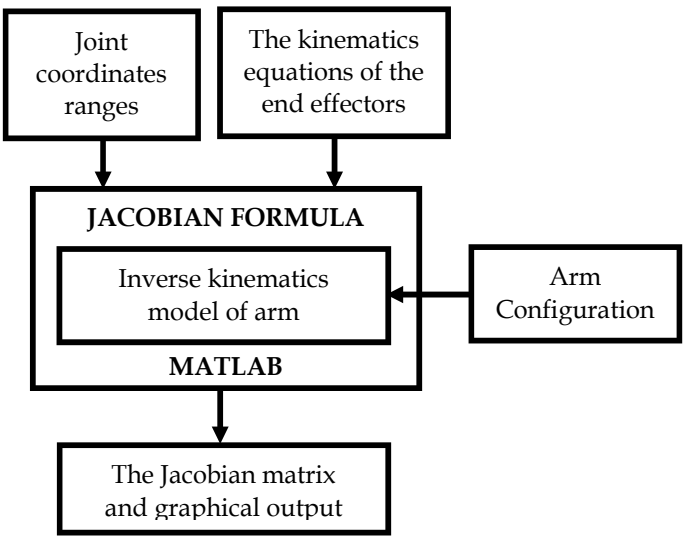


Figure 11. The flowchart of the Jacobian matrix

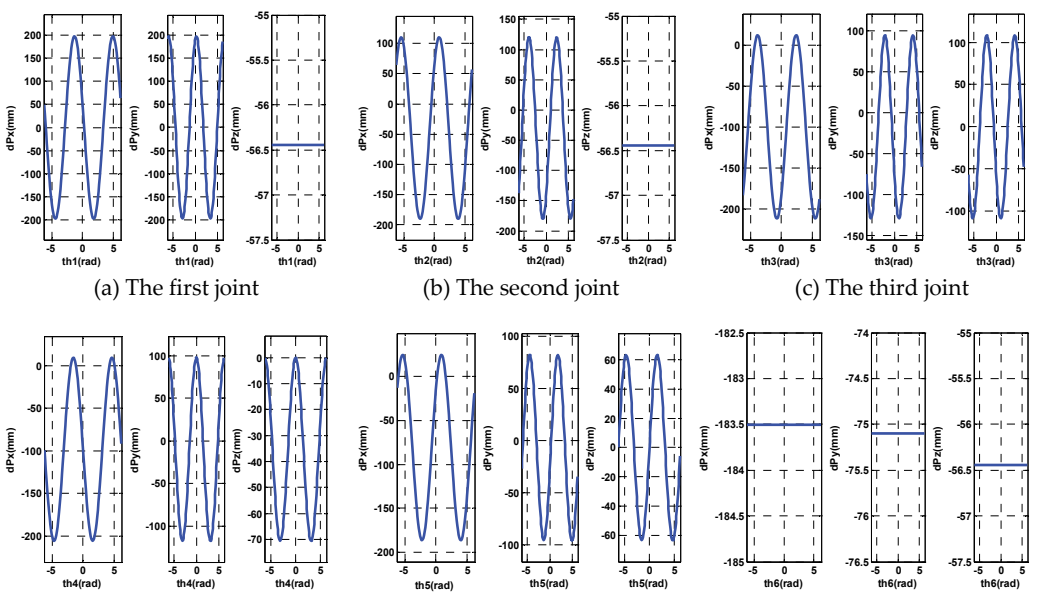
The Jacobian matrix of the robot can be calculated by MATLAB symbolic Toolbox using the kinematics equations as shown in the Figure 11.

The differential equations of motion of the end effector for the surgical robot are represented by:

$$
\begin{aligned}
dP_x = V_x = {} & ((((C_1*S_2+S_1*C_2)*C_3*C_4-(-C_1*C_2+S_1*S_2)*S_4)*S_5+(-C_1*S_2-S_1*C_2)*S_3*C5*L_4- \\
& (-C_1*S_2-S_1*C_2)*S_3*L_3-S_1*L_2*C_2-C_1*L_2*S_2-L_1*S_1)*d\theta_1+((((C_1*S_2+S_1*C_2)*C_3*C_4- \\
& (-C_1*C_2+S_1*S_2)*S_4)*S_5+(-C_1*S_2-S_1*C_2)*S_3*C_5)*L_4-(-C_1*S_2-S_1*C_2)*S_3*L_3-S_1*L_2*C_2- \\
& C_1*L_2*S_2)*d\theta_2+(((C_1*C_2-S_1*S_2)*S_3*C_4*S_5+(C_1*C_2-S_1*S_2)*C_3*C_5)*L_4- \\
& (C_1*C_2-S_1*S_2)*C_3*L_3)*d\theta_3+((C_1*C_2-S_1*S_2)*C_3*S_4-(-C_1*S_2-S_1*C_2)*C_4)*S_5*L_4*d\theta_4+ \\
& (((-C_1*C_2+S_1*S_2)*C_3*C_4-(-C_1*S_2-S_1*C_2)*S_4)*C_5-(C_1*C_2-S_1*S_2)*S_3*S_5)*L_4*d\theta_5
\end{aligned}
$$

$$
\begin{aligned}
dP_y = V_y = {} & ((((-C_1*C_2+S_1*S_2)*C_3*C_4-(-C_1*S_2-S_1*C_2)*S_4)*S_5+(C_1*C_2-S_1*S_2)*S_3*C_5)*L_4- \\
& (C_1*C_2-S_1*S_2)*S_3*L_3+C_1*L_2*C_2-S_1*L_2*S_2+L_1*C_1)*d\theta_1+((((-C_1*C_2+S_1*S_2)*C_3*C_4- \\
& (-C_1*S_2-S_1*C_2)*S_4)*S_5+(C_1*C_2-S_1*S_2)*S_3*C_5)*L_4-(C_1*C_2-S_1*S_2)*S_3*L_3+C_1*L_2*C_2- \\
& S_1*L_2*S_2)*d\theta_2+(((C_1*S_2+S_1*C_2)*S_3*C_4*S_5+(C_1*S_2+S_1*C_2)*C_3*C_5)*L_4- \\
& (C_1*S_2+S_1*C_2)*C_3*L_3)*d\theta_3+((C_1*S_2+S_1*C_2)*C_3*S_4-(C_1*C_2-S_1*S_2)*C_4)*S_5*L_4*d\theta_4+ \\
& (((-C_1*S_2-S_1*C_2)*C_3*C_4-(C_1*C_2-S_1*S_2)*S_4)*C_5-(C_1*S_2+S_1*C_2)*S_3*S_5)*L_4*d\theta_5
\end{aligned}
$$

$$
dP_z = V_z = ((-C_3*C_4*S_5+S_3*C_5)*L_4-S_3*L_3)*d\theta_3+S_3*S_4*S_5*L_4*d\theta_4+(-S_3*C_4*C_5+C_3*S_5)*L_4*d\theta_5
$$



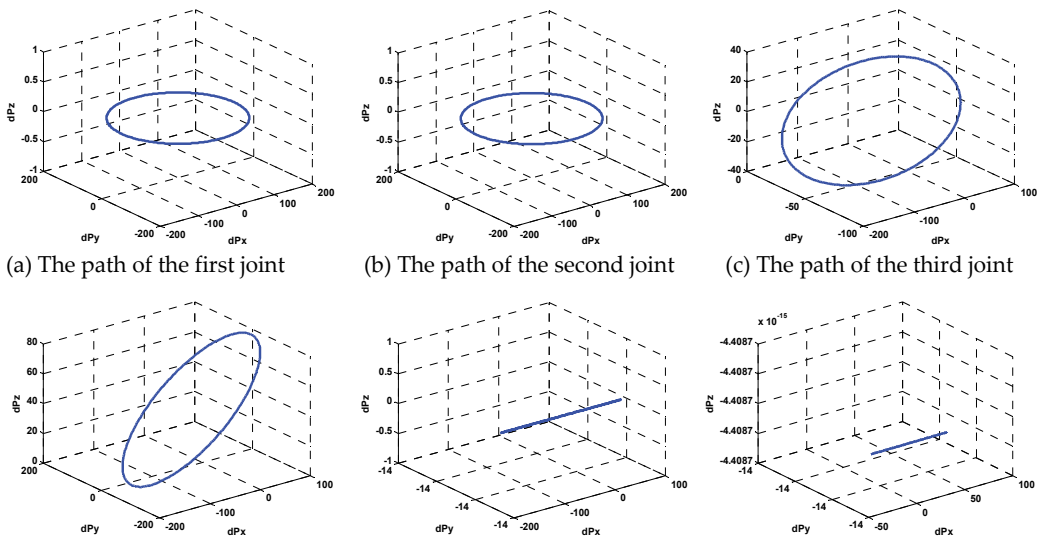
(a) The first joint (b) The second joint (c) The third joint

(d) The forth joint (e) The fifth joint (f) The sixth joint

Figure 12. The relation between angle of joints and the end effector differential translation

The previous differential equations of motion of the end effector represent the relation between the magnitude of the elements of Jacobian (the elements of end effector motion) and the joints angle. Figures 12 a, b, c, d, e and f represents the relation between angle of joints and the end effector differential translation. This shows six figures is (a, b, c, d, e and f) for six joints. It is to be noted that in Figures 12-a and 12-b for the first and second joints respectively the relation between angle of joint and the end effector differential translation in the Z direction is constant i.e. the motion of the end effector in the Z direction is not affected by the change in angle of the first and second joints. In the same time the motion of the end effector in all directions (X, Y and Z) is not affected by the change in angle of the six joints because the frame of the end effector is in the same directions of the frame of the joint number six. This can clearly be inferred from the geometrical model and the frame of the robot given in Figures 6 and 7 respectively.

The relations between the magnitude of the elements of Jacobian (the elements of end effector motion at all directions) i.e. dP_X, dP_Y and dP_Z give of the path of the robot joints as shown in Figures 13 a, b, c, d, e and f.



(a) The path of the first joint (b) The path of the second joint (c) The path of the third joint

(d) The path of the forth joint (e) The path of the fifth joint (f) The path of the sixth joint

Figure 13. The path of the robot joints

3. The trajectory planning

3.1. Introduction to the trajectory planning

Robot study is divided into two parts; they are the kinematics and dynamics. This means that using the equations of motion of the robot, its position can be determined if the joint variables are known. Path and trajectory planning relates the way a robot is moved from one location to another in controlled manner. In this chapter, a study of the sequence of

movements is to be made to create a controlled movement between motion segments, in straight-line motion, or in sequential motions. Path and trajectory planning requires the use of both kinematics and dynamics of robots. In practice, precise motion requirements are so intensive that approximations are always necessary [23].

What is the different between the path and the trajectory planning?

A Path is defined as a sequence of robot configurations in particular order without regard to timing of these configurations as shown in Figure 14. A trajectory is concerned about when each part of the path must be obtained thus specifying timing (Velocity and Acceleration). Or a trajectory is a spatial position/time curve that usually represents a desired manipulation motion in either link or Cartesian space as shown in Figure 15.

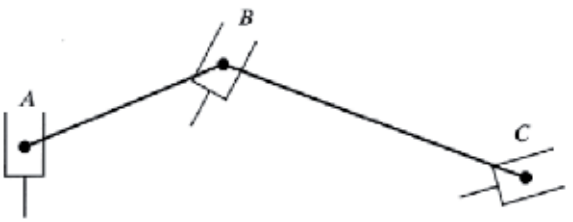


Figure 14. Sequential robot movements in a path [23]

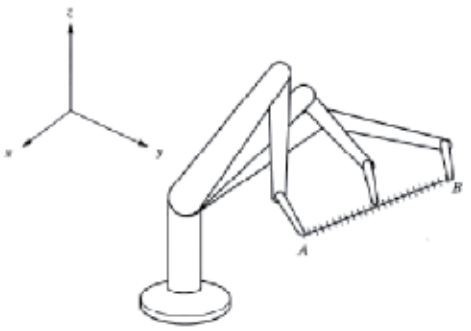


Figure 15. Sequential motions of a robot to follow a straight line [23]

3.2. The methods to calculate trajectory planning

There are four different methods which have been derived to calculate the trajectory planning of the robot using MATLAB code. The trajectory planning in the four methods needs two parameters, namely the joint angle range and the final time required to complete the process. The first one was selected from the workspace needed to complete the process In any case, this dependent on the type of surgical applications and can easily be identified. The final time required is a very important parameter to derive the trajectory planning of the robot. As all the results that have been inferred from the orientation, velocity, acceleration and torque of the robot are based on this parameter. If the time increases the velocity required decreases and acceleration i.e. the inertia of the

robot link consequently decreases accidentally irrespective of the methods used. The orientation must be seen carefully examined and the behaviour should be increasing gradually with the time from the initial angle to the final angle. The final time required to complete the process for each method should be optimized and this is called optimal planning. In this work the trajectory planning of the four times is derived and then, comparison of the results at specific joint is made. The proper time which gives satisfactory results is obtained.

3.2.1. Third-Order polynomial trajectory planning

The most common techniques for trajectory planning for industrial robots are the use of polynomial of different orders, such as Cubic and B-splines, linear segments with parabolic blends and the soft motion trajectory [22]. The Linear Segments with Parabolic blends trajectories are faster and more suitable for industrial applications. On the other hand, the higher order polynomials trajectory as well as the soft motion trajectory [24] are easy to design and control especially for the jerk. They are accurate, precise and suitable for medical applications. In this work, the trajectory planning for each joint is designed using third order polynomial as a rest-to-rest manoeuvring where the link starts from rest, accelerates and decelerates at the end of the trajectory. The trajectory is given by [23]:

$$\theta(t) = C_0 + C_1 t + C_2 t^2 + C_3 t^3 \tag{5}$$

In which C_0, C_1, C_2 and C_3 are coefficients to be determined from the initial conditions as follows:

$$\dot{\theta}(\mathrm{t}) = \mathrm{C}_1 + 2\,\mathrm{C}_2\,\mathrm{t} + 3\,\mathrm{C}_3\,\mathrm{t}^2 \tag{6}$$

$$\dot{\theta}(\mathrm{t_i}) = 0 \text{ and } \dot{\theta}(\mathrm{t_f}) = 0 \qquad \text{(Rest to Rest manouvering)}$$

By substituting in to equation (6):

$$\dot{\theta}(\mathrm{t_i}) = \mathrm{C}_1 = 0$$

$$\dot{\theta}(\mathrm{t_f}) = \mathrm{C}_1 + 2\,\mathrm{C}_2\,\mathrm{t_f} + 3\,\mathrm{C}_3\,\mathrm{t_f}^2 = 0 \tag{7}$$

The initial and final location and orientation of robot are known from:

$$\theta(\mathrm{t_i}) = \theta_\mathrm{i} \text{ and } \theta(\mathrm{t_f}) = \theta_\mathrm{f}$$

By substituting in to equation (5):

$$\theta(\mathrm{t_i}) = \mathrm{C}_0 = \theta_i$$

$$\theta(\mathrm{t_f}) = C_0 + C_1 t_f + C_2 t_f^{\,2} + C_3 t_f^{\,3} = \theta_f \tag{8}$$

By solving equations (7), (8) simultaneously:

$$C_2 = 3(\theta_f - \theta_i)/t_f^{\,2} \text{ and } C_3 = 2(\theta_f - \theta_i)/t_f^{\,3}$$

3.2.2. Fifth-Order polynomial trajectory planning

The third order trajectory only takes into account starting and end velocities. The equations of the fifth order polynomial takes into account starting and end accelerations. In this case, the total number of boundary conditions are six, allowing a fifth.

The initial and final velocities are zero. (Rest to Rest manoeuvring) and the trajectory is given by [23]:

$$\theta(t) = C_0 + C_1 t + C_2 t^2 + C_3 t^3 + C_4 t^4 + C_5 t^5 \tag{9}$$

In which C_0, C_1, C_2, C_3, C_4 and C_5 are coefficients to be determined from the initial and final conditions as follows:

$$\dot{\theta}(t) = C_1 + 2C_2 t + 3C_3 t^2 + 4C_4 t^3 + 5C_5 t^4 \tag{10}$$

$$\ddot{\theta}(t) = 2C_2 + 6C_3 t + 12C_4 t^2 + 20C_5 t^3 \tag{11}$$

$$\dot{\theta}(\mathrm{t_i}) = 0 \text{ and } \dot{\theta}(\mathrm{t_f}) = 0 \qquad \text{(Rest to Rest manoeuvring)}$$

$$\ddot{\theta}(\mathrm{t_i}) = \ddot{\theta}_i \text{ and } \ddot{\theta}(\mathrm{t_f}) = \ddot{\theta}_f$$

$$C_0 = \theta_i \;\; , C_1 = \dot{\theta}_i \,,\; C_2 = \frac{\ddot{\theta}_i}{2},\; C_3 = \frac{20\theta_\mathrm{f} - 20\theta_\mathrm{i} - (8\dot{\theta}_\mathrm{f} + 12\dot{\theta}_\mathrm{i})\,\mathrm{t_f} - (3\ddot{\theta}_\mathrm{i} - \ddot{\theta}_\mathrm{f})\mathrm{t_f}^2}{2t_f^{\,3}},$$

$$C_4 = \frac{30\theta_\mathrm{f} - 30\theta_\mathrm{i} + (14\dot{\theta}_\mathrm{f} + 16\dot{\theta}_\mathrm{i})\,\mathrm{t_f} + (3\ddot{\theta}_\mathrm{i} - 2\ddot{\theta}_\mathrm{f})\mathrm{t_f}^2}{2t_f^{\,4}} \text{ and } C_5 = \frac{12\theta_\mathrm{f} - 12\theta_\mathrm{i} - (6\dot{\theta}_\mathrm{f} + 6\dot{\theta}_\mathrm{i})\,\mathrm{t_f} + (\ddot{\theta}_\mathrm{i} - \ddot{\theta}_\mathrm{f})\mathrm{t_f}^2}{2t_f^{\,5}}$$

3.2.3. Linear segments with parabolic blends

Linear segments can be blended with parabolic sections at the beginning and the end of the motion segment, creating continuous position and velocity. Acceleration is constant for the parabolic sections, yielding a linear velocity at the common points A and B as shown in the Figure 16.

3.2.3.1. First parabolic blends (t_0 to t_a)

$$\theta(\mathrm{t}) = \mathrm{C_0} + \mathrm{C_1 t} + \frac{1}{2}\,\mathrm{C_2 t^2} \tag{12}$$

$$\dot{\theta}(t) = C_1 + C_2 t \text{ and } \ddot{\theta}(t) = C_2$$

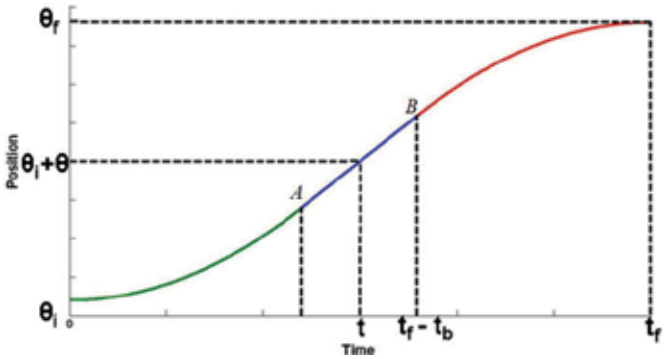


Figure 16. Scheme for linear segments with parabolic blends

The position of the robot at time t0 is known and using the inverse Kinematic equations of the robot, the joint angles at via points and at the end of the motion can be found.

To blend the motion segments together, the boundary conditions at each point are used to calculate the coefficients of the parabolic segments.

3.2.3.2. Straight line (t_a to t_b)

$$\theta_A = \theta_B + wt \text{ , } \dot{\theta}_A = \dot{\theta}_B + w \text{ and } \ddot{\theta} = 0$$

3.2.3.3. Second parabolic blends ($t_f - t_b$ to t_f)

$$\theta(t) = C_f + \frac{w}{2t_b}(t_f - t)^2 \tag{13}$$

$$\dot{\theta}(t) = \frac{w}{t_b}(t_f - t) \text{ and } \ddot{\theta}(t) = -\frac{w}{t_b}$$

$$t_b = \frac{(\theta_i - \theta_f + wt_f)}{w} \text{ , } C_2 = \frac{w}{t_b} \text{ and } w_{max} = \frac{2(\theta_f - \theta_i)}{t_f}$$

3.2.4. Soft motion trajectory planning

In this method we consider the trajectory planning of points generated by a motion planning technique. The motion planner calculates the trajectory which the end effector must follow in space. However, the temporal characteristics of this movement are independent. One important difference between industrial robotic manipulators and service robot applications is the human interaction, which introduce safety and comfort constraints. In this work, we define soft motions conditions to facilitate this cohabitation. An on-line trajectory planner is proposed here. It generates the necessary references to produce soft

motion and a control loop that guarantees the end effector's motion characteristics (jerk, acceleration, velocity and position) in the Cartesian space, by using quaternion feedback. Two visual feedback control loops are proposed: a visual servoing control loop in a shared position - vision schema and a visual guided loop (which is the general case of soft motion trajectory) are given by:

3.2.4.1. The motion with a maximum jerk (J_{max})

$$\theta(t) = \theta_0 + \dot{\theta}_0 \mathrm{t} + \frac{1}{2}\ddot{\theta}t^2 + \frac{1}{6}J_{\max}t^3 \tag{14}$$

$$\dot{\theta}(t) = \dot{\theta}_0 + \ddot{\theta}_0 \mathrm{t} + \frac{1}{2}\mathrm{J}_{\max}t^2 \,,\ \ddot{\theta}(t) = \ddot{\theta}_0 + \mathrm{J}_{\max} \text{ and } \mathrm{J(t)} = \mathrm{J}_{\max}$$

3.2.4.2. The motion with a maximum acceleration (A_{max})

$$\theta(t) = \theta_0 + \dot{\theta}_0 \mathrm{t} + \frac{1}{2}\ddot{\theta}_{\max}t^2 \tag{15}$$

$$\dot{\theta}(t) = \dot{\theta}_0 + \ddot{\theta}_{\max} \,,\ \ddot{\theta}(t) = \ddot{\theta}_{\max} \text{ and } \mathrm{J(t)} = 0$$

3.2.4.3. Finally, the equations for the motion with a maximum velocity (V_{max})

$$\theta(t) = \theta_0 + \dot{\theta}_{\max}\mathrm{t} \tag{16}$$

$$\dot{\theta}(t) = \dot{\theta}_{\max} \,,\ \ddot{\theta}(t) = 0 \text{ and } \mathrm{J(t)} = 0$$

The initial conditions are:

$$\ddot{\theta}(0) = 0 \,,\ \dot{\theta}(0) = 0 \,,\ \theta(0) = \theta_0 \text{ and } D = \theta_D - \theta_0$$

Where: D is general traversed angular movement.

The final conditions are:

$$\ddot{\theta}(t_f) = 0 \,,\ \dot{\theta}(t_f) = 0 \text{ and } \theta(t_f) = \theta_D$$

We have two limit conditions to obtain the traversed angular movement:

- **Condition (1):**

Where $\dot{\theta}_{\max}$ is reached. It means, $\ddot{\theta}_{\max}$ is reached too. Then we have to find the traversed angular (Dthr1). The limit times used to calculate (Dthr1) are:

$$T_j = T_{j_{\max}} \,,\ T_a = T_{a_{\max}} \text{ and } T_v = 0$$

The angular movement (Dthr1) becomes:

$$\text{Dthr1} = \frac{\ddot{\theta}_{\max}\dot{\theta}_{\max}}{J_{\max}} + \frac{\dot{\theta}_{\max}{}^{2}}{\ddot{\theta}_{\max}}$$

- **Condition (2):**

Where only $\ddot{\theta}_{\max}$ is reached. Then we have to find the traversed angular (Dthr2). The limit times used to calculate (Dthr2) are:

$$T_j = T_{j_{\max}}\ ,\ \ T_a = 0 \ \text{ and } \ T_v = 0$$

The angular movement (Dthr2) becomes:

$$\text{Dthr2} = 2\frac{\ddot{\theta}^{3}_{\max}}{J^{2}_{\max}},\ T_{j_{\max}} = \frac{\ddot{\theta}_{\max}}{J_{\max}} \ \text{ and } \ T_{a_{\max}} = \left(\frac{\dot{\theta}_{\max}}{\ddot{\theta}_{\max}}\right) - \left(\frac{\ddot{\theta}_{\max}}{J_{\max}}\right)$$

The soft motion trajectory planning of 6-DOF surgical robot is divided in to three cases depending on the maximum Jerk algorithm as:

- **Case (1) (General case):** If $\text{D} \geq \text{Dthr1}$

$$T_j = T_{j_{\max}}\ ,\ \ T_a = T_{a_{\max}} \ \text{ and } \ T_v = \frac{D - \text{Dthr1}}{\dot{\theta}_{\max}}$$

- **Case (2):** If $\text{Dthr2} > \text{D} \geq \text{Dthr2}$

$$T_v = 0\ ,\ \ T_j = T_{j_{\max}} \ \text{ and } \ T_a = \sqrt{\frac{\ddot{\theta}^{2}_{\max}}{4J_{\max}} + \frac{D}{\ddot{\theta}_{\max}}} - \frac{3\ddot{\theta}_{\max}}{2J_{\max}}$$

- **Case (3):** If $\text{D} < \text{Dthr2}$

$$T_v = 0\ ,\ T_a = 0 \ \text{ and } \ T_j = \sqrt[3]{\frac{D}{2J_{\max}}}$$

3.3. The robot trajectory planning parameters

In this work, four different methods are applied here to design the joints trajectories third order polynomial, fifth order polynomial, linear segments with parabolic blends and soft motion trajectory. The trajectories have the same initial and final angles and different four duration times (5, 10, 20 and 60 s) are applied to choose the best duration time give the correct dynamic response. As well but they differ in the acceleration and the jerk. After designing the joints trajectories, the hub torques of the robot actuators can be simulated using MATLAB. The parameters of six joints were obtained using inverse kinematics analysis as shown in Table 4.

The flowchart representing the sequence of generating the trajectory planning is shown in the Figure 17.

Link#	1	2	3	4	5	6
θ_i(degree)	-180°	-90º	0	-180°	-90º	-180°
θ_f (degree)	180°	90º	180°	180°	90º	180°
L (mm)	0	50	15	36	0	39.5
$\ddot{\theta}_i$ (degree /s2)	5	5	5	5	5	5
$\ddot{\theta}_f$ (degree /s2)	-5	-5	-5	-5	-5	-5
$\dot{\theta}_{max}$(degree /s)	135	70	70	135	70	135
$\ddot{\theta}_{max}$ (degree /s2)	80	40	40	80	40	80
J_{max} (degree /s3)	160	80	80	160	80	160
t_i (s)	0	0	0	0	0	0
t_f (s)	5	5	5	5	5	5

Table 4. Full robotic mobility information used in trajectory planning

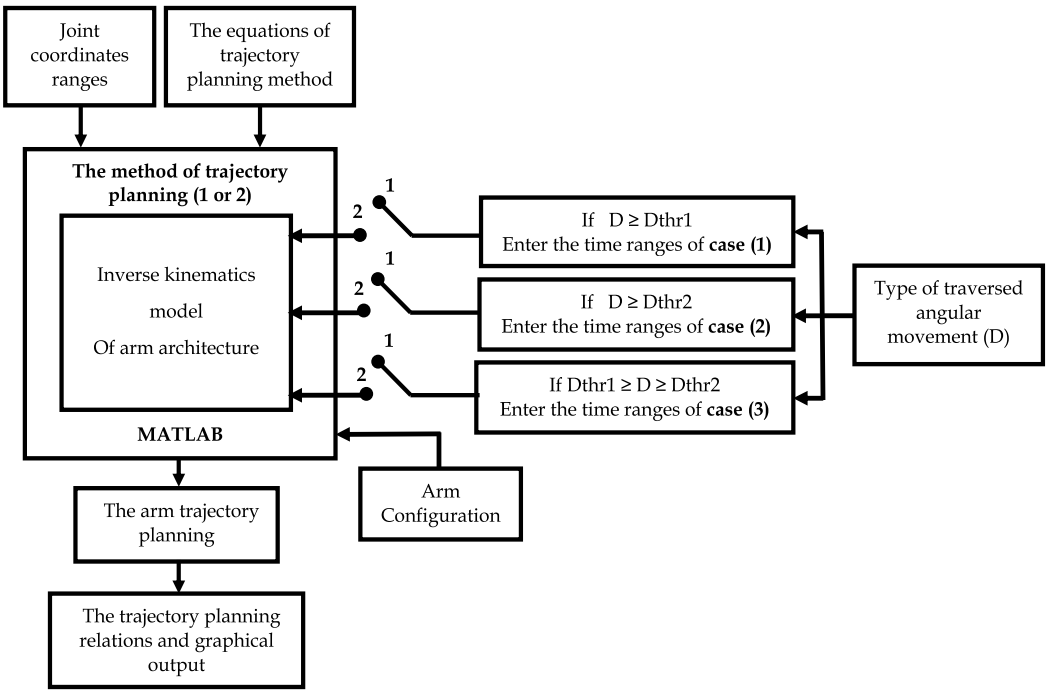


Figure 17. The flowchart of the trajectory planning

Where: Position.1 for first methods of trajectory planning and Position .2 for Soft motion trajectory planning.

4. The dynamic model

4.1. Introduction to the dynamic model of robot

Manipulator dynamics is concerned with the equations of motion, the way in which the manipulator moves in response to torques applied by the actuators, and external forces. The

history and mathematics of the dynamics of serial-link manipulators are well covered in the literature. The equations of motion for an n-axis manipulator are given by [25]:

$$Q=M(q)\,\ddot{q}+C(q.\dot{q}\ )\,\dot{q}+G(q) \tag{17}$$

These equations of motion can be derived by six methods, namely the Newton's second law method, D'Alembert method, Lagrange method, Hemilton method, Lagrange-Euler method and Newton-Euler method. But not all previous methods can be used to derive the equations of motion for the robot subject of this work. That is not all methods can easily derived the equations of motion for a robot having multi-degree of freedom.

4.2. Dynamic equations for multiple-degree of freedom robots

As we have stated, the dynamic equation for two-degree of freedom system is much more complicated than a one-degree of freedom system. Similarly, these equations for multiple-degree of freedom robot are cumbersome and complicated, but can be found by calculating the kinetic and potential energies of the links and joints [25]. For the robot considered in this work the robot has 6-DOF and the most appropriate method to derive the equations of motion is likely to be Lagrange-Euler technique.

4.2.1. Derivation of the equations of motion by Lagrange-Euler technique

The Lagrange-Euler technique is utilized here to calculate the kinetic energy, potential energy and to derive the dynamic equations in symbolic form using the MATLAB symbolic toolbox for the six-degree of freedom robot. The equations of motion are given in a concise form similar to that given in [25].

4.2.1.1. The total kinetic energy

The total kinetic energy of multiple-degree of freedom robot is given in a concise form as:

$$K_i=\frac{1}{2}\sum_{i=1}^{n}\sum_{p=1}^{i}\sum_{r=1}^{i}Trace(U_{ip}J_iU_{ir}T)\dot{q}_p\dot{q}_r+\frac{1}{2}\sum_{1}^{n}I_{iact}\dot{q}_i^{\,2} \tag{18}$$

$$U_{ip}=\frac{\partial^0 T_i}{\partial\theta_P}=Q_P\,{}^0T_i\ \ ,\ U_{ir}=\frac{\partial^0 T_i}{\partial\theta_r}=Q_r\,{}^0T_i\ ,\ U_{ir}=U_{ri},U_{iP}=U_{Pi}\ \text{and}$$

$$Q_i(revolute)=\begin{bmatrix}0&-1&0&0\\1&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}\ ,\ Q_i(prismatic)=\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0\end{bmatrix}$$

But in our case all joints in the robot arm used in surgical applications are considered revolute joints where $Q_P = Q_r = Q$.

4.2.1.2. The potential energy

The potential energy of multiple-degree of freedom robot may be given in a concise from as:

$$P = \sum_{j=1}^{n} P_i = \sum_{j=1}^{n} [-m_i g^T ({}^0T_j X \overline{r_j})] \tag{19}$$

$$g^T = \begin{bmatrix} g_x & g_y & g_z & 0 \end{bmatrix}$$

4.2.1.3. Robot equations of motion

The equations of motion of multiple-degree of freedom robot be given in a compact form as:

$$T_i = \sum_{j=1}^{n} D_{ij}\ddot{q}_j + I_{i(act)}\ddot{q}_i + \sum_{j=1}^{n}\sum_{k=1}^{n} D_{ijk}\dot{q}_j q_k + D_i \tag{20}$$

$$D_{ij} = \sum_{p=\max(i,j)}^{n} Trace(U_{pj} J_p U_{pi}^T), \quad D_{ijk} = \sum_{p=\max(i,j,k)}^{n} Trace(U_{pjk} J_p U_{pi}^T),$$

$$D_i = \sum_{p=1}^{n} -m_p g^T U_{pj} \overline{r_p}, \quad U_{Pik} = \frac{\partial U_{Pi}}{\partial \theta_k} = U_{Pi} Q_k \, {}^0T_k$$

$$J_i = \begin{bmatrix} \frac{(-I_{XX} + I_{YY} + I_{ZZ})}{2} & I_{XY} & I_{XZ} & m_i \overline{X_i} \\ I_{XY} & \frac{(I_{XX} - I_{YY} + I_{ZZ})}{2} & I_{YZ} & m_i \overline{Y_i} \\ I_{XZ} & I_{YZ} & \frac{(I_{XX} + I_{YY} - I_{ZZ})}{2} & m_i \overline{Z_i} \\ m_i \overline{X_i} & m_i \overline{Y_i} & m_i \overline{Z_i} & m_i \end{bmatrix}$$

Link #	**1**	**2**	**3**	**4**	**5**	**6**
I_{act} (Nmm)	10000	10000	10000	10000	10000	10000
m (kg)	6.055	6.055	6.055	6.055	6.055	6.055
L (mm)	0	50	15	36	0	39.5
g (m/s2)	9.81	9.81	9.81	9.81	9.81	9.81
R (mm)	75	75	75	75	75	75
I_{xx}	(1/12)*m_i*(3*(R_2)+(L_i/4)2)					
I_{yy}	(1/12)*m_i*(3*(R_2)+(L_i/4)2)					
I_{zz}	m_i * R_2					
I_{xy}	0					
I_{xz}	0					
I_{yz}	0					
X	L_i/2					
Y	0					
Z	0					

Table 5. Full robotic mobility parameters

For the MATLAB simulation, the parameters of the robotic arm are given in table 5. The flowchart for the algorithm employed to calculate the torque history for each actuator based on the derived equations of motion (Equation 20) is shown in the Figure 18. The simulated results for the actuators torques change depending on the method of trajectory planning used to calculate the torque. For example the simulated results for the actuators torques calculated by the Third-Order Polynomial trajectory planning are presented in section 3. It can be seen that the torque history over the selected period of time (5, 10, 20 and 60 s) has considerable fluctuations.

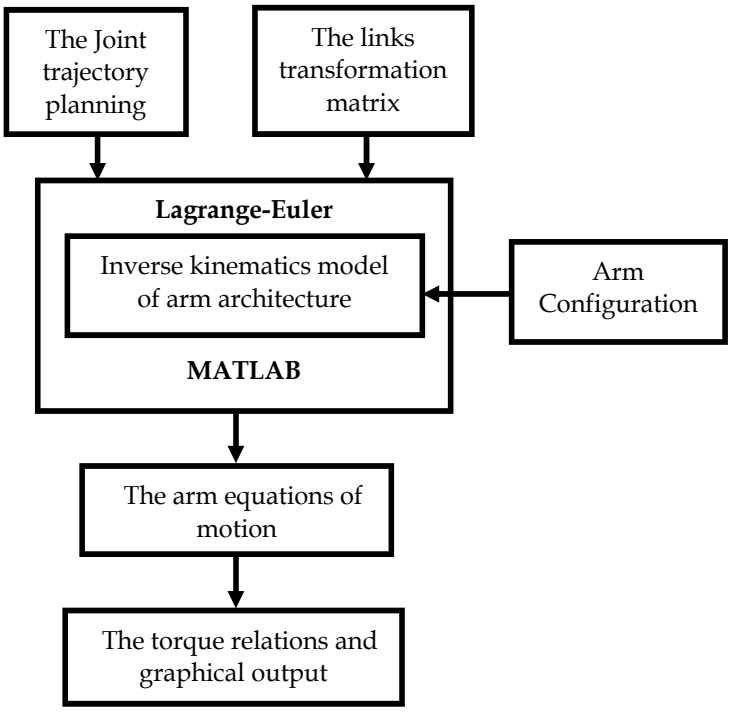


Figure 18. The flowchart of the dynamic model

5. The dynamic response

5.1. Introduction to the dynamic response and dynamic response analysis

The results presented in this chapter are those of chapters IV and V that is the trajectory planning and dynamic modeling respectively as they are very much related to each other. The results are divided into two sections. The first one is for the trajectory planning of the surgical robot which is divided into four parts, each represents a method from four chosen different methods for trajectory planning. As mentioned in chapter IV the joints limits for some of the joints are similar as the joints (1, 4, and 6) and (2, 5). Only the trajectory planning of similar joints will be represented. It should be noted that the results of trajectory planning for all methods were derived using four different times (5, 10, 20 and 60 s) in order to select

the best time which gives the best and smooth orientation, velocity, acceleration (i.e. the inertia of the robot link) and torque. The dynamic modeling results are dependent on the trajectory planning results. This requires the results of the dynamic modeling to be represented after the trajectory planning results i.e. after comparison of the results of the trajectory planning and selecting the best time to be used.

Finally, a comparison of the results is held to choose the best method that gives the smooth set trajectory planning and best performance of the robot under investigation.The simulation results were obtained using MATLAB.

5.2. The trajectory planning analysis

As previously stated that trajectory planning was derived using four methods of the trajectory planning to select the best method that gives the smooth set trajectory planning and best performance of the robot under investigation.

The trajectory planning results is presented in two ways. The first one shows the trajectory planning of the first joint of robot derived using four different times and the second one shows the torque history for the first joint and a comparison between the four times of the first joint of robot in terms of the orientation, velocity, acceleration and the torque history to select the best time that can be used to derive the trajectory planning and torque history for all joints of robot model.

5.2.1. Third order Polynomial trajectory planning

Figures 19 and 20 show comparisons between the orientation, velocity, acceleration, torque and four different time ranges (5, 10, 20 and 60 s) of the first joint of the surgical robot. It is clearly shown in Figure 19 that the orientation behaviour increases gradualy with the time from the initial angle to the final angle and as the time increases the velocity required decreases and also the acceleration i.e. the inertia of the robot link decreases. It is also clearly shown in Figure 20 that the original torque history has considerable fluctuations. It is clear that the highest hub torque is for joint one while actuator torque of joint 6 is the lowest.

From Figures 19 and 20 the optimum final time required to complete the process for the robot can be selected. By inspection of Figure 19 for the trajectory planning of the robot we find that for all times the same behaviours i.e. the velocity and acceleration are inversely proportional with the times and the orientation behaviour increases gradualy with the time from the initial angle to the final angle. The largest time i.e. 60s to complete the results of the surgical robot can be selected. Since this time has the lowest acceleration i.e the lowest inertia. In Figure 20 for the torque history in the time 60s we find that the torque decreases with time and the over shooting decreases gradualy near the steady state time 60s.

It is seen from Table 4 that the joints angle ranges are very large. So, it is assured that the robot accelerations will not exceed the maximum permissible limits for robot's capabilities if the robot is satisfying the trajectory in one segment.

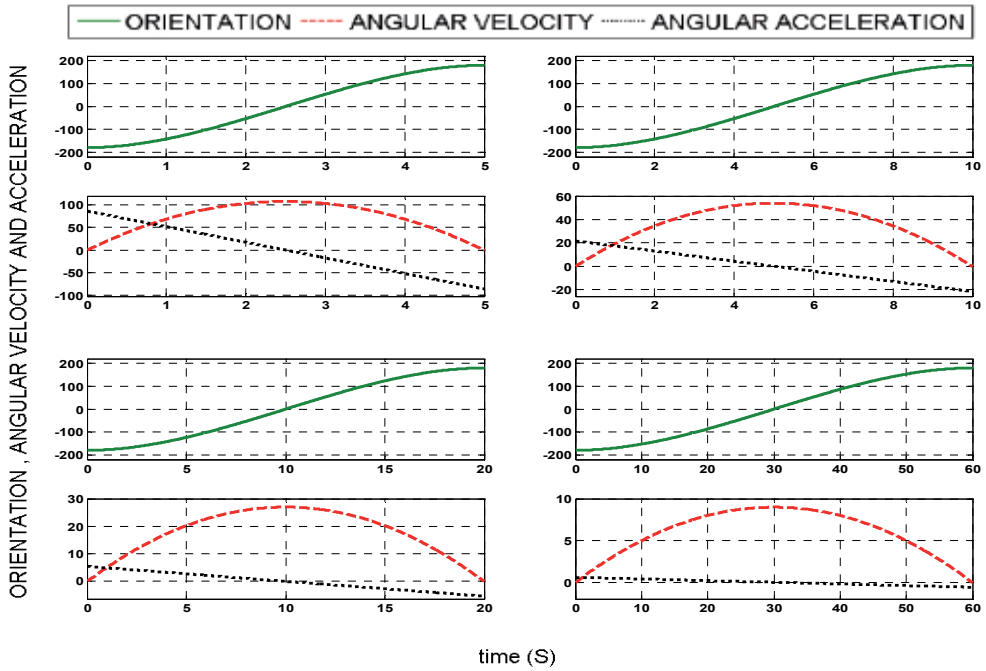


Figure 19. The time comparison of third order trajectory planning of the first joint

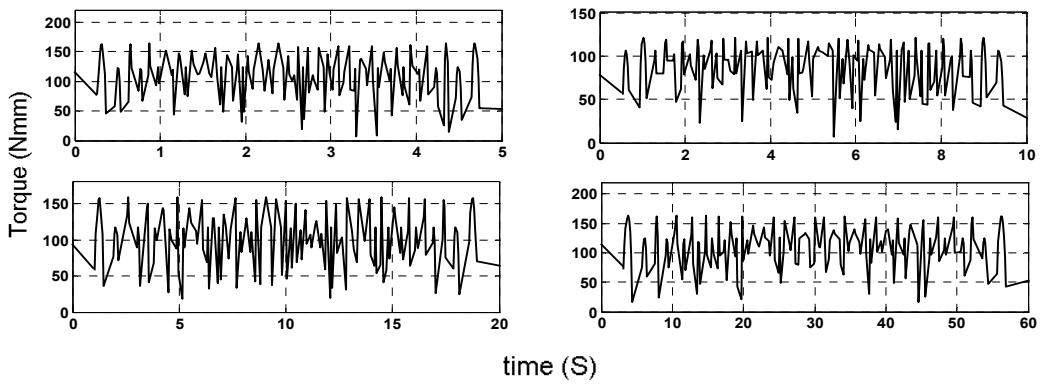
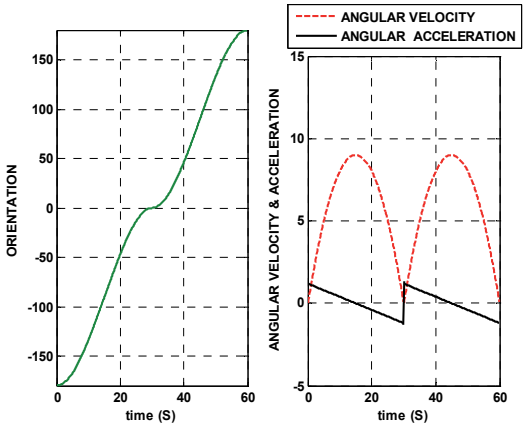


Figure 20. The comparison between the time and torque of the first joint using third order trajectory planning

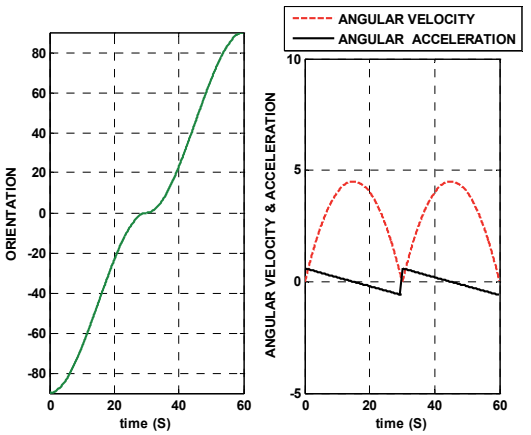
For Third-order Polynomial trajectory planning the maximum accelerations for robot's capabilities is given by:

$$\ddot{\theta}_{\max} = \left| \frac{6(\theta_f - \theta_i)}{(t_f - t_i)} \right| \tag{21}$$

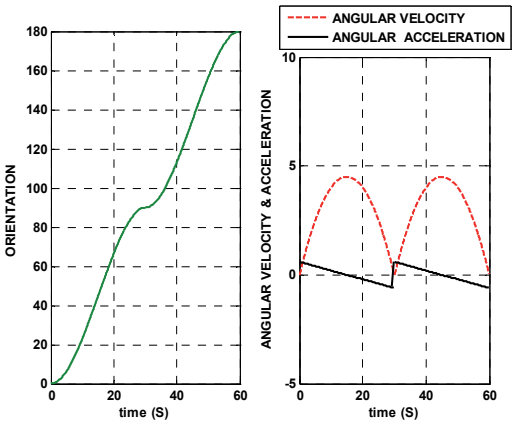
From Figure 19 the robot acceleration needed at the beginning of the motion is 0.62 (degree/s^2) as well as -0.62 (degree/s^2) deceleration at the end of the motion.



(a) The first, fourth and sixth joints



(b) The second and fifth joints



(c) The third joint

Figure 21. The orientation, angular velocity and angular acceleration for six joint micro-robot using Third order Polynomial trajectory planning

But from the equation 6.1 the maximum permissible accelerations for this robot joints is 0.6 (degree/s^2). So to be ensured that robot joint accelerations will not exceed the maximum accelerations for robot's capabilities the robot should satisfy the trajectory planning in two or more segments.

Figures 21a, b and c show the modified trajectories for the six joints, after dividing the trajectory in to two segments.

The joints angle ranges of the initial and final angles are large compared to the values of the velocity and accelerations. So it is more suitable to represent the angle-Time relation and velocity and acceleration – Time relations in two separate figures.

5.2.2. Fifth order Polynomial trajectory planning

Figures 22 and 23 show comparisons between the orientation, velocity, acceleration, and torque for four different time ranges (5, 10, 20 and 60 s) of the first joint of the robot. As is clearly shown in Figure 22 the orientation behaviour increases gradually with time from the initial angle to the final angle for the times (5, 10 and 20 s) only. But for the time 60s the orientation increases gradually from the initial angle to 40 degree then decreases to -40 degree and then increases reaching the final angle. And if the time increases the velocity required decreases. This is similar as to what is shown in Figure 23 which presents the original torque history that has considerable fluctuations. It is clear that the highest hub torque is for joint one while actuator torque of joint 6 is the lowest.

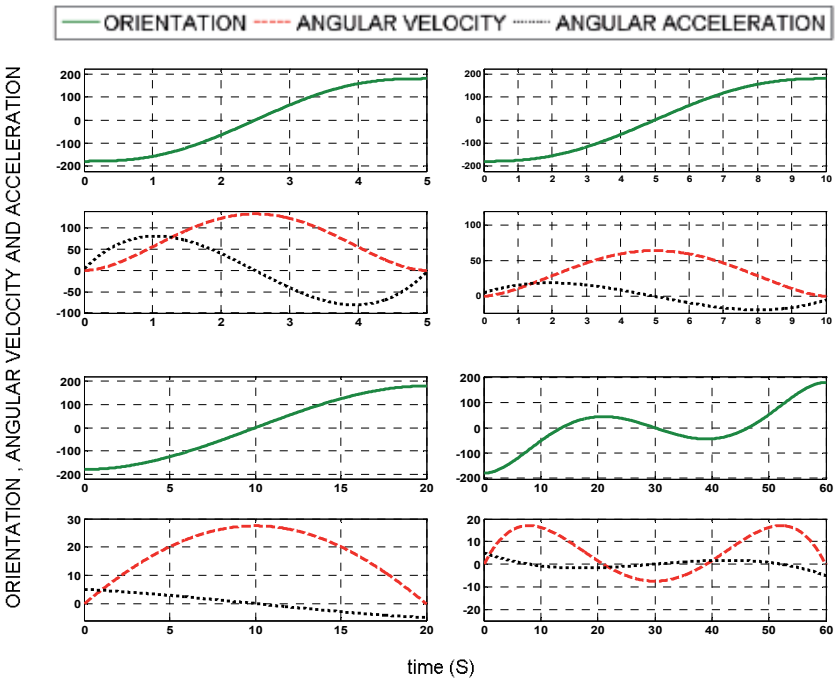


Figure 22. The time comparison of fifth order trajectory planning of the first joint

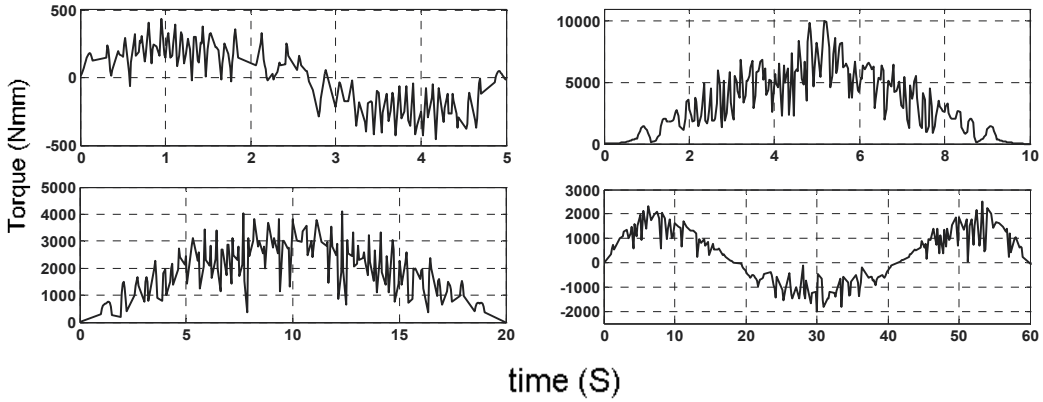


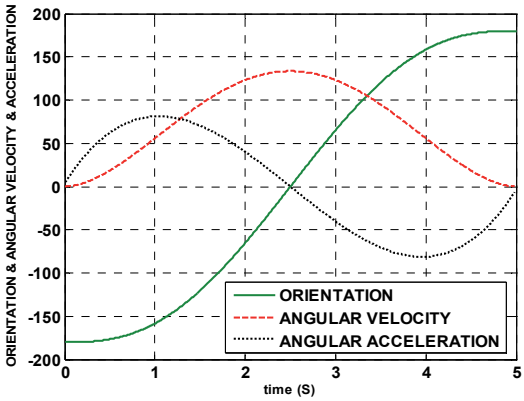
Figure 23. The comparison between the time and torque of the first joint using fifth order trajectory planning

From Figures 22 and 23 the optimum final time required to complete the process for the robot can be selected. It is seen from figures that the trajectory planning of the robot for the times (5, 10 and 20 s) has the same properties i.e. the velocity is inversely proportional with time. The orientation behaviour increased gradualy with the time from the initial angle to the final angle. Where the time 60s is omitted from selected. For figure 23 for the torque history in the time (5, 10, 20 and 60 s) it is found for $t_f = 5$ s, the dominant part in the torque history is the inertia matrix. Increasing the final time to 10, 20 and 60s shifts the dominant term from inertia matrix to Centrifugal and Coriolis matrices. This is due to the vanishing of the acceleration at most of the joint trajectory. Another consequence of increasing the final time is the dramatic increase in the peak value of the joint torque which requires big actuator size for the same task (i.e. the same joint parameters). The time 5s to complete the results of the surgical robot can be selected. Since this time has orientation behaviour was increased gradualy with the time from the initial angle to the final angle and the torque history curves were affected by the inertia of the link of robot.

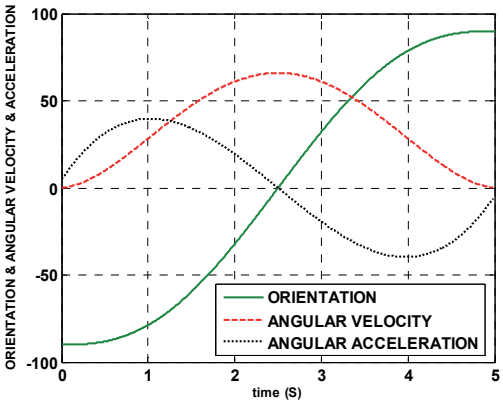
Figures 24a, b and c show the trajectories for the six joints of the robot using Fifth order Polynomial trajectory planning.

5.2.3. Linear segments with parabolic blends

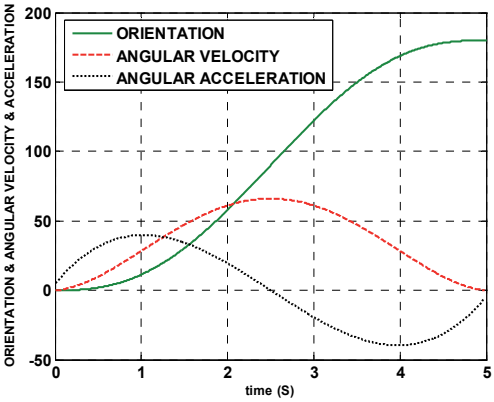
Figures 25 and 26 show comparisons between the orientation, velocity, acceleration, torque and four different time ranges (5, 10, 20 and 60 s) of the first joint of the surgical robot. It is clearly shown in the Figure 25 that the orientation behaviour increases gradually with the time from the initial angle to the final angle and as the time increases the velocity required decreases and also the acceleration i.e. the inertia of the robot link decreases**.** It is also clearly shown in the Figure 26 that the original torque history has considerable fluctuations. It is clear that the highest hub torque is for joint one while actuator torque of joint 6 is the lowest.



(a) The first, fourth and sixth joints



(b) The second and fifth joints



(c) The the third joint

Figure 24. The orientation, angular velocity and angular acceleration for six joint micro-robot using Fifth order Polynomial trajectory planning

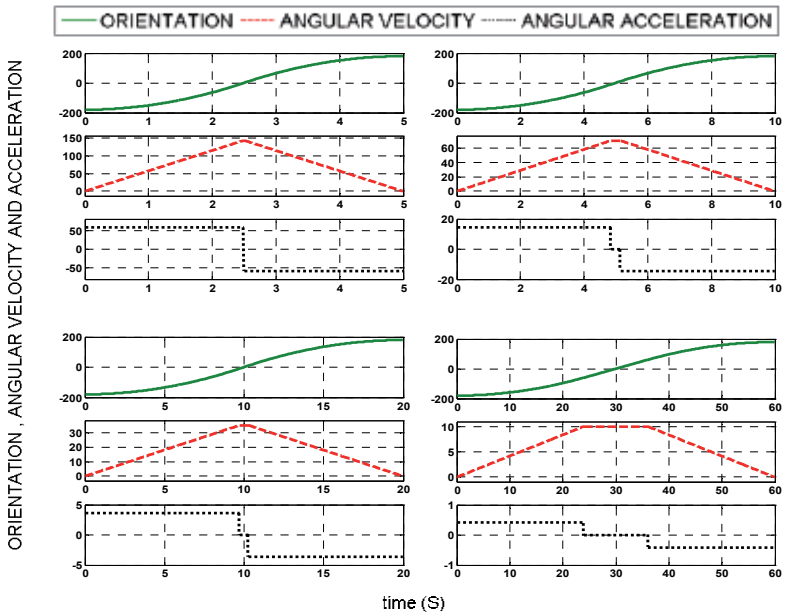


Figure 25. The time comparison of liner segments with parabolic blends trajectory of the first joint

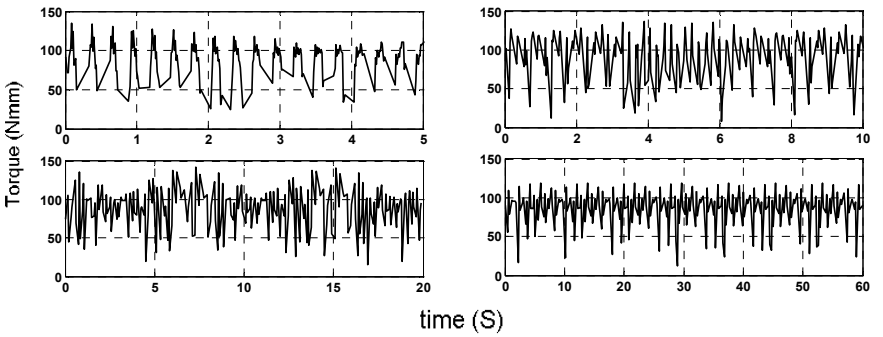
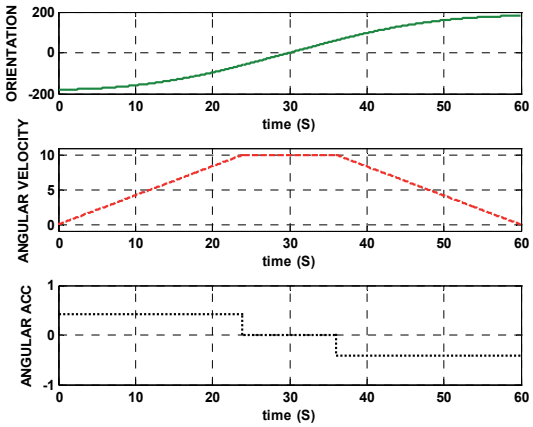


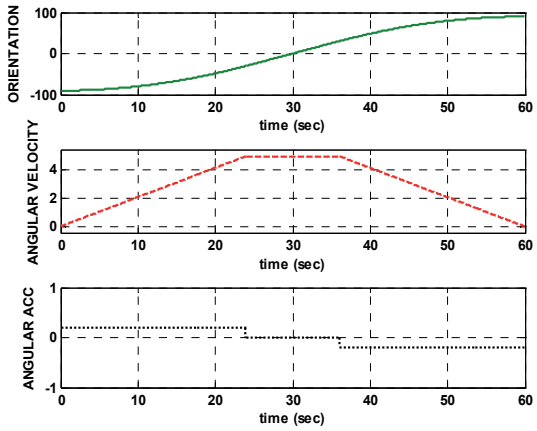
Figure 26. The comparison between the time and torque of the first joint using liner segments with parabolic blends trajectory planning

From Figures 25 and 26 the optimum final time required to complete the process for the robot can be selected. By inspection of Figure 25 the trajectory planning has three segments they are First parabolic blends, Straight line and Second parabolic blends. The straight line segment was very important segment because the velocity in this segment is constant and the acceleration was zero i.e. no inertia of the link of the robot in this segment of time. It is therefore better to have a larger period for this segment. The largest time i.e. 60s to complete the results of the robot can be selected. Since this time has the largest period of straight line segment. By inspection of Figure 26 for the torque history in the time 60s we find that the torque decreases with the time and the shooting decreases gradualy near the steady state time 60s.

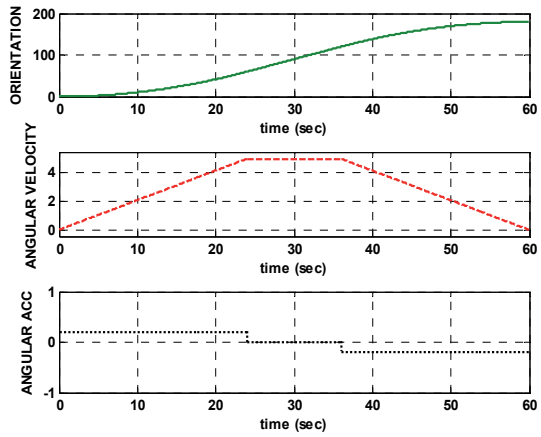
Figures 27a, b and c show the trajectories for the six joints of the robot using linear segments with parabolic blends.



(a) The first, fourth and sixth joints



(b) The second and fifth joints



(c) The third joint

Figure 27. The orientation, angular velocity and angular acceleration for six joint micro-robot using parabolic blends trajectory planning

5.2.4. Soft motion trajectory planning

Figures 28 and 29 show comparisons between the orientation, velocity, acceleration, jerk, torque and four different time ranges (5, 10, 20 and 60 s) of the first joint of the surgical robot. It is clearly shown in the Figure 28 that the orientation behaviour increases gradually with the time from the initial angle to the final angle and as the time increases the velocity required decreases and also the acceleration i.e. the inertia of the robot link decreases. It is also clearly shown in Figure 29 that the original torque history has considerable fluctuations. It is clear that the highest hub torque is for joint one while actuator torque of joint 6 is the lowest.

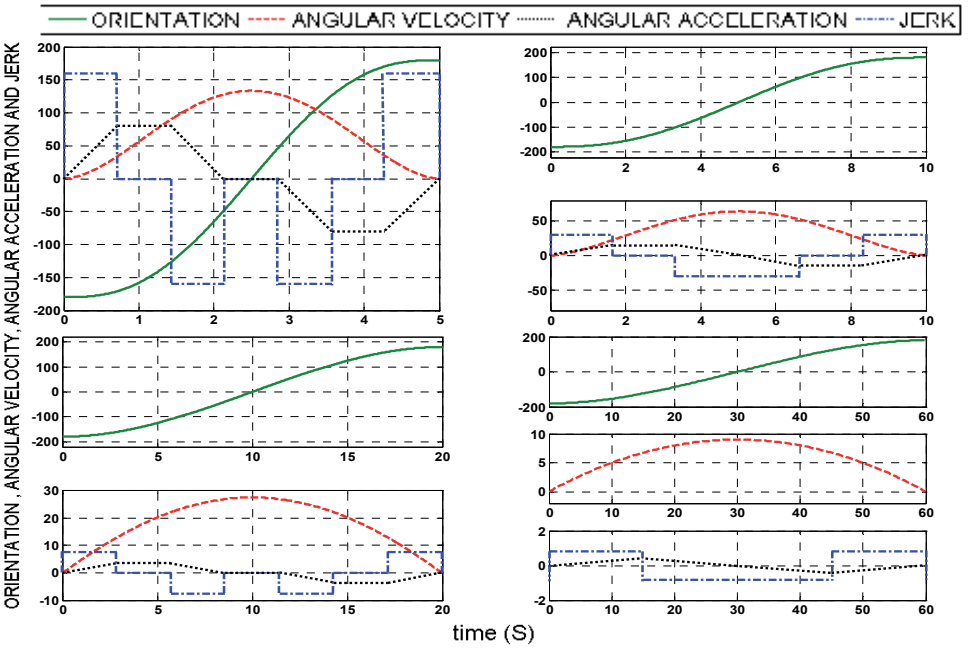


Figure 28. The time comparison of Soft motion trajectory planning of the first joint

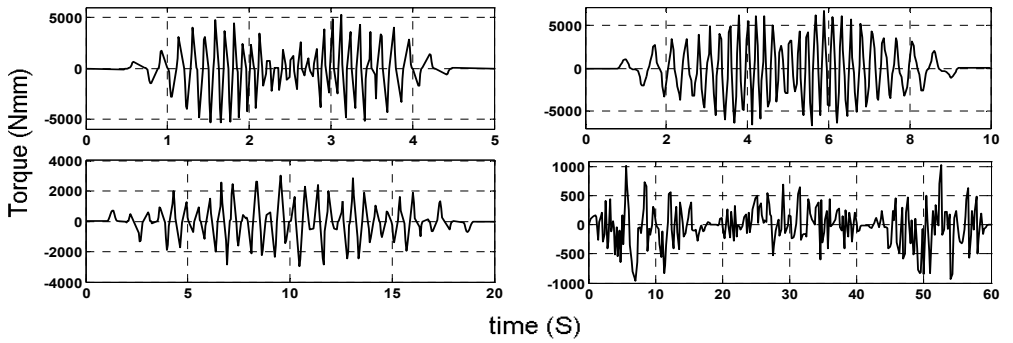
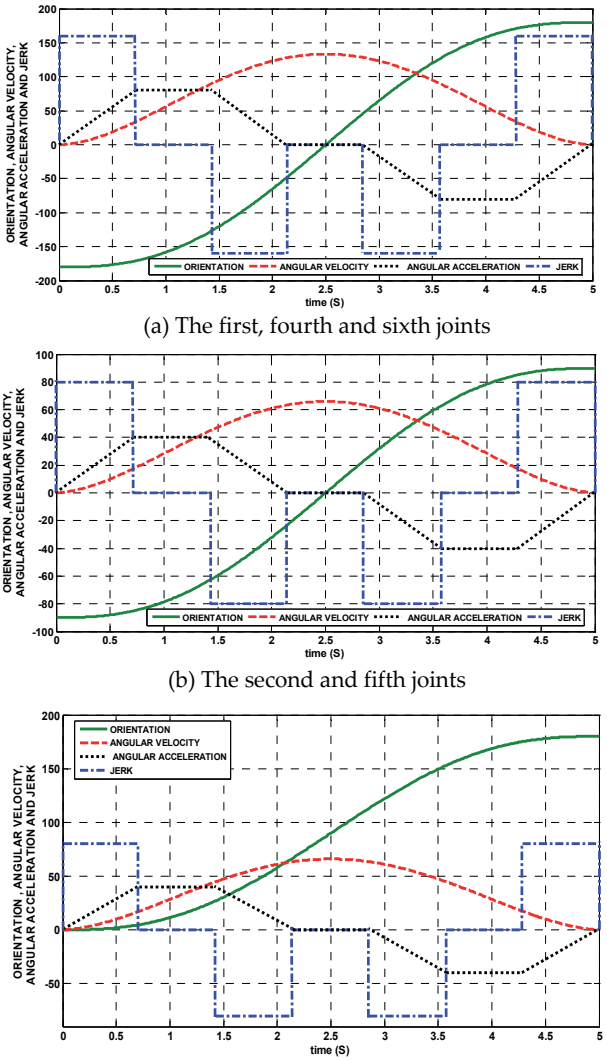


Figure 29. The comparison between the time and torque of the first joint using soft motion trajectory planning



(a) The first, fourth and sixth joints

(b) The second and fifth joints

(c) The third joint

Figure 30. The orientation, angular velocity and angular acceleration for six joint micro-robot using soft motion trajectory planning

From Figures 28 and 29 we can select the optimum final time required to complete the process for the robot. By inspection of Figure 28 we find the trajectory planning three segments they are maximum jerk, maximum acceleration and maximum velocity. The maximum acceleration and maximum velocity segments were very important segments because the velocity in the maximum velocity segment is constant and the acceleration was zero i.e. no inertia of the link of the robot in this segment of time and the acceleration in the maximum acceleration segment is constant and the jerk of the link of the robot was zero. It is therefore better to have a larger period for this segment. It is seen from figures that the segments were released from time 60s but the the maximum velocity segment only was released from time 10s. Where the times (10

and 60 s) is omitted from selected i.e. the times (10 and 60 s) have properties were not satisfactory for trajectory of robot. And by inspection of Figure 29 for the torque history in the times (5 and 20 s) we find for $t_f = 5$ s, the dominant part in the torque history is the inertia matrix. Increasing the final time to 20s, shift the dominant term from inertia matrix to Centrifugal and Coriolis matrices since the effect of angular velocity will be obviously high. This is due to the vanishing of the acceleration at most of the joint trajectory. Another consequence is of increasing the final time is the dramatic change in the peak value of the joint torque which requires big actuator size for the same task. So we can select the time 5s to complete the results of the robot because the torque history curve was affected by the inertia and it has the important segment i.e. maximum acceleration and maximum velocity segments.

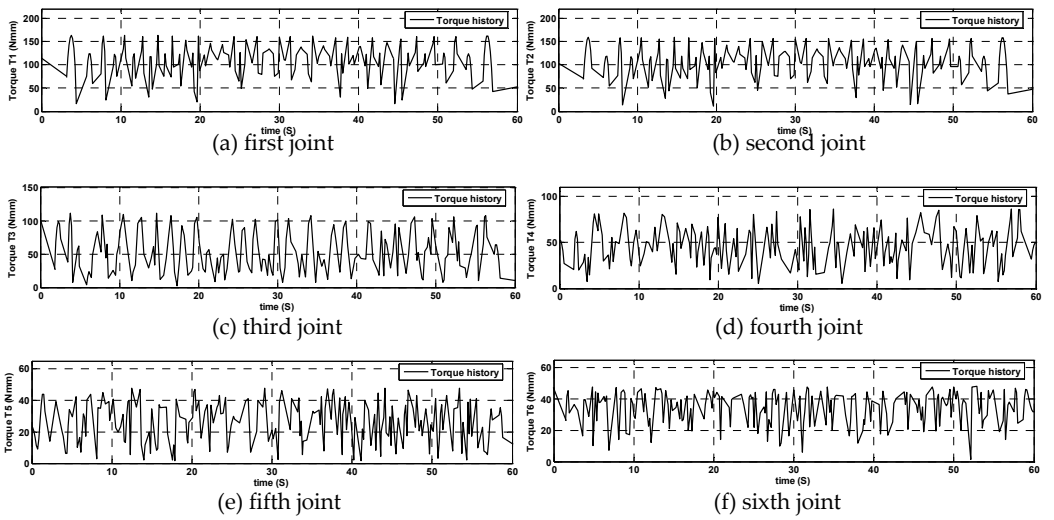
Figures 30a, b and c show the trajectories for the six joints of the robot using soft motion trajectory planning.

5.3. The dynamic response analysis

As previously stated that the dynamic analysis of the surgical robot was derived using Lagrange-Euler technique and the results of the dynamic analysis were depended on the method of trajectory planning. In the trajectory planning results were derived using four different methods. The dynamic analysis results of surgical robot are divided into four parts each part in for each special method of the four different methods which have been derived from the trajectory planning.

Figures 31 to 34 show the original torque history which clearly shown considerable fluctuations. It is clear that the highest hub torque is experienced at joint one while actuator torque of joint 6 is the lowest.

5.3.1. The dynamic analysis results using Third order Polynomial trajectory planning



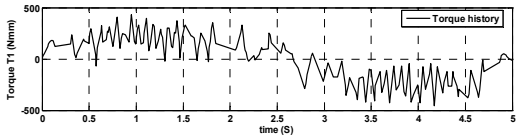
(a) first joint (b) second joint

(c) third joint (d) fourth joint

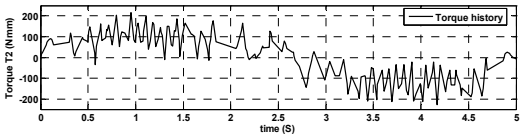
(e) fifth joint (f) sixth joint

Figure 31. Torque history for Third order Polynomial trajectory planning

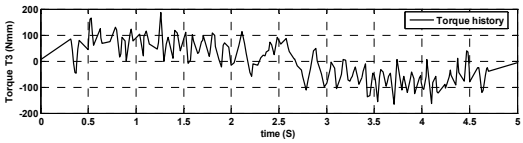
5.3.2. The dynamic analysis results using Fifth order Polynomial trajectory planning



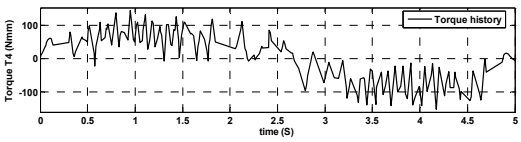
(a) first joint



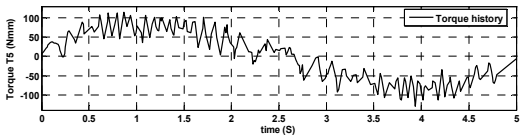
(b) second joint



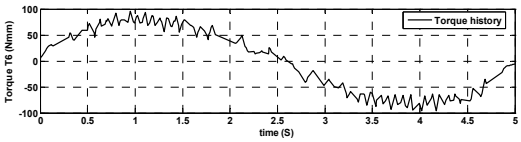
(c) third joint



(d) fourth joint



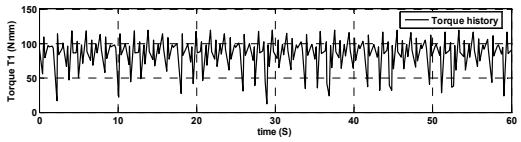
(e) fifth joint



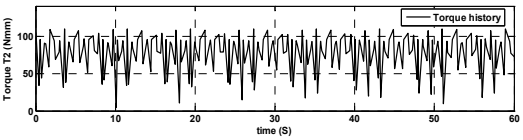
(f) sixth joint

Figure 32. Torque history for Fifth order Polynomial trajectory planning

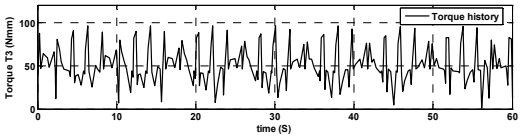
5.3.3. The dynamic analysis results using Linear segments with parabolic blends trajectory planning



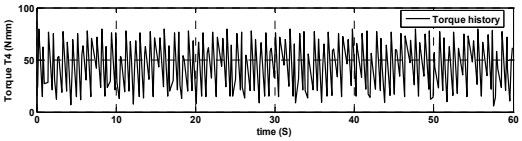
(a) first joint



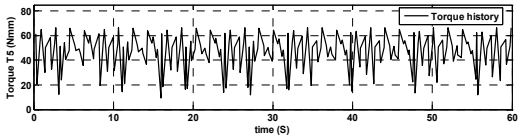
(b) second joint



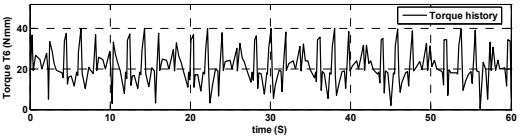
(c) third joint



(d) fourth joint



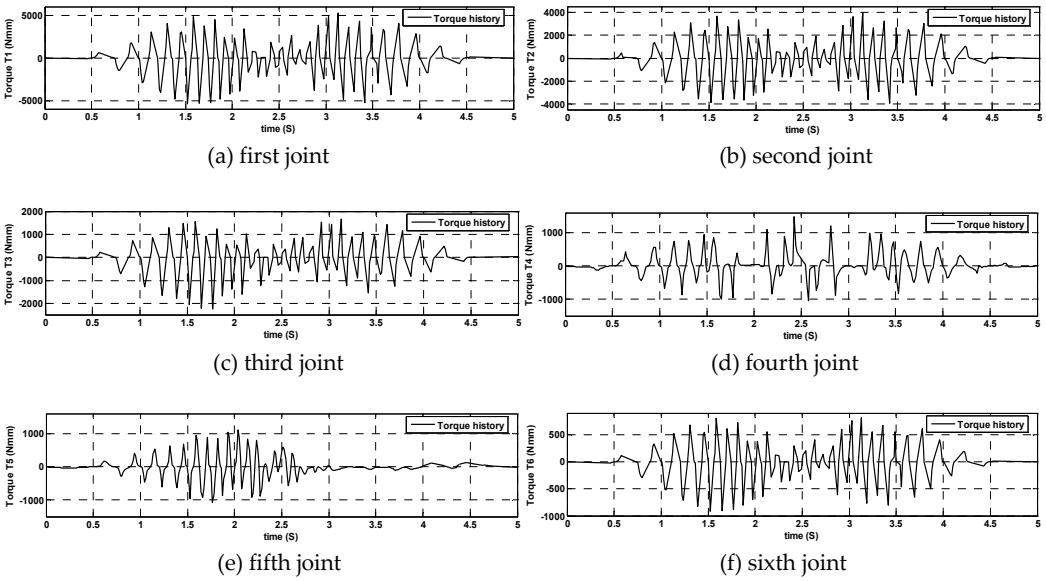
(e) fifth joint



(f) sixth joint

Figure 33. Torque history for Linear segments with parabolic blends trajectory planning

5.3.4. The dynamic analysis results using Soft motion trajectory planning



(a) first joint (b) second joint

(c) third joint (d) fourth joint

(e) fifth joint (f) sixth joint

Figure 34. Torque history for Soft motion trajectory planning

6. Conclusions

A kinematic and dynamic analysis for a six-degree-of-freedom surgical robot were presented in this work. The kinematic model is based on Denavit-Hartenberg representation and the workspace of the end-effector is defined by solving the inverse kinematics problem. Four different methods were used to derive the trajectory planning for the six joints and were designed and employed to calculate the torque history for the six actuators. The dynamic equations of motion in symbolic form were derived using the Lagrange-Euler technique and the torque history was obtained using MATLAB for each joint. The proposed algorithm is flexible and can be extended to any robot configuration provided that the Denavit-Hartenberg presentation was available and the physical limits of joints are defined. The original torque history has considerable fluctuations. It was shown that the highest hub torque was of joint 1 while actuator torque of joint 6 was the lowest. It should be also noted that changing the final time for the joint trajectory changes the torque history considerably. The final time required to complete the process was selected depending on the method used to derive the trajectory planning as previously stated.

It was clearly shown in this work that the best method of trajectory planning that gives the smooth set trajectory planning and best performance of the robot under investigation was the soft motion trajectory planning because the most important reason for this selection was the torque history that has the lowest number ever of shooting and the shooting was distributed regularly over the period of time unlike the other methods which have a long number of shootings and were distributed randomly. Also the reason for selecting this

method was the disappearance of the shooting quite before the final time of trajectory i.e. the steady state time. T

Author details

Wael A. Al-Tabey
Department of Mechanical Engineering Faculty of Engineering,
Alexandria University, Alexandria, Egypt

Appendix

Appendix (A): Notation

C_j	cos θ_j
S_j	sin θ_j
j	the total number of joints
i	the total number of coordinates
n	the total number of links
k and p	coefficient (1, 2, 3… n)
x, y, z	local joint coordinates
X,Y,Z	global joint coordinates
Y_i	the Kinematics equations of the end effectors
X_j	the joint variables (θ_1, θ_2, θ_3, θ_4, θ_5, θ_6)
D	the end effector differential translation matrix
D_θ	the joint differential motion matrix
J	the robot Jacobian matrix (i x j)
θ_i	the initial values of each joint angle
θ_f	the final values of each joint angle
t_f	the time duration
$\theta(t)$, $\dot{\theta}(t)$, $\ddot{\theta}(t)$, $J(t)$	the orientation, angular velocity, angular acceleration and angular jerk respectively
$\ddot{\theta}_0, \dot{\theta}_0, \theta_0$	the initial conditions
T_{jpa}	Jerk positive initial time
T_{aca}	Acceleration constant initial time
T_{jna}	Jerk negative initial time
T_{vc}	Velocity constant time
T_{jnb}	Jerk negative final time
T_{acb}	Acceleration constant final time
T_{jpb}	Jerk positive final time
q	the vector of generalized joint coordinates
$\dot{q}$	the vector of joint velocities
$\ddot{q}$	the vector of joint accelerations
M	the inertia matrix

C	the Coriolis and centrifugal matrix
G	the gravity matrix
Q	the vector of generalized force associated
g^T	he gravity matrix (1x4)
g_x, g_y and g_z	the gravity components in x, y and z direction respectively
$\overline{r_j}$	the location of the center of mass of link relative to the frame representing the link
T_j	the torque on joint (j)
J_i	the inertia matrix
I_{XX}, I_{YY} and I_{ZZ}	the principal moments of inertia for the link
I_{XY}, I_{XZ} and I_{YZ}	the parallel moments of inertia for the link
m_i	the mass of the link
$\overline{X_i}$, $\overline{Y_i}$ and $\overline{Z_i}$	the distance between the X, Y and Z axis to the center of the link mass respectively

Appendix (B): MATLAB Code

```
%*****************************************************************************************
%                                   Micro-Robot Management
%                              Complete MATLAB Code for kinematic
%*****************************************************************************************
syms th1 th2 th3 th4 th5 th6 L1 L2 L3 L4
%************************THE INPUTS OF ROBOT ****************************************
L1=input('please enter The length of Link NO(1) (mm)=');
L2=input('please enter The length of Link NO(2) (mm)=');
L3=input('please enter The length of Link NO(3) (mm)=');
L4=input('please enter The length of Link NO(4) (mm)=');
th1min=input('please enter minth1 (degree)=');
th1max=input('please enter maxth1 (degree)=');
th2min=input('please enter minth2 (degree)=');
th2max=input('please enter maxth2 (degree)=');
th3min=input('please enter minth3 (degree)=');
th3max=input('please enter maxth3 (degree)=');
th4min=input('please enter minth4 (degree)=');
th4max=input('please enter maxth4 (degree)=');
th5min=input('please enter minth5 (degree)=');
th5max=input('please enter maxth5 (degree)=');
th6min=input('please enter minth6 (degree)=');
th6max=input('please enter maxth6 (degree)=');
%*****************************************************************************************
th1=th1min:2:th1max;th2=th2min:th2max;th3=th3min:th3max;
th4=th4min:2:th4max;th5=th5min:th5max;th6=th6min:2:th6max;
%*****************************************************************************************
```

```
c1=cos(th1);c2=cos(th2);c3=cos(th3);c4=cos(th4);c5=cos(th5);c6=cos(th6);
s1=sin(th1);s2=sin(th2);s3=sin(th3);s4=sin(th4);s5=sin(th5);s6=sin(th6);
%***********************D.H matrix*****************************************************************
A1=[c1,-s1,0,L1*c1;s1,c1,0,L1*s1;0,0,1,0;0,0,0,1];A2=[c2,0,s2,L2*c2;s2,0,-c2,L2*s2;0,1,0,0;0,0,0,1];
A3=[c3,0,-s3,0;s3,0,c3,0;0,-1,0,0;0,0,0,1];A4=[c4,0,-s4,0;s4,0,c4,0;0,-1,0,L3;0,0,0,1];
A5=[c5,0,-s5,0;s5,0,c5,0;0,-1,0,0;0,0,0,1];A6=[c6,-s6,0,0;s6,c6,0,0;0,0,1,L4;0,0,0,1];
A01=A1;A02=A1*A2;A03=A1*A2*A3;A04=A1*A2*A3*A4;A05=A1*A2*A3*A4*A5;
A06=A1*A2*A3*A4*A5*A6;
%************The Kinematics equations of the end effectors***********************************
px= A06(4,1),py= A06(4,2),pz=A06(4,3)
subplot(1,3,1);plot(xa,ya);
xlabel('Px');ylabel('Py');grid on
subplot(1,3,2);plot(xa,z);
xlabel('Px');ylabel('Pz');grid on
subplot(1,3,3);plot(ya,z);
xlabel('Py');ylabel('Pz');grid on
%*****************************************THE end*********************************************
%********************CREATED BY DR/WAEL A. AL-TABEY*********************************
```

7. References

[1] Frumento, C. (2005). Development of micro tools for surgical applications. A Ph. D. Co-Tutorship Thesis, University' Degli Studi De Genova/ Uiversite Piere et Marie Currie Paris.

[2] Venita, C. Sanjeev, D. and Craig, T. (2006). Surgical robotics and image guided therapy in pediatric surgery: Emerging and converging minimal access technologies, Seminars in Pediatric Surgery, Volume 14, pp. 267– 275.

[3] Tsai, T. and Hsu, Y. (2004). Development of a parallel surgical robot with automatic bone drilling carriage for stereotactic neurosurgery, IEEE SMC, International Conference on Systems, Man and Cybernetics, Hague, Netherlands, October 10-13, 2004.

[4] Miller, A. and Christensen, H. (2003). Implementation of Multi-rigid-body Dynamics within a Robotic Grasping Simulator, Proceedings of International Conference, Volume 2, pp. 2262, 14-19 Sept. 2003.

[5] Featherstone, R. and Orin, D. (2000). Robot Dynamics: Equations and Algorithms, Proc. IEEE Int. Conf. on Robotics and Automation, pp. 826-834.

[6] Wang, S. (2008). Conceptual design and dimensional synthesis of Micro-Hand, Mechanism and Machine Theory, Volume 43, Issue 9, pp. 1186-1197.

[7] Alıcı, G. and Shirinzadeh, B. (2004). Loci of singular configurations of 3-DOF spherical parallel manipulator, Robotics and Autonomous Systems, Volume 48, pp. 77–91.

[8] Ben-Horin, R. (1998). kinematics, dynamics and construction of a planarly actuated parallel robot, Robotics and computer –integrated Manufacturing, Volume 14, pp. 163–172.

[9] Bonnifait, P. and Garcia, G. (1999). 6-DOF dynamic localization of an outdoor mobile robot, Control Engineering Practice, Volume 7, pp. 383–390.

[10] Abdellatif, H. and Heimann, B. (2008). Computational efficient inverse dynamics of 6-DOF fully parallel manipulators by using the Lagrangian formalism, Mechanism and Machine Theory, Volume 7, pp. 383–390.
[11] Zhu, Z. (2005). Kinematic and dynamic modelling for real-time control of Tau parallel robot, Mechanism and Machine Theory, Volume 40, pp. 1051–1067.
[12] Gouliaev, V. and Zavrazhina, T. (2001). Dynamics of a flexible multi-link cosmic robot-manipulator, Journal of Sound and vibration, Volume 243(4), pp. 641-65.
[13] Carrera, E. and Serna, M. (1996). Inverse dynamics of flexible robots, Mathematics and Computers in Simulation, Volume 41, pp. 485–508.
[14] Martins, F. (2008). An adaptive dynamic controller for autonomous mobile robot trajectory tracking, Control Engineering Practice, Volume 16, pp. 1354– 1363.
[15]Valero, F. (2006). Trajectory planning in workspaces with obstacles taking into account the dynamic robot behaviour, Mechanism and Machine Theory, Volume 41, pp. 525– 536.
[16] Geng, T. (2005). Dynamics and trajectory planning of a planar flipping robot, Mechanics Research Communications, Volume 32, pp. 636–644.
[17]Alessandro, G and Vanni, Z. (2008). A technique for time-jerk optimal planning of robot trajectories, Robotics and Computer-Integrated Manufacturing, Volume 24, pp. 415– 426.
[18] Pires, S. (2007). Manipulator trajectory planning using a MOEA, Applied Soft Computing, Volume 7, pp. 659– 667.
[19] Chettibi, T. (2004). Minimum cost trajectory planning for industrial robots, European Journal of Mechanics A/Solids, Volume 23, pp. 703-715.
[20] Alessandro, G. and Vanni, Z. (2007). A new method for smooth trajectory planning of robot manipulators, Mechanism and Machine Theory, Volume 42, pp. 455– 471, 2007.
[21] Corke, P. (1996). A Robotics Toolbox for MATLAB, IEEE Robotics and Automation Magazine, 3(1):pp.24–32.
[22] Ata, A. (2007). Optimal Trajectory Planning for Manipulators: A Review, Journal of Engineering Science and Technology, Volume 2, No. 1, pp. 32-54.
[23] Niku, S. (2001). Introduction to Robotics Analysis, Systems, Applications, London, International (UK) Limited, pp. 147-159.
[24] Herrera, I. and Sidobre, D. (2006). Soft Motion and Visual Control of Service Robot, The Fifth International Symposium in Robotics and Automation, August 2006.
[25] Niku, S. (2001). Introduction to Robotics Analysis, Systems, Applications, London, International (UK) Limited, pp. 128-138.
[26] Al-Tabey, W. (2010). Effect of Trajectory Planning on Dynamic Response of Micro-Robot for Surgical Application, 9th WSEAS International Conference on System Science and Simulation in Engineering (ICOSSSE '10), ISBN: 978-960-474-230-1, Iwate Prefectural University, Japan, October 4-6, 2010.
[27] Tawfik, K. Ata, A. Al-Tabey, W. (2009). Kinematics and dynamics analysis of micro-robot for surgical applications, World Journal of Modelling and Simulation, Vol. 5, No. 1, pp. 22-29, ISSN 1 746-7233.

Chapter 13

Remote Process Control and Monitoring Using Matlab

Vedran Vajnberger, Semir Silajdžić and Nedim Osmić

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46464

1. Introduction

Remote control is one of the best solutions for managing inaccessible systems. Many researches have been made in the field of the remote control [1-4]. These researches have improved certain aspects of industry, medicine, military, etc. The benefits of remote control are numerous such as: operating in hazard environment, telemedicine, missile guidance, etc. The most important characteristic of remote control is operating in real-time as shown in papers form references [5-7].

Many applications in the field of medicine and industry use different kind of motor-based systems especially stepper motors because of their wide-range of sufficient characteristics like the fact that they can be used as constant power devices with accurate positioning and fast response [8-12].

In today's society, robots are used in various areas especially in those where high precision is required. Some of the examples where robotic arms found their appliance are: in vehicle construction where efficiency and reliability are required, in chemical industry where environment is not suitable for human, in medicine where robotic arm precision is used in operations, etc [13]. Robots have improved life standards and we are upgrading their performances in order to make our lives easier and more comfortable.

This chapter describes implementation of the proposed remote control of the stepper motor and robotic arm with five DOF via web server and VNC server [14, 15]. VNC server was used to receive visual feedback from robotic arm. The quality of image was important and it couldn't be sent via MATLAB server because real time characteristic would be lost. The decision to use microcontroller was based on its characteristics. Developing such system is cheaper than developing on other platforms such as PLC or FPGA.

The realized system exhibits the following:

1. implementation based on PIC16F877a microcontroller,
2. adjustment
   a. of the velocity, number of steps and rotation direction of the stepper motor,
   b. of the rotation direction of the DC motors inside each joint of the robotic arm
3. precise control over RS-232 serial communication,
4. extension to remote control the whole system,
5. feedback
   a. data acquisition and transmission to confirm the proper operation of the stepper motor
   b. visual feedback and contact sensors to confirm the proper operation of the robotic arm
6. realized GUI.

Both systems consist of all previously mentioned characteristics, where character 'a)' refers to stepper motor and character 'b)' to robotic arm.

This chapter is structured as follows: In section 2. whole data analysis of the realization of the system in accordance with previously specified requirements is described. Section 3. provides hardware description that was realized manually. Section 4. explains hardware programming.

2. Problem analysis

The problem of realizied systems that remotely control stepper motor and robotic arm can be divided into several interconnected units, as it will be described in the following text and shown on Figure 1. and Figure 2. To understand the problem and its solution, a brief description of each part will be provided in the rest of this section.

a. Control via microcontroller

The Microchip microcontroller PIC16F877a © generates impulses on four pins that are used to stimulate the movement of the stepper motor gear as shown in Figure 2. Stepper motor consists of multiple "toothed" electromagnets ranged around a coil of iron. Those electromagnets need to be stimulated by some kind of external control unit, in our case a microcontroller. The microcontroller provides electrical impulses which stimulate the gear's teeth to magnetically be attracted to electromagnet's teeth. When the gear's teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet. When the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one, and from there the process is repeated. Each of those slight rotations is called a "step," with an integer number of steps making a full rotation. In that way, the motor can be turned by a precise angle [16].

The impulses (U1÷U4) and response on Figure 2. show the movement of the stepper in one direction by four steps. Each step corresponds with one triggering impulse.

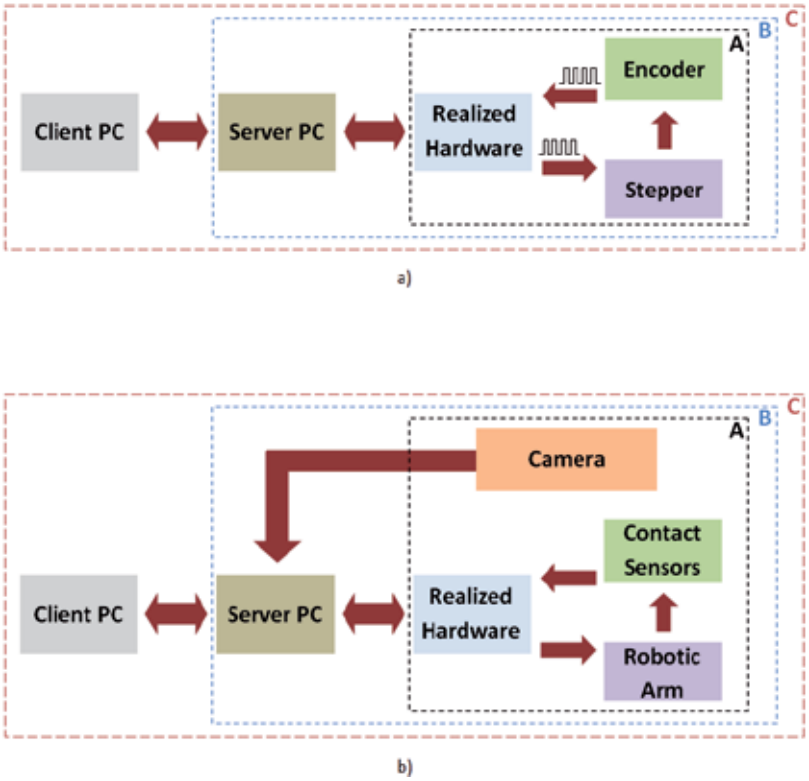


Figure 1. a) Interconnected units of stepper motor system; b) Interconnected units of robotic arm system

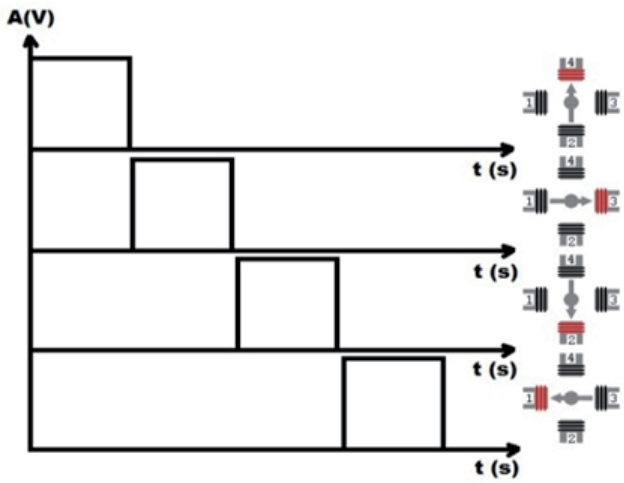


Figure 2. Triggering impulses and gear response

If triggering impulses are reversed (U4÷U1) movement of the motor is in the opposite direction. The length of the triggering impulses depends on desired velocity. The width

between those impulses is infinitesimal. The shorter the length of the triggering impulses is, the higher velocity becomes; i.e. to reach 60 [rpm], the length of each triggering impulse needs to be 5 [ms], to reach 120 [rpm] length should be 2.5 [ms]. As earlier mentioned, besides velocity control, there was the urge for exact positioning. This task was realized by sending a certain amount of triggering impulses in the correct order.

Even though stepper motors are used in open-looped systems because of their characteristics, in this case an additional encoder was installed on the system. Its purpose is to check if the stepper responses accurately to the given commands i.e. combination of impulses.

Similar to the robotic arm (Figure 1.b), Microchip microcontroller PIC16F877A© generates impulses on ten pins (PORT D and two pins from PORT C) which are triggering relays. Signals from relays are used to stimulate the movement of the DC motors implemented inside joins of the robotic arm.

For every degree of freedom (DOF) two pins of microcontroller and two relays are assigned (pins RD0 and RD1 for base, RD2 and RD3 for shoulder, RD4 and RD5 for elbow, RD6 and RD7 for wrist and RC0 and RC1 for fist).

Depending on the state of two pins, there are four situations:

- if both pins are low, two relays controlled by them are open and motor of the appropriate DOF is not running,
- if one pin is high and another is low, current flows in one direction and motor is running in appropriate direction,
- for opposite state of pins, motor is running in opposite direction,
- 'forbidden combination' is when both pins are high, because then both relays are active and source is short circuited.

Figure 3. explains the movement of robotic arm.

b. Control via RS-232

In this chapter, the advantages of a microcontroller were used to establish a communication with the server PC. The used communication is the serial communication RS-232. Serial communication is the most common low-level protocol for communicating between two or more devices [17, 18]. Texas Instruments' MAX232 is used to make adjustment between microcontroller's TTL logic level (5V÷0V) and logic level for RS232 standard (-12V÷12V). An example of the conversion is shown in Figure 4.

The communication was established through MATLAB using three m-files. One m-file was written to create serial port object and to configure its properties. The communication between the server PC and the microcontroller is realized by using second m-function called "send". It has five input arguments: *send(s1, message1, message2, message3, message4)*. The first input argument is the name of a created serial port object. Other arguments are one-byte values that are sent to the microcontroller. Those values represent desired mode of operation (velocity or positional mode), direction of rotation, velocity and number of steps (available only in positional mode). The first bit of message1 is start bit (1 for start of

rotation and 0 for stop). The second bit represents desired mode of operation (1 for velocity and 0 for positional mode). The third bit determines direction of rotation. The last five bits of message1, together with message2, represent desired velocity (13 bits for velocity). Message3 and message4 are low and high bytes of desired number of steps. The last m-file ends the serial port session.

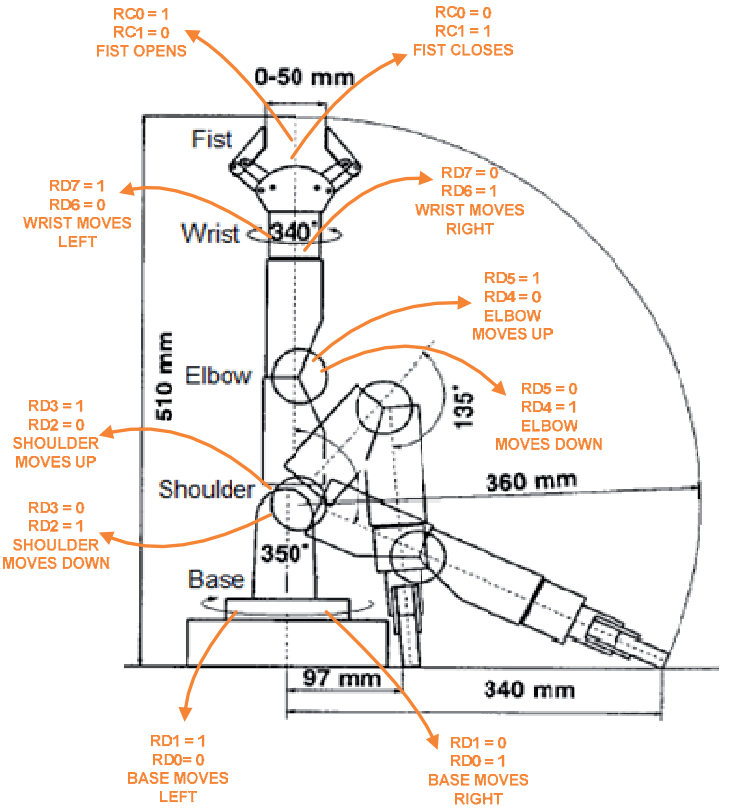


Figure 3. Movement description of robotic arm

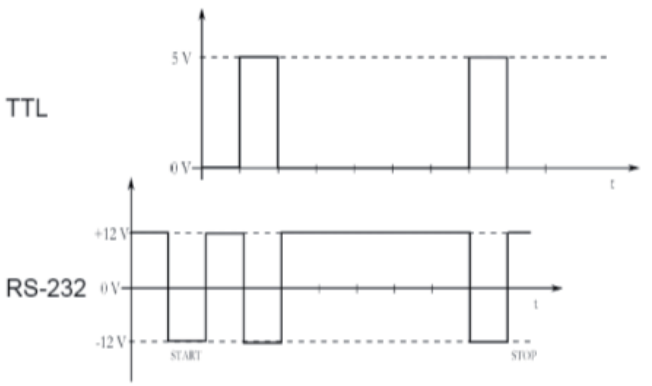


Figure 4. Conversion TTL logic level into RS232 logic level

For the robotic arm system, communication was established through MATLAB using two m-files. First m-file creates serial port and configures its properties. The communication between the server PC and the microcontroller is realized using second m-file. In this m-file, function was created to collect data set by user inside GUI.

c. Remote control via web server

Because MATLAB was used to send control commands to the microcontroller, an additional toolbox called TCP/UDP/IP was installed. This toolbox establishes a connection between two computers (server and client) using the TCP/IP protocol as shown in Figure 5.

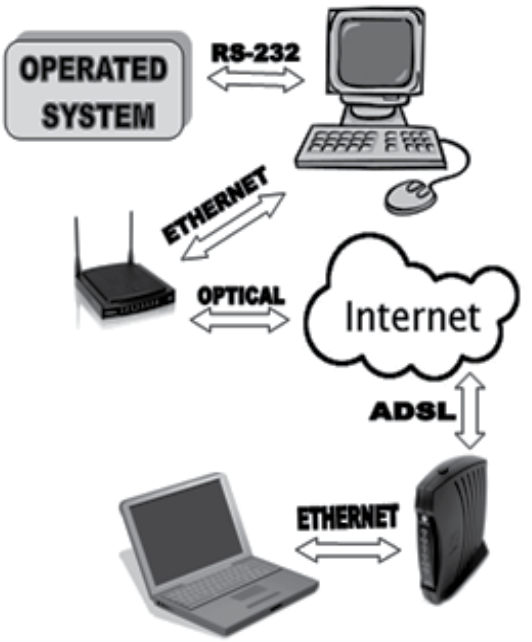


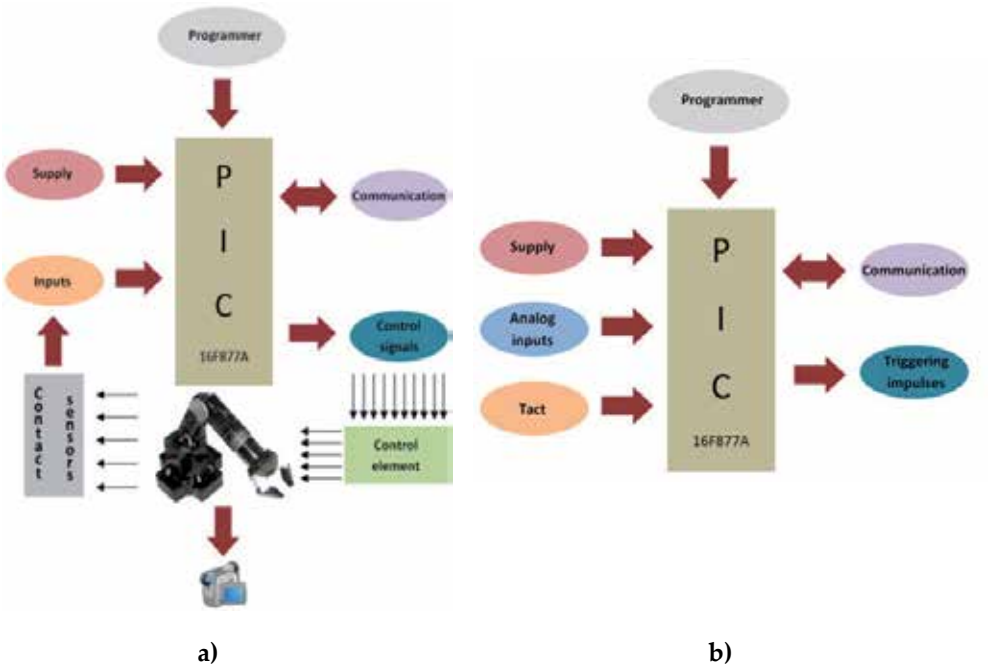
Figure 5. Established connection

One of the computers is used as the server, while the other one is used as a client. The web server is established on the server computer. All commands that control the stepper and robotic arm are sent from the client computer. On the other hand, all feedback needed for acknowledgment of the proper remote control, i.e. did the stepper reached the desired velocity or did robotic arm reached assigned position, is send from the server to the client via established connection. All that was needed to establish a connection was provided in the additional toolbox using the created m-files such as pnet, pnet_putvar and pnet_getvar. Using the pnet m-file a handler called 'con' was created. This handler was used in combination with m-files pnet_putvar and pnet_getvar to send the necessary data to the connection and to collect that data from connection respectively. To access the visual information of the robotic arm acquired by the camera, VNC server was used. The camera is connected to the server computer and by using VNC server we can gain access to the server from a remote desktop.

Two m-files were created (for host and client) using previously mentioned m-files. The required toolbox can be found in the references along with all explanations for each m-file [19].

3. Hardware structure

Figure 6. shows detailed hardware structure of the implemented distributed systems. It is a product of fully independent work.



a) b)

Figure 6. a) Block structure of stepper motor system; b) Block structure of robotic arm system

Production implied the implementation of the entire hardware circuit and construction of work algorithm. The structure consists of a microcontroller and controlling elements for both systems.

The stepper motor system is operated with commands from host PC which are sent through serial connection. Microcontroller interpretates those commands and generates trigger impulses on PORT D. These impulses are sent to unipolar transistors which are combined to form a power amplifier. This power amplifier is directly connected to the step motor. The step motor has an encoder disc mounted to its shaft. The encoder recognizes alterations from encoder disc and sends them as impulses to the analog input on PIC. These impulses are used to determine proper work of the stepper motor. Figure 7. shows the most important components used in this hardware realization.

The robotic arm, shown in Figure 3., has five degrees of freedom modeled after the human arm. The controlling element consists of ten relays (Figure 8.), all assembled in accordance with scheme on Figure 9.

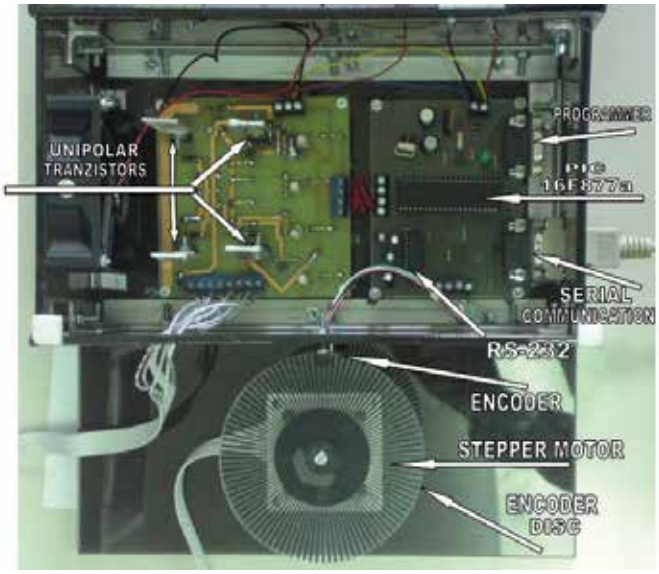


Figure 7. Realized system for stepper motor control

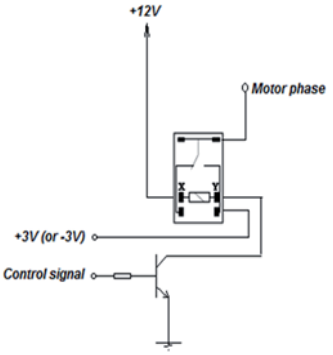


Figure 8. Relay scheme

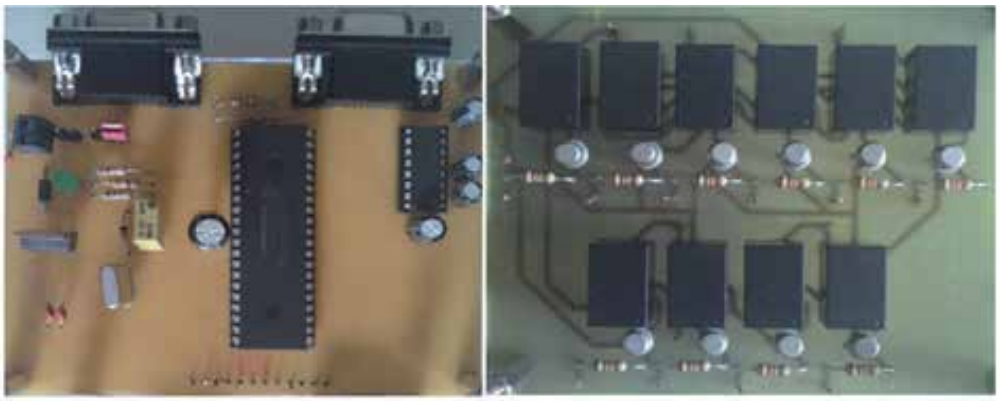


Figure 9. Microcontroller sheme and controlling elements – relays

The operating princip of the relay is simple. When current flows between pins X and Y, it leads to the creation of a magentic field. The influence of the magnetic field causes a change in the position of the switch inside the relay. Control signals from microcontroller are sent to the controlling element. Those control signals need to enable the flow of current which is necessary to trigger the DC motors placed inside the joints of the robotic arm. The system is operated from the client's computer GUI. The commands from server are sent through serial connection. Microcontroller interpretates those commands and generates the control signals. These signals are sent to the controlling element made of relays. The controlling element is directly connected to the robotic arm. The proper work of the robotic arm is monitored via camera. There are also contact sensors on each joint of the arm used to prevent movements that could cause a malfunction. Those signals are transmitted back to the microcontroller and, in the case of a possible malfunction, the movement of the affected joint will be stopped immediately. Figure 9. shows the individual parts of the realized structure and Figure 10. shows the final structure.

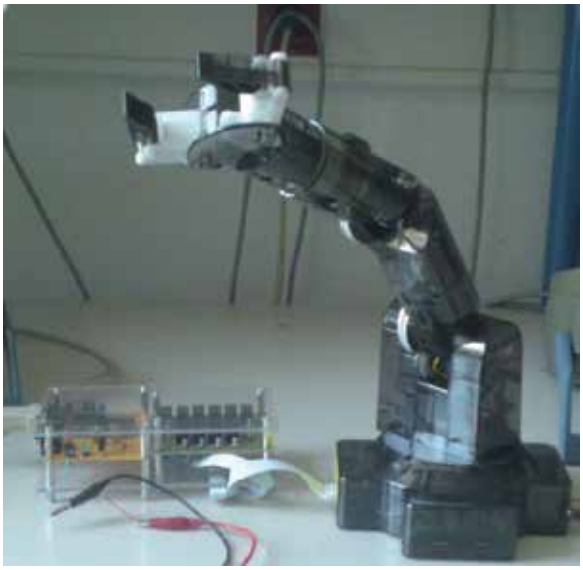


Figure 10. Final structure of the robotic arm system

4. Hardware programming

a. Stepper motor

After making the hardware, the next step was to program the circuit itself. The code is written in CCSC Compiler, version 4, from manufacturer Custom Computer Services Inc. [20]. The user chooses mode of operation, desired direction of rotation, velocity and number of steps using GUI. That information is collected into four 8-bits values and sent to server PC via ETHERNET. Server PC forwards those messages to microcontroller using RS-232. Microcontroller analyses received messages and sets/resets appropriate bits according to chosen mode, direction, velocity and number of steps. In the main program, if START bit is

set, microcontroller generates appropriate sequence of triggering impulses based on chosen direction and velocity. The width of one impulse is given by:

$$T_{width} = \frac{60\,000}{RPM \cdot 200}\,[ms] \tag{1}$$

Timer1 is set to appropriate value so it overflows every 100 ms and an interrupt is generated. In Timer1 Interrupt Service Routine microcontroller sends number of steps made in last 100 ms to server PC via RS-232. Timer0 is used as counter. Output of encoder is led to pin RA4/T0CKI. Timer0 increments on every step. Client PC uses received information about number of steps made, calculate the velocity and shows it in GUI (Figure 11.).

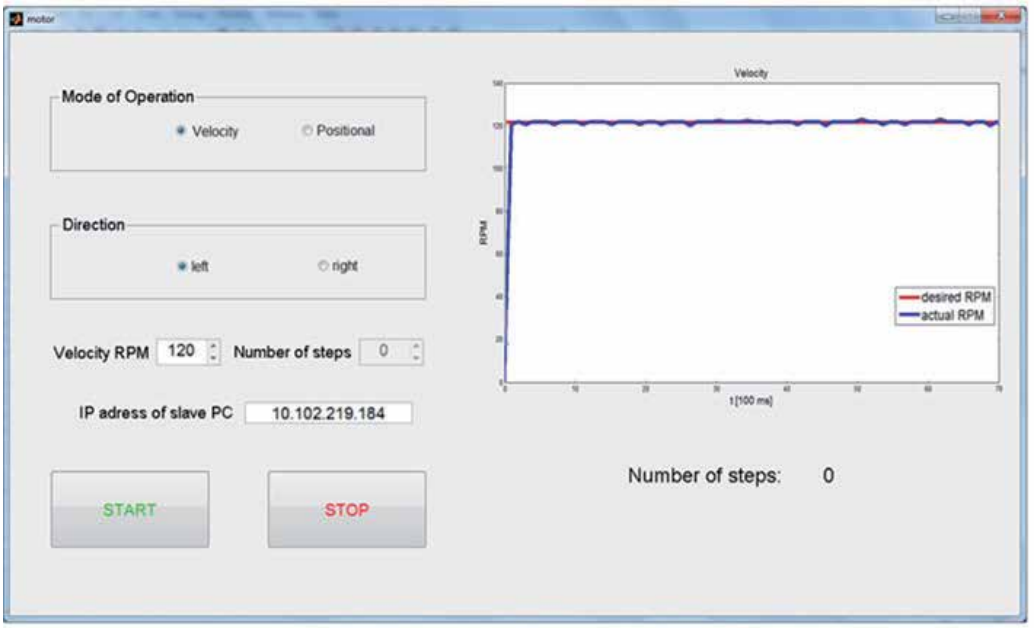


Figure 11. Graphical User Interface for stepper motor system

How hardware is programmed for stepper motor is shown on Figure 12.

b. Robotic arm

After making the hardware, the next step was to program the circuit itself. The code is written in MPLAB IDE v7.5 [21]. Figure 13. shows how hardware is programmed.

To establish communication via ETHERNET, Real VNC program [22] and MATLAB Server are used. Real VNC program is used to obtain visual feedback by camera, and MATLAB Server is used for transmission of control messages from client PC to server PC. VNC Server was installed on server PC (PC connected to realized hardware structure), while VNC Viewer was used by client PC. By running the VNC Viewer and entering IP address of server PC, connection between two computers is established.

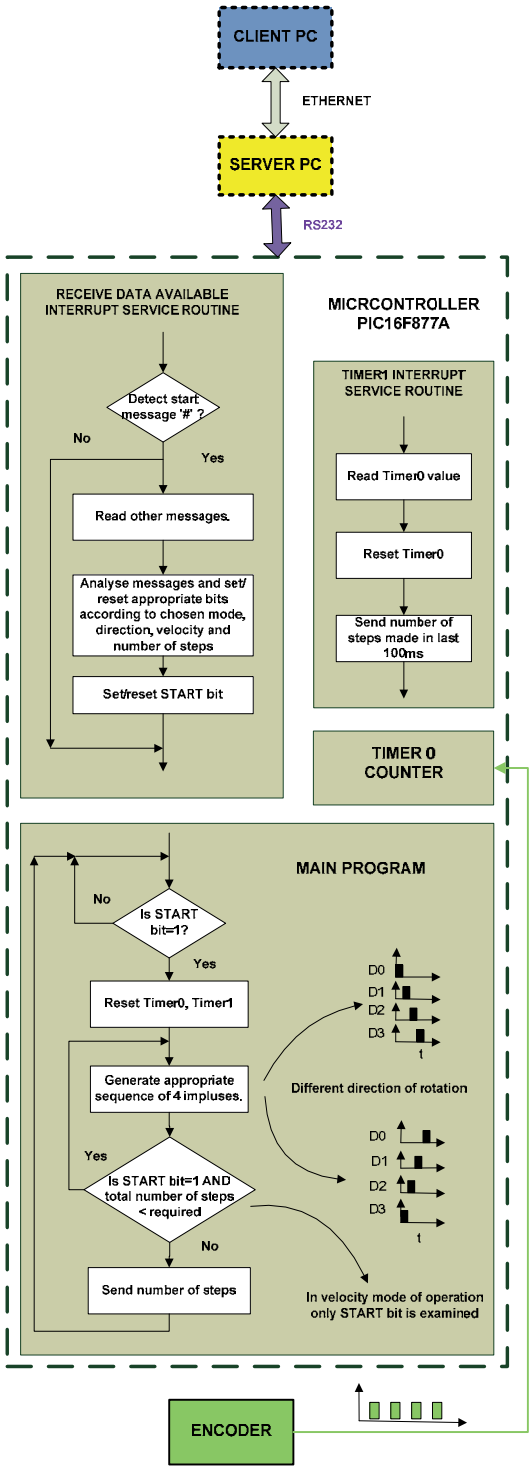


Figure 12. Schematic display of the software for hardware structure with stepper motor

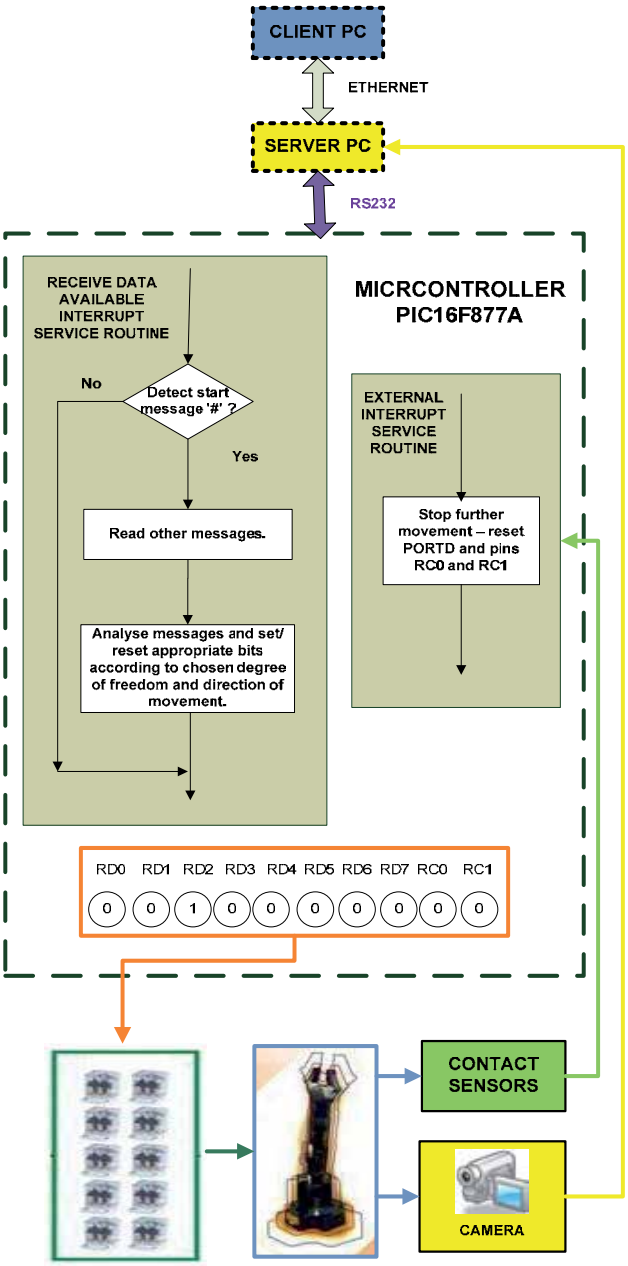


Figure 13. Schematic display of the software for hardware structure with robotic arm

When user chose degree of freedom and direction of movement, function from MATLAB is called. This function codes user's requirement into a short message which is sent to server and then to microcontroller via RS-232. Depending on the content of received message, microcontroller generates appropriate value to the PORT C or PORTD which is described in subchapter 2. section A.

Using the GUI (Figure 14.) the user chooses degree of freedom to manipulate with and the direction of movement.

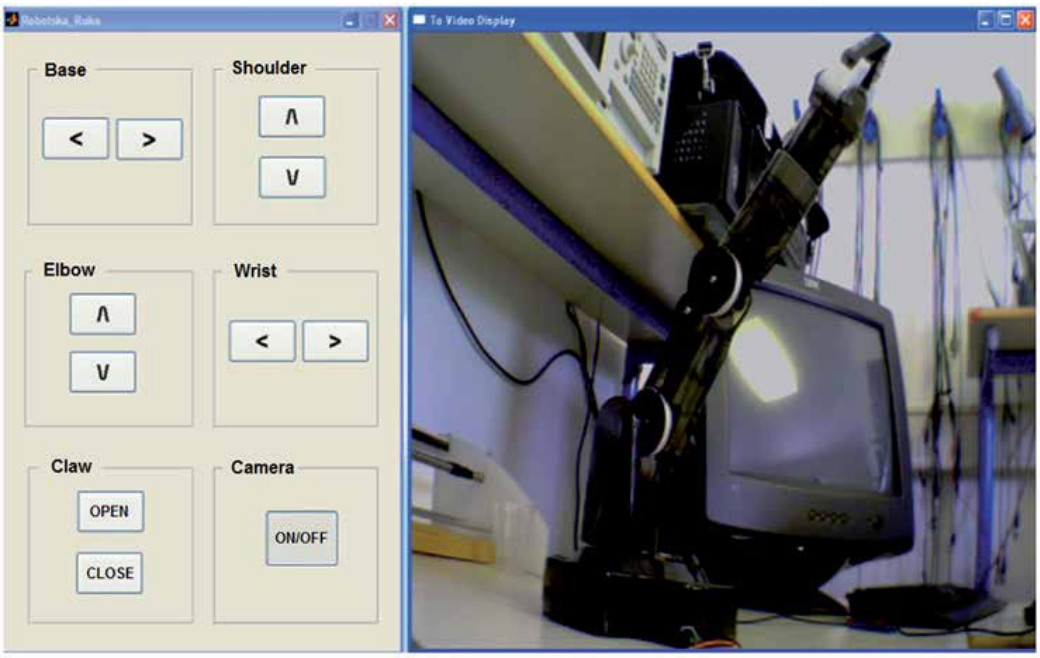


Figure 14. Graphical User Interface for robotic arm system

The user monitor movement of robot arm by camera. Beside this visual feedback, for every DOF contact micro-sensor is implemented to avoid possible damage of the arm. When DOF reaches its final position, the micro-sensor becomes active and further movement is stopped.

5. Testing results

The effectiveness of a stepper motor is rated by the response of a single phase to the provided trigger impulse. The experimental results obtained by applying one impulse are shown on Figure 15., Figure 17. and Figure 19. The responses to certain defined velocities are shown on Figure 16., Figure 18. and Figure 20.

The responses are almost instant as shown on Figure 15., Figure 17. and Figure 19. The small deviation is the result of mechanical characteristics of stepper motor. The noise seen in those figures is caused by imperfection of the encoder and its disc. The response signal collected from encoder is raised by 0.5 (V) because of the encoder's saturation, which represents a nonlinear acting.

Figure 16., Figure 18. and Figure 20. show various measurements of desired and actual velocity. It can be seen that the actual velocity follows the desired one. At certain moments there are deviations of actual velocity. These deviations are caused by restrictions of the

used encoder. The shape of those deviations is the result of linear approximation of the characteristic.

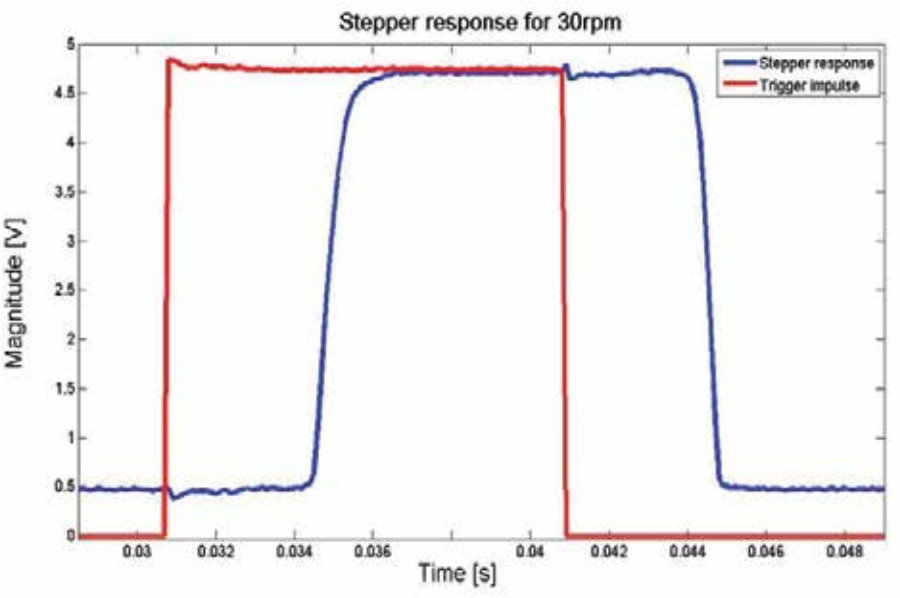


Figure 15. Response to one step for velocity of 30 RPM

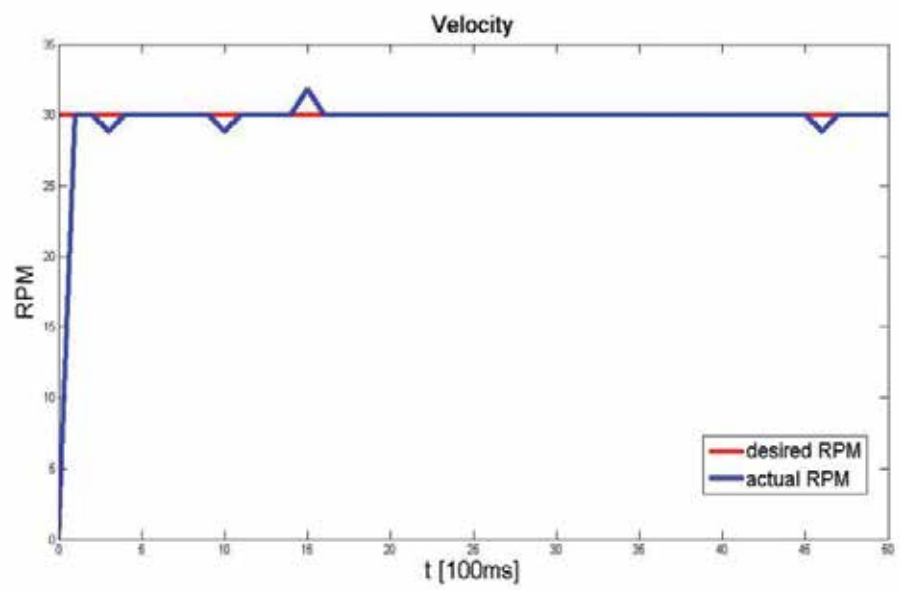


Figure 16. Desired and actual velocity of 30 RPM

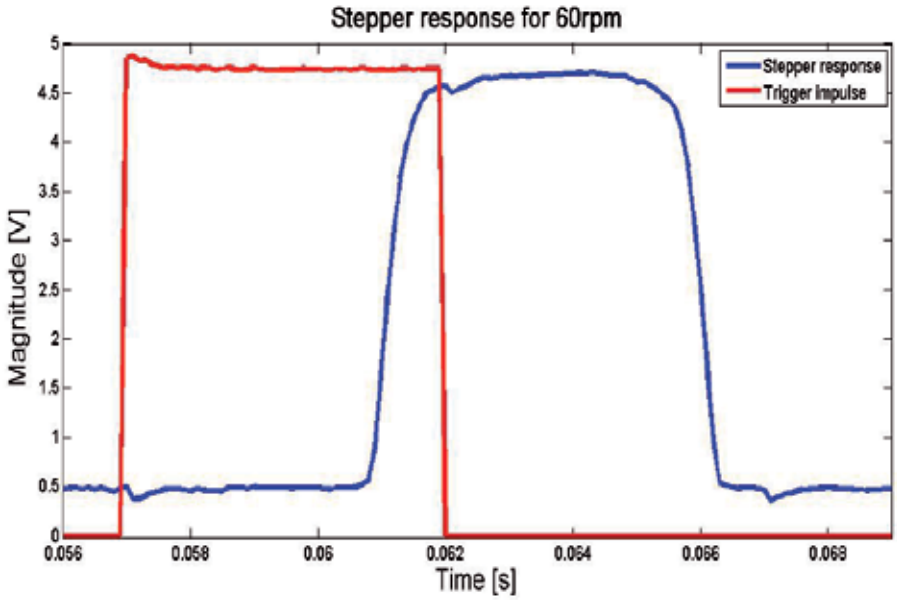


Figure 17. Response to one step for velocity of 60 RPM

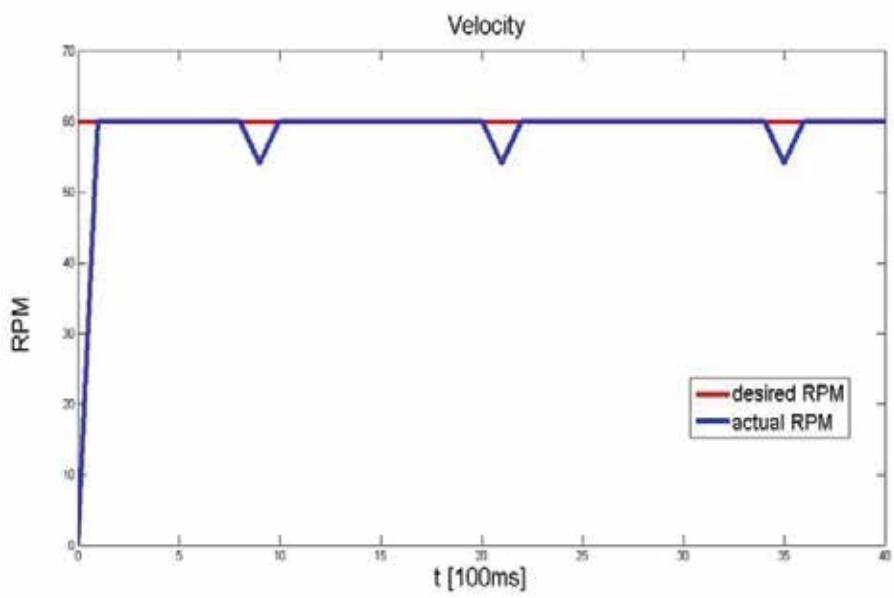


Figure 18. Desired and actual velocity of 60 RPM

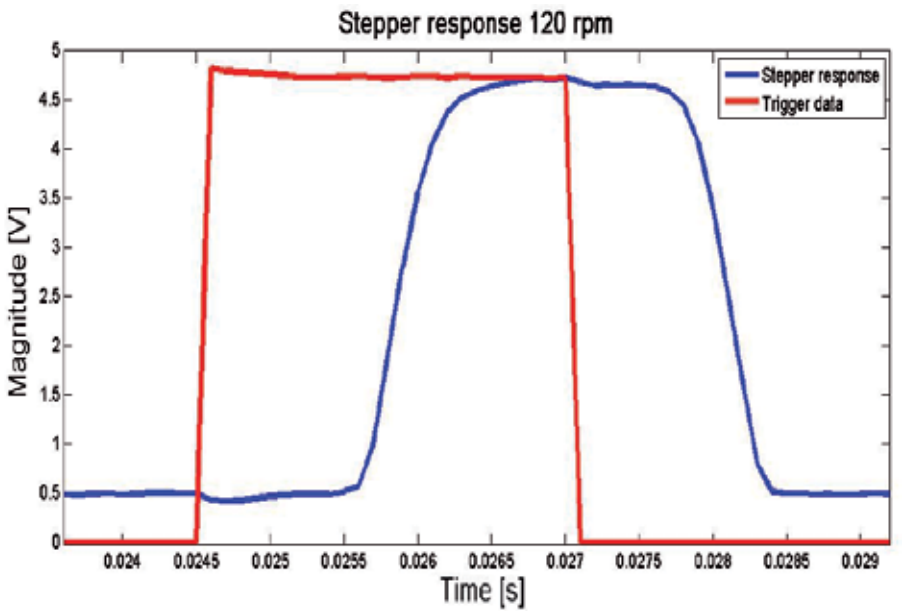


Figure 19. Response to one step for velocity of 120 RPM

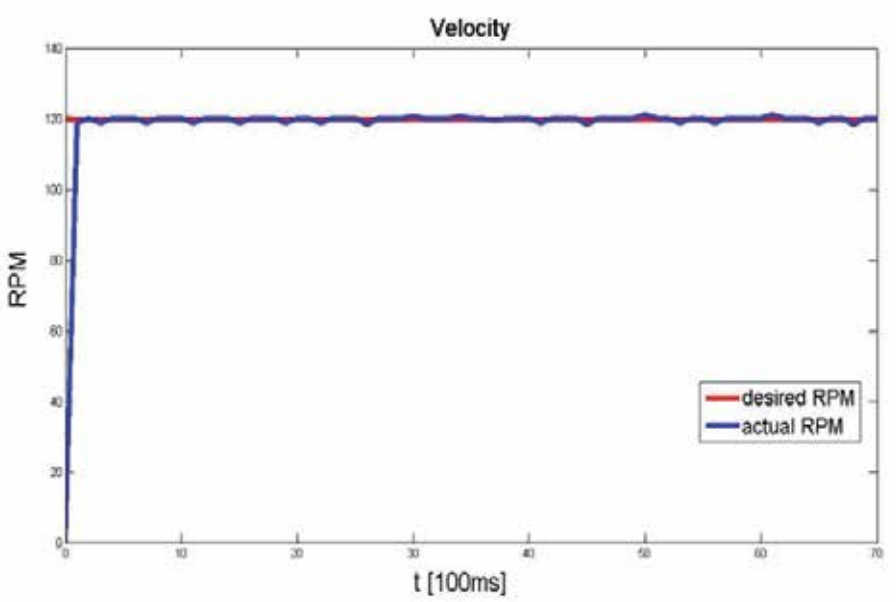


Figure 20. Desired and actual velocity of 120 RPM

6. Conclusion

This paper presented the design of a remotely controlled stepper motor and robotic arm via web server. Operating algorithms and GUI were realized for both systems. Through the GUI for the stepper motor user can operate the motor in two modes: velocity and positional. The feedback received from the encoder is sent through the established connection from server to the client. Experimental results demonstrate the effectiveness of the remotely controlled stepper motor. Using the GUI for the robotic arm, user can operate each joint of robotic arm separately. The feedback received from the camera is sent through the established connection from server to the client. Experimental results are not shown in this paper, because as stated before, this model of robotic arm does not possess encoders. That is the reason why camera was used as visual feedback. The system operates in real time and visual feedback provides us information about current state of robotic arm. System is based on microcontroller and its development is not expensive, unlike the systems which are based on other technologies i.e. PLC. These systems are used in environments which are dangerous for humans.

Author details

Vedran Vajnberger, Semir Silajdžić and Nedim Osmić
Faculty of Electrical Engineering,
Department of Automatic Control and Electronics, Sarajevo, Bosnia and Herzegovina

7. References

[1] T. B. Sheridan (1993) Space teleoperation through time delay review and prognosis, *IEEE Transaction on Robotics and Automation*, vol. 9, pp. 592-606.
[2] C. Sayers (1996) Remote Control Robotics, New York: Springer Verlag.
[3] Velagic, J., Coralic, M. and Hebibovic, M. (2004) The Remote Control of Robot Manipulator for Precise Time-Limited Complex Path Tracking, Proceedings of the IEEE International Conference on Mechatronics and Robotics (MechRob2004), Volume 2, September 13-15, Aachen, Germany, pp. 841-846
[4] D. Lee, and M.W. Spong (2006) Passive Bilateral Teleoperation with Constant Time Delay, *IEEE Transactions on Robotics and Automation*, vol. 22, no.2, pp. 269-281.
[5] Vladimir Lucan, Petr Simacek, Jari Seppälä, Hannu Koivisto, "Bluetooth and Wireless LAN Applicability for Real-time Control"
[6] Xin Liu , Yongtian Wang , Yue Liu , Dongdong Weng, Xiaoming Hu (2009) A Remote Control System Based on Real-Time Image Processing, 2009 Fifth International Conference on Image and Graphics
[7] Chui Yew Leong and Abdul Rahman Ramli, Intelligent Systems and Robotics Laboratory (ISRL), Institute Of Advanced Technology, Universiti Putra Malaysia. 43400 Serdang, Selangor. "Development of a real-time embedded remote triggering and monitoring system with SC12"

[8] G.Srinivasarao, S.Sao, "Security system based on stepper motor control using microcontroller"

[9] Ahmet Altintas, Department of Electrical Education Dumlupinar University, Faculty of Technical Education, 43500 Simav, Kütahya, Turkey, "A graphical user interface for programming stepper motors used at different kinds of applications", Mathematical and Computational Applications, Vol. 14, No. 2

[10] Betin, F. Pinchon, D. Capolino, (2000) "Fuzzy logic applied to speed control of a stepping motor drive", G.-A. Dept. of Electr. Eng., Univ. of Picardie Jules Verne, Cuffies , , Industrial Electronics, IEEE, Jun 2000.

[11] Jan B.A. Habraken, Kora de Bruin, Morgan Shehata, Jan Booij, Roel Bennink, Berthe L.F. van Eck Smit and Ellinor Busemann Sokole, "Evaluation of High-Resolution Pinhole SPECT Using a Small Rotating Animal", Departments of Nuclear Medicine, Radiology, and Medical Technological Development, Academic Medical Center, University of Amsterdam, Amsterdam, The Netherlands, , Basic Science Investigations.

[12] Fang Liu, Zhenlin Hu, Lei Qiu, Chun Hui, Chao Li, Pei Zhong and Junping Zhang (2010) "Boosting high-intensity focused ultrasound-induced anti-tumor immunity using a sparse-scan strategy that can more effectively promote dendritic cell maturation", *Journal of Translational Medicine*

[13] Salcudean, S. E., Ku, S., and Bell, G., 1997, "Performance Measurement in Scaled Teleoperation for Microsurgery," Proceedings of the First Joint Conference in Computer Vision, Virtual Reality and Robotics in Medicine and Medial Robotics and Computer-Assisted Surgery (CVRMed-MRCA '97), Grenoble, France, pp. 789–798.

[14] Jasmin Velagić, Nedim Osmić, Semir Silajdžić, Tarik Terzimehić and Vedran Vajnberger (2010) Remote Control of Stepper Motor via Web Server, Conference on Control and Fault – Tolerant Systems (SysTol'10), Nice, France, 10/2010.

[15] Vedran Vajnberger, Tarik Terzimehić, Semir Silajdžić and Nedim Osmić (2011) Remote Control of Robot Arm with Five DOF, International symposium for information and communication technologies, electronics and microelectronics – MIPRO 2011, Opatija, Croatia

[16] Liptak, Bela G. (2005). *Instrument Engineers' Handbook: Process Control and Optimization*. CRC Press. pp. 2464.

[17] Yidong Wang, Kaiguo Yan, Guozi Sun, Peihuang Lou, "Serial Communication in DNC Information Systems", The CIMS Center of NUAA, Nanjing 210016, China, Published by Moxa Technologies)

[18] Yidong Wang, Kaiguo Yan, Guozi Sun, Peihuang Lou, "Serial Communication in DNC Information Systems", The CIMS Center of NUAA, Nanjing 210016, China, Published by Moxa Technologies)

[19] Peter Rydesaeter, TCP/UDP/IP toolbox 2.0.6. user guide, March 2001.

[20] htttp://www.ccsinfo.com, User guidance

[21] MPLAB® IDE USER'S GUIDE, 2005 Microchip Technology Inc.,

[22] Personal Edition VNC Server 4.4 User Guide, 2008.,

Section 3

MATLAB for Educational Purposes

Chapter 14

Education of Future Advanced Matlab Users

Michal Blaho, Martin Foltin, Peter Fodrek and Ján Murgaš

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46465

1. Introduction

Technology surrounds us in every step we take. Computers, mobile phones or cars would not exist without smart researchers and innovators. We need more and more engineers and scientists to sustain technological growth. The current state of technology and technological growth is facing two problems. The first problem is population declining in many countries. Germany is a country with least population growth in Europe, for example. The second problem is that many industrial and research centres demand high quality engineers (Blau, 2011). These two problems come hand in hand as you can imagine. Society can affect both of them in some way but providing high quality engineers is what we can affect at fastest. Learning is an important process in our lives. Educational institutions prepare students to fulfil requirements from the society. There are many modern methods and technologies for learning but most of them have one in common today – computers.

Computer aided learning has found a way in learning process from primary schools to universities (Abdullah et al., 2010, Bertrand, 1989). Computers are also significantly involved in teaching technology serving sciences like mathematics, physics and information technology. Technical computing plays an important role in these specializations. Many software applications are accessible for teaching as Matlab, Octave, Scilab and Mathematica, for example. Other specialized programs exist as well, but they are often used to solve specialized problems and tasks. Therefore we are not going to mention them. The worldwide most spread applications at educational institutions used by is Matlab, which is considered as standard in technical computing and science (The MathWorks, 2012a). Matlab is a very powerful tool for computing and simulation. Basic mathematical core provides functions for high performance computing. On top of that, Matlab provides add-ons (toolboxes) to enhance its usage via adding more functions in specialized fields of technology, economics, medicine or biology. Matlab is also applied in many publications in different fields. Matlab is not only exploited in computations but also in the process of teaching and learning. In the Matlab environment there are small GUI applications that can be created to improve learning (Andreatos & Zagorianos, 2009).

Extensive usage of Matlab demands high quality courses at educational institutions (Dogan, 2011). In fact multiple courses for different topics can be taught with the use of Matlab. To provide students with solid background for these topics, an introductory course for Matlab can be created. At our faculty we create such a course. The aim of this course is to gain basic skills that students can use in more advanced courses. Our course is divided into two parts. In the first part there are Matlab basics that every student must to know (Matlab basics, graphics and Simulink). At top of the basic specialized topics a more advanced courses are introduced to help during teaching topics that students will need later. Although we teach the Matlab course at our faculty we believe that Matlab can be used at secondary schools as supporting tool for the teachers. Matlab basics and graphics can be also taught to students at secondary school level (M. Varga & Z. Varga, 2010).

Students have different types learning capabilities. We cannot satisfy all of them, but we could motivate them to achieve the best possible performance. One of the proper motivator is their evaluation (Blaho et al. 2010a, 2010b). If students can see that the hard effort is rewarded than they want to learn more and more. The other motivator is collaboration. Many students are competitive by nature and they want to achieve better performance than others. On the other hand the collaboration can help them to solve problems that they are not able to solve for hours or days. Therefore collaborative learning is a very important for them because students learn to work in the team as well.

The Internet is popular among the students. Students search more and more for information on the Internet. Studies show that many students consider the Internet as a great source of information and using it during learning process. The Internet is full of e-learning projects, documents, presentations and multimedia content that students may find useful for their learning process (Foltin 2012b, 2012c). It is necessary that all our contents for courses are available online as well. Courses can be also supported with custom video tutorials where topics are explained and shown. Mathworks also provide online content like the documentation, webinars and events on their official web site. Conferences are also good events for students to exchange their knowledge among each other (MathWorks 2012a, 2012b, 2012c). Learning and teaching process is often high cost. Free software and open source applications are possible solution (Foltin et al. 2011).

In this chapter we will describe possibilities for an introductory Matlab course. We will show how to divide lectures and practices into several topics necessary and helpful for the students. Weekly task evaluations are proposed to motivate students to achieve better results. We will described how to find an interesting resources for the course or how to create own one. Collaboration among students is described within the Internet with the Facebook, articles or forums. Over all we focused on the educational use of Matlab, concepts for lecturers and the course improvements.

2. Matlab course

Matlab is a very powerful tool for technical computing and simulation. It is time saving tool for high performance computations. Matlab is optimized for this kind of tasks. It supports

us when we lack of real world system in the education. Matlab contains many add-ons that are very useful in many fields of science. Matlab usage is a very popular in many science projects, bachelor, master and dissertation theses in the university environment. The basic Matlab course was established at the Faculty of Electrical Engineering and Information Technology in Bratislava to teach students to use Matlab and its add-ons. Matlab is a basic tool for every student at our faculty. We try to move this course into the first year of study for this reason. The knowledge, which students take from this course is a very useful in the other technical courses.

2.1. Lectures

As we mentioned before at the Faculty of Electrical Engineering and Information Technology Bratislava there is Matlab used in many mathematical and technical courses. Students use Matlab for their computations and solutions in their bachelor, diploma or dissertation theses. Matlab course is introductory course to Matlab, Simulink and Matlab toolboxes. The faculty has two terms per year. Each term lasts twelve weeks. Course can be taken in second term of the first year of undergraduate study. It consists of one lecture (100 minutes) and practice (100 minutes) per week. The lectures are not compulsory, but practices are. Count of the attendance at lectures is more than a half of all students. The lectures are combination of explanation of the current topic, solving of basic tasks and discussion. All of tasks are solved directly on the computer using Matlab. The students are allowed to see how to solve the concrete problem and can write the notes into their hand-out. The full preprint of lecture is available a week ahead on the web pages of the course. The students can print this hand-out before the lecture. The hand-out with notes are useful source of the information for the practices. For this reason there are usually students who attend the lectures more successful than students who do not have the notes from the lectures. The aim of the Matlab course is to introduce students to the Matlab, to learn basic principles, syntax and solutions of some common problems. We focus on the most common problems which students may use in their future study or on the others courses. We cover these topics:

- Matlab basics and programming
- Graphics and GUI
- Simulink
- LTI continuous and discrete systems
- Non-linear systems
- Identification
- Fuzzy logic
- Virtual reality
- Stateflow
- Neural networks
- Real-time

As you can see, there are many topics for just one course. But with the lecture time we just cover a simple introduction to these topics. That is not a big problem because they will have

detailed courses for most of the mentioned topics in the future. The main idea is to show basic principles and solutions to students that Matlab could provide. They can extend knowledge gained from the Matlab course on the other courses.

2.2. Practices

The practices are related to the lectures. Students are solving some problems similar to the topic on the actual lecture. Problem is at the first solved verbally and using the board. Students then try to find out solutions in the Matlab environment. This is a time for notes from the lecture. Practices are set up in the way that students have not much time to learn the topic on the exercise and they have to always come prepared. Every student has his computer and work alone. They can use the computers with Microsoft Windows, GNU/Linux or Mac OS X operating systems. Students can also use their own notebooks. These facts improve the environment in the class. There are two lessons that take hour and forty minutes together. Teacher checks and rate every students work about ten minutes before end of the practice. The tasks are prepared for students that they should finish the given tasks before hour and half without problem. Many of students can solve given problems successfully in time.

2.3. Matlab at secondary schools

Although we teach Matlab course at university the question is if we could find usage of Matlab at lower levels of education. The answer to this question is of course yes. We think that Matlab is great supportive program for secondary schools teachers and it should be used in mathematics and physics courses. For example let us look at this two equations:

$$x = x_0 + v_0\, t \cos(\alpha) \tag{1}$$

$$y = y_0 + v_0\, t \sin(\alpha) - 0.5\, g\, t^2 \tag{2}$$

Equations represents trajectory of an object thrown in 2D plane from start point (x_0, y_0) with speed (v_0) at some angle (α). With given acceleration due to gravity (g) we can compute distance (x) and height (y) of objects trajectory at certain time (t). In the next figure you can see trajectories for different angles with same start point, speed and gravity.

In the most of the secondary schools teachers and students compute results on the table or the paper. Matlab can be used to visualize different trajectories in just few lines of code for better understanding and imagination of the problem. With additional commands we could animate whole movement. This is one of the examples how to use Matlab in secondary schools physics classes. Another question is if Matlab should be considered only as a supporting program for the teachers or if students should also learned know how to create own scripts and programs. We think that we should consider both. Benefits of the first part of the question were answered with our example before. Writing Matlab scripts and programs should not be a problem for the students also because of its simplicity. We think that for students it would be also interesting. They would also learn some basic concepts of algorithm writing.

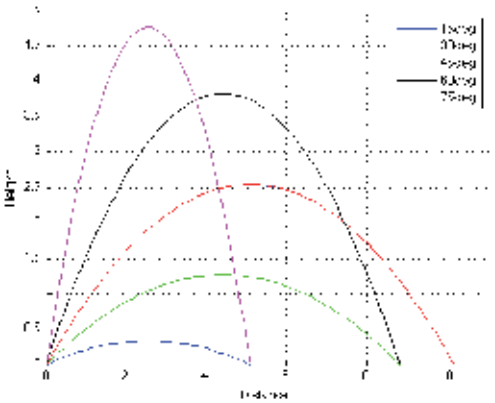


Figure 1. Trajectories of thrown object

3. Matlab course topics

In previous chapter we generally talked about the Matlab practices and lectures at the universities. We briefly showed how you could use the Matlab benefits in secondary schools also. This chapter will discus topics suitable for basic Matlab course. We divided topics into three categories. Essential topics are parts of Matlab that every student should know because they provide strong platform for advanced topics and other Matlab toolboxes. Advanced topics should cover additional parts of the Matlab necessary for later student courses. We will describe topics that we cover in our course but other interesting parts of the Matlab suitable for other study programs are described in other interesting topics.

3.1. Essential topics

With Matlab core components come several add-ons as well. It is necessary to know all about basic Matlab concepts before we start to use specializes toolboxes suitable for our needs. In our opinion the essential topic for advanced Matlab users should be Matlab basics, graphics, Graphical User Interface and Simulink. Next we will talk in the detail about each of them and on what we should focus at most.

3.1.1. Matlab basics

Understanding of the Matlab basic principles is key to be successful advanced Matlab user. Students should know how to use Matlab windows (Command Window, Workspace, History) and the benefits of each of them. Effective usage of Matlab help system is also necessary to solve Matlab warnings and errors. It is our choice if we use fast help command or Matlab Product help where user can find detailed information. Basic computations can be made in the Command Window through statements or in scripts (M-files) for example area of the circle. For additional support we can use the built-in Matlab functions for complex numbers or trigonometry functions. What makes Matlab powerful tool are vectors and

matrices. There are several ways to create and manipulate them. Working with indexes of matrix is a very important and challenging parts in the Matlab for students, even harder with indexes range. Programing in the Matlab with functions, loops or conditions is one of the highlights of Matlab basics. In our course we are creating simple sorting algorithms. Students must then use almost every part of Matlab basics that we mentioned in this section. We are also teaching advanced data types – structures and cells. Data types are often used as types for output or input arguments to several Matlab functions.

3.1.2. Graphics and graphical user interface

Functions and scripts provide sets of commands necessary for computing output data. Reading plain data is hard and we need some mechanism to represent them. Matlab contains several plotting commands for 2D and 3D figures. In 3D plotting we need to create grid for 3D space through meshgrid command. Some students have a hard time to understand this concept. Changing plot properties is another important knowledge. We must create handler to plot and using get and set commands we can change line colour or type. We can also use standard Matlab handles like current axis (gca), current figure (gcf) or current object (gco). Matlab enable creating graphical user interface. With objects like buttons, labels, inputs or check box we can create interactive experience for users with no prior Matlab knowledge. Students like creating user interfaces, but concept of the handles structure and callbacks can be sometimes difficult for them. We are trying explained this concept clearly and showed them how to call callback from another to reduce necessary commands.

3.1.3. Simulink basics

We can start model dynamical systems after Matlab basics lectures and practices. Dynamics of any system can be expressed with differential equations and computed by ode solvers. With Matlab ode solvers users must write functions with differential equations. More natural way provides the Matlab extension Simulink. In Simulink we building simulation schema from graphical blocks, signals and then customizing parameters in the blocks. Simulation can be continuous or discrete. After Simulink basics we create simple DC motor model schema from differential equations of the mechanical and electrical parts. Critical part for the students is relation between variable and variables differential, how this is modelled through integrator block and how to set initial conditions. On the practice students are modelling following equations:

$$x_1' = x_2 \tag{3}$$

$$x_2' = -g/l \sin(x_1) - k/m\, x_2 + 1/(l^2 m)\, T \tag{4}$$

Equations represent simple pendulum differential equations where variables can be mass (m), length of the pendulum (l), acceleration due to gravity (g) and k is representative of the amount of damping present. Simulation schema for these equations is in the next figure.

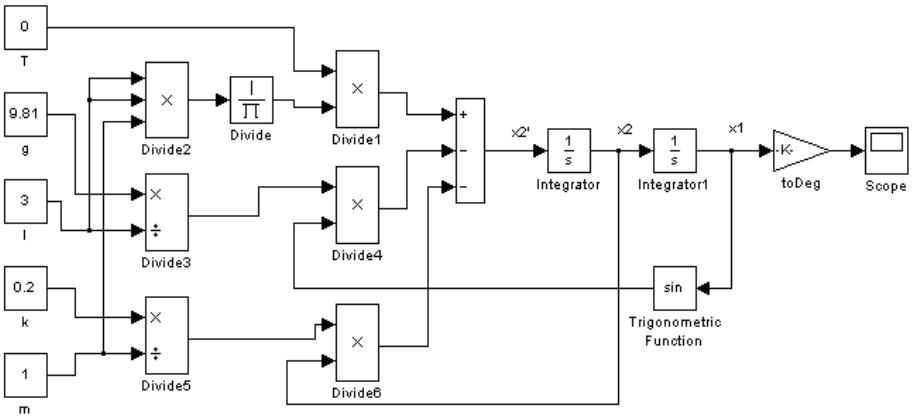


Figure 2. Simulink schema of simple pendulum

Students can change system parameters to simulate how motion of the pendulum change. Standard simulation is with no input (T) and 90 degrees as initial pendulum position.

3.2. Advanced topics

After the essential topics lectures and practices it is time to choose advanced topics. Advanced topics depend on a study program of the students. Because our students studying cybernetics and robotics we focused after basic topics to topics related to their curriculum. In this session we are describing some of them next.

3.2.1. Identification

In real world we often do not know structure of the systems (gain, constants, etc.). Systems can be identified through regression methods with input and output data. Regressions methods are part of the Matlab in System Identification Toolbox. Widely used model estimations are ARX and ARMAX models. Another possibility to achieve approximation of the system is with neural networks. In our course there we use both of them to introduce basic principles of both methods for the system identification.

3.2.2. Virtual reality

Signal values from the Simulink models can be plotted into charts. These signals can be also used for animation of models and Simulink 3D Animation provides this capability. Virtual Reality Mark-up Language represents 3D models using basic shapes and properties like geometry, appearance or translation. Virtual reality worlds can be built using several programs but the Simulink 3D Animation includes V-Realm Builder. Build 3D world is saved as *wrl file and with VR Sink Simulink block we can interact with our simulation. Building 3D animations is one of the interesting topics of our course for students and they are really enjoying it. With the differential equations from previous practices we are

animating simple pendulum that oscillates from right angle to equilibrium. Visualization of this system is in the next figure.

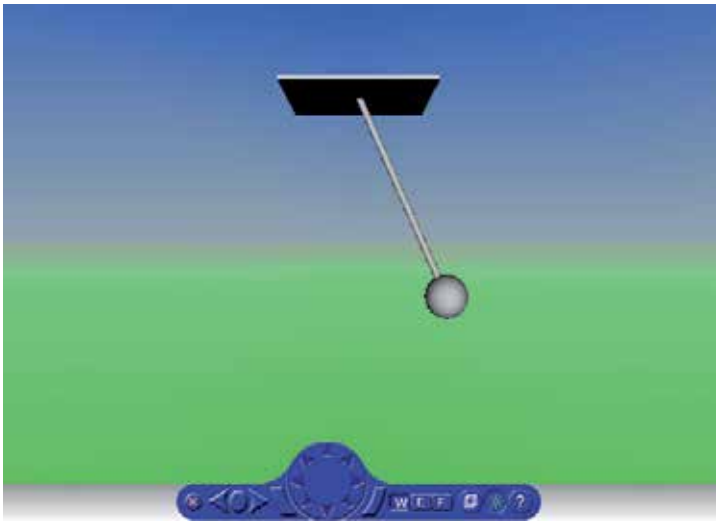


Figure 3. Trajectories of thrown object

3.2.3. Fuzzy logic

Describing behaviour of real world systems with differential equations is often very challenging. Humans can control cars without knowing its precise description. Fuzzy logic can model systems in the terms of If-Then rules natural to our thinking. Inputs and outputs to the fuzzy system are modelled through fuzzy membership functions. In our course there we introduce students into the basic concepts of fuzzy logic and then they modelling some decision problem. One of the problems is student own grading system where there are two tests, one exam and they must write rules based on the point gain for each one of them. Some interesting grade scales came often as result.

3.2.4. Stateflow

Most of the systems can be described as continuous (variables depends on time) or discrete (variables are time-sampled). Some systems react to events and we do not know exactly when system change happens. These systems are called event-driven. Event-driven systems are important to know because they can be found in the industrial applications. They can be modelled by several methods like Petri nets or Statecharts. Matlab prefers Statecharts through Stateflow, which is design environment for developing them. The states and transitions between states are represented graphically. Graphical representation is very useful within simulation where user can see which states are active and when system reacts to event through the transitions. For the Stateflow practise we are asking students to observe real word and look for examples of event-driven systems. They came out with interesting observations like the elevators, intelligent traffic light sensors or door openers in public

transportations which all react to the human touch. We are choosing and modelling one system into the Stateflow together.

3.2.5. Real time

Simulation in the Simulink does not respect real time. If the Simulink schema is simple to the simulation it will be faster than in real time and if schema is complex simulation will be slower than in real time. If we interact with real systems though the Simulink we must ensure precise timing. Interaction with the real time systems is necessary for the measuring physical values (inputs) and reaction to them through outputs. Port of computers can be used to do that with sound card, USB, parallel or serial port even with specialized measuring cards. Real Time Toolbox that is not part of the Matlab can provide real time in Simulink (Humusoft s.r.o., 2012).

Real Time toolbox is only available for the 32 bit Microsoft Windows operating systems and can provide precise timing (standard usage is around 0.1s). Another real time communication can be made by OPC toolbox. With OPC toolbox we can connect, read and write data to OPC server. With large amount of OPC server distributors we can communicate with almost every Programmable logic controller widely used in praxis. For the educational purposes OPC server simulators can be used for example from the company named Matricon (MatriconOPC 2012). This server generates signal that we can read and draw in the Matlab/Simulink. We can also write our data to provided space of the server and then to the process or controller.

3.3. Other interesting topics

In the advanced topics there we described topics that are useful for students at our faculty. There are many toolboxes suitable not only for engineering study programs but also for other like economics (Financial Toolbox, Fixed-Income Toolbox, Financial Derivatives Toolbox) or natural sciences (Bioinformatics Toolbox, SimBiology). Toolboxes that we did not mention in this session can be introduced in other courses, bachelor or master theses. We think that future is in code generation for various devices (FPGA, Embedded devices), or hardware in the loop simulation but variety of the Matlab toolboxes specialised for other topics are popular as well.

4. Tasks evaluation

Results of the students are evaluated through point assigning for each course independently at the Faculty of Electrical Engineering and Information Technology Bratislava (Blaho et al. 2010a, 2010b). Students can achieve 100 points at most. Most times points are divided into two parts. The first amount of the points can student achieved for their activity at practices or tests through term. The second part they can achieve on the exam test. In the most cases it is 30/70 (practice/exam) or 40/60. Weight is on the exam as you can see. Maximum amount of the points allowed during term is 50 for the undergraduate courses.

4.1. Week task

There are two main evaluation (or result classification) systems. The first system classifies students via two or three checkpoints through term. All comes to these tests at the end. Point results depends on how successful are students to answer the test tasks. It is a little bit problematic when students doing great at the practices, but they vacillate on the tests at the end of course. They also do not have to learn until a test is in sight. This problem had been observed long time ago at the mathematics teaching. Students came from high schools and they were not used to learn so much or they had different knowledge background of the mathematics. Many of them fail some of the basics mathematics courses.

Mathematicians try another pointing system that "forces" students to learn more often than just before the tests. Students were evaluated at the every exercise. On the next exercise they had small test on previous topic. The results of the student grow significantly. We also used this schema. We had two reasons for that. One reason was that we had many topics covered in our course. It would be not useful to test student on topics they forget long time ago. Another was that we want to "force" students to do something more. We choose to give more points during the term, so we give 40 points maximum for the practices. There were 10 practices for 4 points each. The first and the last were not evaluated. You can see results from year 2011 course classification distribution in the next figure.

With weekly tasks evaluation students gained good amount of point that they carry to the exam. This type of evaluation does not fit everyone and in the next part of this session we will be talking about its advantages and disadvantages.

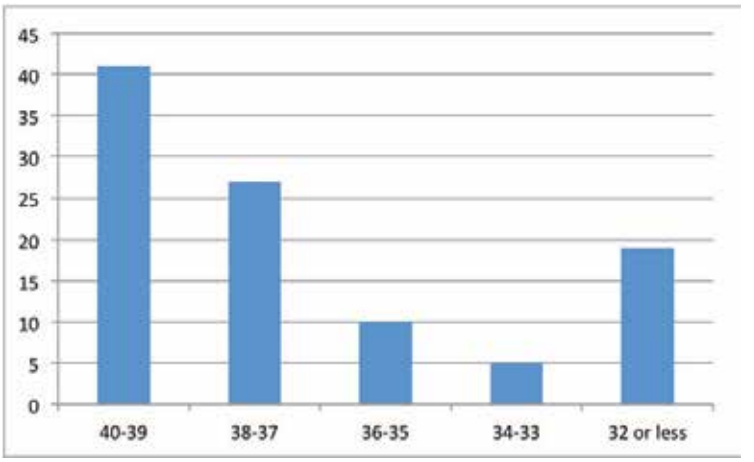


Figure 4. Point distribution for practices

4.2. Advantages and disadvantages

Every student has different capability to learn. We cannot choose unified methodology to learn them all that we know. But with proper motivation they will be willing to learn by themself more within their free time outside the university campus. Our week task

evaluation is kind of motivation factor. Reachable classifications are motivators for being excellently prepared for practices. Now it is time to talk about advantages and disadvantages of our method. We think of these advantages:

- Students must come prepared for practice
- Student are evaluated for what they learned on the lectures last week
- No homework
- Students will also learn work under time limit and stress

Of course every method has some disadvantages. We think of these disadvantages:

- Limited time for educating outside of exercises
- Collision with other courses tests
- Limited time for finishing tasks
- Work under stress
- Problem with student absence on practice (small ability of repeat the same practice)

This method has also some advantages and disadvantages as you can see. The question is if this system works for students and if it is acceptable for them.

5. Study materials

Computers really changed our lives and today are almost impossible to imagine life without them. We often use them without knowing it. Computer by itself is a really powerful tool for computing, but for many years also for communication and sharing information. In fact, the basic idea for the Internet was collaboration across the continents with other researchers. We significantly use this tool today not only for sharing information but also in the teaching process. E-learning is most popular and modern activity in this process. We are also using web technologies for improving our student's results. Although possible we do not use any Learning management system (LMS).

5.1. Open source technologies

For some student education can be expensive even if it is for free (like in our country). Reduction of the costs is our priority too and we try to be supportive for them. We are trying to work with open source technologies that are comparable to commercial technologies. We support students to use free UNIX-based operating systems (Linux distributions). For creation of lectures and practices we use Open source productivity suites like OpenOffice.org or LibreOffice. They are used to exchange documents and spread sheets. All presentations are created using these office suites.

5.2. Course online study materials

Internet is great source for information exchange as we mentioned before. Modern student's first choice for material search is Internet at the moment as well. For this fact we build web page for the course to make it easy for them to find materials. On our web page students can

find all about the course. They can find contacts to us in case they forgot them, some news for the course and another useful page links. Our web pages are prepared using iWeb software. This is WYSIWYG editor for doing web pages. The advantage of this software is they provide very good graphical user interface. It is not a problem to add the new lectures and important information for students by several clicks. For this reason it is possible to update the web page as often as possible. Web pages are divided into sections. The most wanted sections are about the lectures and practices. Students can find lectures and exercises materials there in electronic form. Also detailed information about both is available from there. A few topics have own video tutorials and students are able to view them at home and they can be better prepared for the practices. Video tutorials have main advantage. Students can directly see what the teacher is doing and therefore that is easier to then to understand topic. Some of the video tutorials are mentioned in references. We would like to publish these free video tutorials in Apple Store iTunes U in the future.

Computers connected to the Internet access are present at the practice room. Students can find their problems solutions on the web pages. On our webpage at which we publish student's articles we decided to write small papers about Matlab in Slovak language. Papers slowly cover the Matlab topics so the students have many tutorials. Beyond the articles discussion to the paper is available. We also publish some experiences on our publishing portal named Posterus (Systémy priemyselnej informatiky s.r.o. [SPR] 2012a).

5.3. Mathworks online study materials

The lectures are not only source of the materials. Students use many different materials at the practices. The main material for Matlab study could be the Matlab reference manual that covers all topics and functions that students can use. Matlab reference manual is also accessible online. Student can search the Matlab topics and functions help without installing Matlab tools. Mathworks support education of all parts of Matlab through various online content (The Mathworks 2012a, 2012b, 2012c). Online events called webinars are recorded regularly and they are available for later use. Webinars are categorized by date of the recording, application, product and industry. This helps users to fast find necessary contents. Many languages are supported as well and that helps overcome language barrier for the novice users. We also contributed by creating webinar for Stateflow introduction.

5.4. Seminars and conferences

Online content is very useful but human interaction is important as well. This can be done on events like seminars and conferences. Seminars take place regularly by local Matlab providers just like webinars. Participants from other institutions come to seminars to learn and discus about related topics. Another types of interactions are conferences where students or scientists can present their work. We are trying to participate at Matlab conferences to extend our knowledge and trying to incorporate student in them as well.

5.4.1. Conferences for students

Conferences for the students are great place to present their results. At our faculty students can present their work at two conferences. SVOC conference is for the students from bachelor and master level of study. On this conferences student can also gain some presentation skill that will help them at the final exams later. ELITECH conference is for the students from postgraduate level of study. This conference is more scientific oriented and can be as start conference for other local or international conferences.

5.4.2. Conferences for scientist and educators

The Humusoft Company organizes conference focus on the technical computing every autumn. Past events were hosted at the Congress Center of the Czech Technical University in Prague. Conferences were dedicated to presentations of the MATLAB / Simulink, dSPACE and COMSOL Multiphysics users. Program of the conference also included the latest information on these products and seminar for COMSOL Multiphysics. We participate at this conference and present out results in the education and science. We also motivate students to participate on the conference. Many of them have bachelor or master thesis in Matlab. We try to direct them to work on interesting topics that are worth to present on the conference. One or two papers at the conference come from our student every year.

Every year Mathworks is hosting global event. MathWorks Academic Virtual Conference is joining professors, educators and Matlab users from the praxis. The aim of the conference is to bridge the gap between theory and practice. Participants are learning how to incorporate latest MathWorks technologies into the classrooms and laboratories. Conference is divided into the three regions (Asia-Pacific, Europe, America) for people from the different time zones. Between the presentations it is the time to visit the exhibition hall. At the hall participants could visit booths to chat with staff, view products demonstrations and download literature. Two kinds of booths are available – MathWorks booths and Participating partners booths. We actively are participating on this conference. After presentations we discussed the topics and try to ask the questions online to the speakers. In the exhibition hall time we joined some booths to talk with MathWorks experts about topics that we are interested in.

6. Interaction of students

Collaborative learning is a very interesting area. Students are helping each other to better understand some topics. Main advantage of this method is that student did not stuck and some part of topic that he did not understand. This is much more times efficient and students can learn much more. This helps collaborative learning type students as well as individual learning type students to achieve better results.

6.1. Matlab forum

To support collaborative learning we founded Matlab forum. Students can ask any Matlab related questions on the forum. It is not forum just for our course. The questions can be

related to various problems at which students came through on their study in every course and are about Matlab. We answer most of the questions, but there are some others responders that can easily support us when we have not enough time. They include local Matlab software distributor employers. Now students from different faculties around Slovak Republic and Czech Republic come to our forum and ask questions about Matlab. Foreign students are also welcomed as they can ask us in English as well (SPR 2012b).

6.2. Facebook

Speaking of the Internet and collaborative learning we must also mentioned social networking. Social networks are very popular today. Majority of people are present at least at one social network. Most favourite social network is the Facebook today. Like the Internet, Facebook also started on the academic field. The website's membership was initially limited by the founders to the Harvard university students at Cambridge, Massachusetts, but was expanded to other colleges in the greater Boston area. Facebook enables to create groups. Groups can be created by topics like interests or activities. We established new Facebook group for the students of the Matlab course (Foltin 2012a). This group has several active members who are mainly students of our course. We use this group for quick and short information about the course. Every member can comment tasks from the practices and lectures. It is very useful feedback from students to teachers. Lot of students use this group for publicity their works. They want to show results of their work to other students and share the algorithm. It can be a platform for new projects in the future. It can be also used as a sort of learning management and e- learning system.

7. Results and opinions of the students

To improve our course even more we study the students study results and their feedback every term. First of all we compare distributions from practices and grade results. Then we are giving some questions to the students at the end of the term. Some information and feedback we have already gained through the Facebook and Matlab forum during the term.

7.1. Results

The figure 4 shows that students did very well in tasks during term. Almost half of them gained maximum points. But the question is: How well did students at exam succeeded? Firstly we must look at the exam grades and the relevant points for that grade:

- A – excellent results with minimal errors (100 – 92 points)
- B – above-average results with less errors (83 – 91 points)
- C – fairly well, average results (74 – 82 points)
- D – satisfactory results but with significant errors (65 – 73 points)
- E – sufficient results that satisfactory minimal criteria (64 – 56 points)

- FX – insufficient results, completing of the course expect more efforts and knowledge form student (less than 56 points)
- FN -student did not come to the exams test.

Points are sum of points acquired during term and exam. Next figure shows final grade for our students.

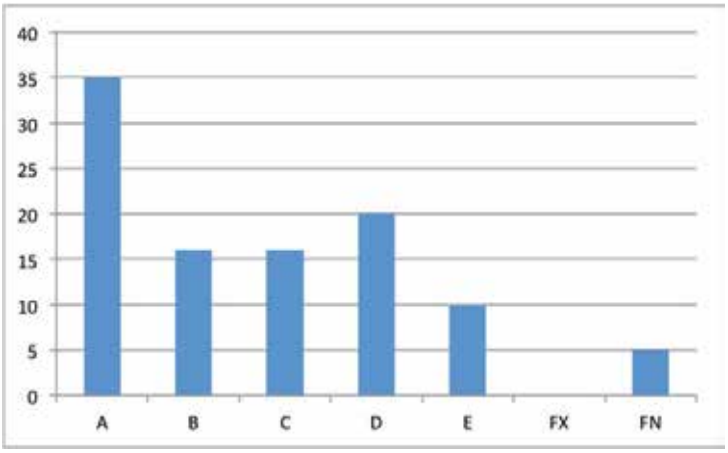


Figure 5. Grade distribution

As you can see there is a change between graphs at exercises and at exam. From the first graph we would expect more students with better grades. We have to ask our-self, what could cause such difference. The first reason could be exam itself. On exam students can acquired up to 60 points. It is a lot education classification so they have to be divided into several questions. The questions have to reflect whole course knowledge so students have to repeat and learn more at the exam. The second reason could be psychological. The good educating result from the exercises could not motivate students enough. Student with high quality result probably thought that they have enough grade of result to pass exam and did not needed to get better grade. Maybe they concentrate elsewhere to pass on the other courses more.

7.2. Opinions of the students

Most valuable information in control comes from the feedback. Such data comes from student responses to the anonymous survey. Our university information system supports such surveys for each course. Most interesting answers are for listed questions:

- How would you rate the lectures?
- How would you rate the practices?
- Were the topics explained clearly?
- How do you rate the course difficulty?
- Do you like the system of work evaluation on every exercise?

The answers depend on the individual students, but most of them agreed at some points. The lectures and practices were rated positive. Explanation was clear, but they want more practical examples. Most of the students answered that course difficulty was normal or course was easy. In the comparison with other courses was that the Matlab course difficult same or easier. The last question was about the evaluation system. Almost everyone thought that this system was suitable. This quiz was taken before exam and therefore the results were not affected by exam results.

8. Conclusion

In this chapter we described the educational side of Matlab usage. Different specializations can benefit from many Matlab products in high quality education. We described how we teach Matlab at our faculty and how are dividing lectures into several topics. We also talked about how to incorporate Matlab into secondary schools education. Topic for future engineers was divided into three categories as well. Basic topics should every engineer to be able to pass, advanced topics depend on study program specialization and other topics are suitable for the research or student thesis. Evaluation of student's performance is a difficult process. In the most cases students are evaluated few times a term. With our proposed evaluation student are forced to study before every practice. Internet is first choice for seeking information for students. It is necessary that we adapt to this fact. We are preparing much online content for students like articles or videos. Collaboration between students is beneficial and we should use social networking. Events like conferences and seminars are helping with education as well. Student's opinion showed that proposed evaluation strategy for the practices suits almost every one of them. Our novel approach to the Matlab teaching process is recommended to copy for another universities as well. We hope that this chapter will helped create high quality courses.

Author details

Michal Blaho, Martin Foltin, Peter Fodrek and Ján Murgaš
Slovak University of Technology, Slovak Republic

Acknowledgement

We would like to thank Slovak Cultural and Educational Grant Agency KEGA under contract number 032STU-4/2011 and Slovak Scientific Grant Agency VEGA under contract number 1/1256/12 for their support of this paper.

9. References

Abdullah, K.A. & Hashim, N. & Yusof, Z. (2010). The development of computer-aided learning for computer numerical control machine: A pilot study, In: *2nd International Congress on Engineering Education (ICEED)*, pp. 94-99, ISBN: 978-1-4244-7308-3

Andreatos, A. S. & Zagorianos, A. D. (2009) Matlab GUI Application for Teaching Control Systems, In: *Proceedings of the 6th WSEAS International Conference on ENGINEERING EDUCATION*, 2009, pp. 208-211

Bertrand, I. (1989). Software engineering techniques for computer-aided learning, In: *Education and Computing*, Vol. 5, Iss. 4, (1989), pp. 215–222

Blaho, M. & Foltin, M. & Fodrek, P. & Poliačik, M. (2010a). Preparing Advanced Matlab Users, *WSEAS Transactions on Advances in Engineering Education*, Vol. 7, Iss. 7, (2010), pp. 234-243, ISSN 1790-1979

Blaho, M. & Foltin, M. & Fodrek, P. & Murgaš, J. (2010b). Research on Preparing Control Engineers and Advanced Matlab Users, *Latest Trends on Engineering Education: 7th WSEAS International Conference on Engineering Education*, pp. 211-214, ISBN 978-960-474-202-8, Corfu, Greece

Blau, J. (September 2011). Germany Faces a Shortage of Engineers, In: *IEEE Spectrum*, 2012, Available from <http://spectrum.ieee.org/at-work/tech-careers/germany-faces-a-shortage-of-engineers>

Dogan, I. (2011). Engineering simulation with MATLAB: improving teaching and learning effectiveness, In: *Procedia Computer Science*, 2011, Vol. 3, pp. 853-858, ISSN: 1877-0509

Foltin, M. & Fodrek, P. & Blaho, M. & Murgaš, J. (2011). Open Source Technologies in Education, *Recent Researches in Educational Technologies: Proceedings of the 8th WSEAS International Conference on Engineering Education (EDUCATION'11) and 2nd International Conference on Education and Educational Technologies (WORLD-EDU'11)*, pp. 131-135, ISBN 978-1-61804-021-3, Corfu Island, Greece

Foltin, M. (March 2012a) Modelovanie a simulácie In: *Facebook*, 2012, Available from <http://www.facebook.com/#!/group.php?gid=313543668309>

Foltin, M. (March 2012b). Vlastnosti grafických objektov In: *Posterus.sk*, 2012, ISSN 1338-0087, Available from <http://www.posterus.sk/wpcontent/ uploads/ML018.mov>

Foltin, M. (Match 2012c). Fuzzy logic toolbox In: *Posterus.sk*, 2012, ISSN 1338-0087, Available from <http://www.posterus.sk/modsim/fuzzy_tbx.mov>

Humusoft s.r.o. (2012). Real Time Toolbox, In: *Humusoft.cz*, 2012, Available from < http://www.humusoft.cz/produkty/rtt/>

MatriconOPC (2012). Introduction to OPC – Tutorial, In: *MatriconOPC.com*, 2012, Available from <https://www.matrikonopc.com/downloads/549/software/index.aspx>

Systémy priemyselnej informatiky s.r.o. (March 2012a). Posterus.sk, In: *Posterus.sk*, 2012, ISSN 1338-0087, Available from <http://www.posterus.sk >

Systémy priemyselnej informatiky s.r.o. (March 2012b). Matlab.sk, In: *Matlab.sk*, 2012, Available from <http://www.matlab.sk >

The Matworks (March 2012a). R2012a Documentation, In: *The Matworks*, 2012, Available from <http://www.mathworks.com/help/index.html>

The Matworks (March 2012b). Mathworks Events, In: *The Matworks*, 2012, Available from <http://www.mathworks.com/company/events/index.html>

The Matworks (March 2012c). MATLAB Virtual Conference 2012, In: *The Matworks*, 2012, Available from <http://events.unisfair.com/index.jsp?eid=1128&seid=23&code=ConfLstng>

Varga, M. & Varga, Z. (2010) Utilizing Matlab in secondary technical education, In: *Proceedings of the 33rd International Convention, MIPRO*, 2010, pp. 970-974, ISBN: 978-1-4244-7763-0

Chapter 15

Using MATLAB in the Teaching and Learning of Semiconductor Device Fundamentals

Ian Grout and Abu Khari Bin A'ain

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46461

1. Introduction

Mathematical analysis tools provide an invaluable (and sometime essential!) tool for use within the engineering disciplines and are readily found in education, research and industrial applications. For example, within the industrial applications, mathematical analysis tools provide an essential aid at all stages of a product development from design through manufacture to test. Although there are a number of useful tools available, since its inception, MATLAB [1] has found a unique role within the engineering disciplines. Given the need to utilise this tool ultimately in both a research context and an industrial application context, there is a need to introduce students at the university level to the effective use of MATLAB, with a focus on the particular discipline area of the student.

In this chapter, the use of MATLAB is presented and discussed within a university education context and in particular the integration of MATLAB into the teaching and learning of semiconductor device fundamentals for electronic and computer engineering students. The aim is to support the student learning of semiconductor device operation, primarily diodes (silicon, germanium, Schottky barrier and Zener) and transistors (bipolar junction transistors (BJTs), junction field effect transistors (JFETs) and metal-oxide semiconductor field effect transistors (MOSFETs)).

MATLAB is primarily used as a data analysis, presentation and reporting tool in this context, but the natural integration of MATLAB into the teaching and learning environment has two real purposes:

1. Firstly, it is an introduction to the tool for generic engineering and scientific design and data analysis.
2. Secondly, it is used to support the learning of semiconductor device operation.

The basic idea is that experiments are undertaken on practical devices, the results obtained are then analysed in MATLAB and finally compared to the ideal device (mathematical)

models. Hence, within MATLAB, actual data is used and mathematical models of ideal devices are developed. This is aimed as an introduction (targeting first year undergraduate students) to both semiconductor devices and to mathematical analysis tools (here MATLAB which would then be used by the students later on in more advanced subjects).

Three different teaching and learning scenarios are presented and the integration of MATLAB into a computer aided learning (CAL) environment that has been custom developed are provided:

Firstly, students undertake experiments in a *traditional learning scenario.* In the laboratory, electronic circuits using semiconductor devices are built and tested. Experimental results are then taken and analysed using MATLAB; specifically, results are entered into arrays (the term *array* used here to mean a *1 x m* matrix) within MATLAB which then allows these results to be analysed and graphically plotted. These results are also compared to the ideal mathematical equations for the devices considered (specifically diodes and transistors). Hence, the learning experience naturally includes an introduction to concepts such as scalar types and arrays (in a generic context, matrices and matrix manipulation), building and manipulating equations, manipulating experiment results, results comparison, graphical plotting and m-files. The student therefore gains experience in both electronic hardware build and test, and results analysis using MATLAB. This is suitable for electronic and computer engineering students at an introductory level. This idea is depicted in figure 1.

Secondly, students undertake experiments using *computer aided learning (CAL) environment*. The experiment electronic hardware is pre-built and connected to a PC via an *experiment interface electronics* unit (essentially a computer port connection such as RS-232 (readily extended to USB) interface that allows for analogue voltages to be created and sampled in the same manner as would manually be done, but now through a software graphical user interface (GUI)). The student therefore gains experience in computer control of experiments and results analysis using MATLAB. This is suitable for electronic and computer engineering students at an introductory level who would not necessarily need to physically build electronic hardware. This idea is depicted in figure 2.

Thirdly, students undertake experiments via a *distance mode of learning* in that they access the experiment electronic hardware and MATLAB via an Internet browser. This arrangement forms a *remote laboratory* whereby the experiment is controlled and results accessed remotely and via an Internet browser. Essentially, the CAL arrangement identified in figure 2 is "web enabled" – that is made accessible via the Internet. The student gains therefore experience in computer control and results analysis using MATLAB, but in a distance mode of learning. This idea is depicted in figure 3.

The above three teaching and learning scenarios provide ways in which MATLAB can be integrated into a flexible teaching and learning environment. However, given that the idea here is that both the use of MATLAB and the electrical characteristics of basic semiconductor devices are to be introduced, the structure of the laboratory experiments must be carefully considered. For example, it would be necessary to introduce the basic concepts of MATLAB

as well as the key commands to be used before attempting to analyse experiment data. One possible laboratory experiment flow is shown in figure 4.

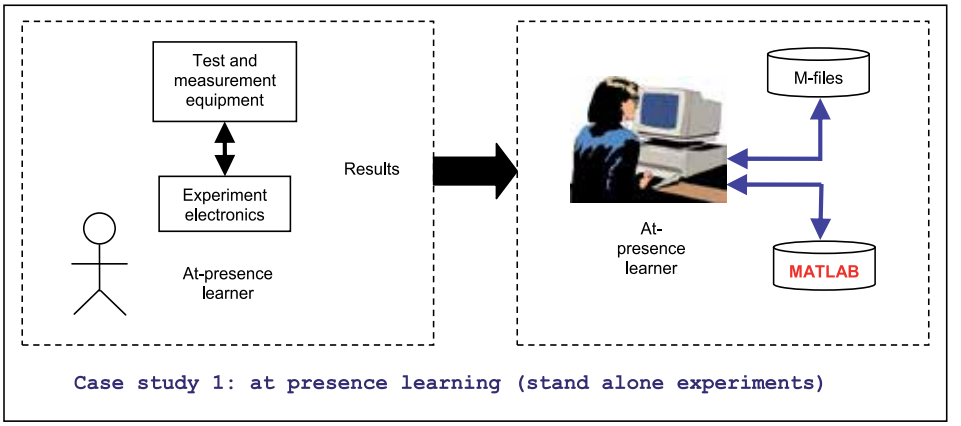


Figure 1. The traditional learning scenario

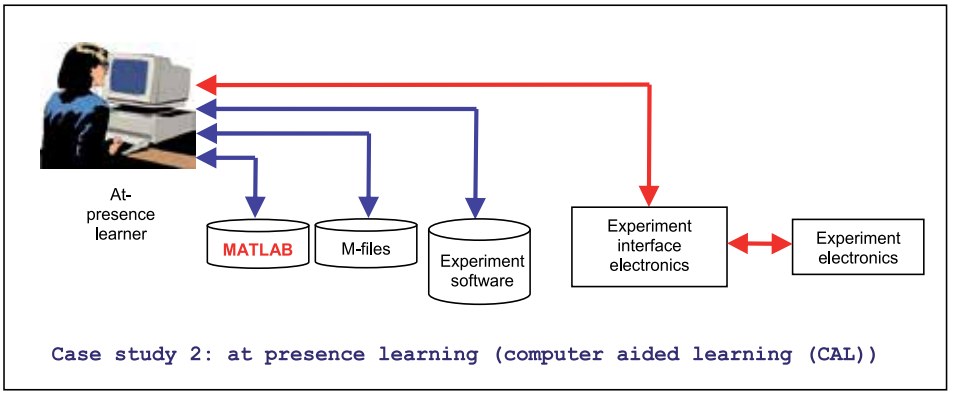


Figure 2. The computer aided learning scenario

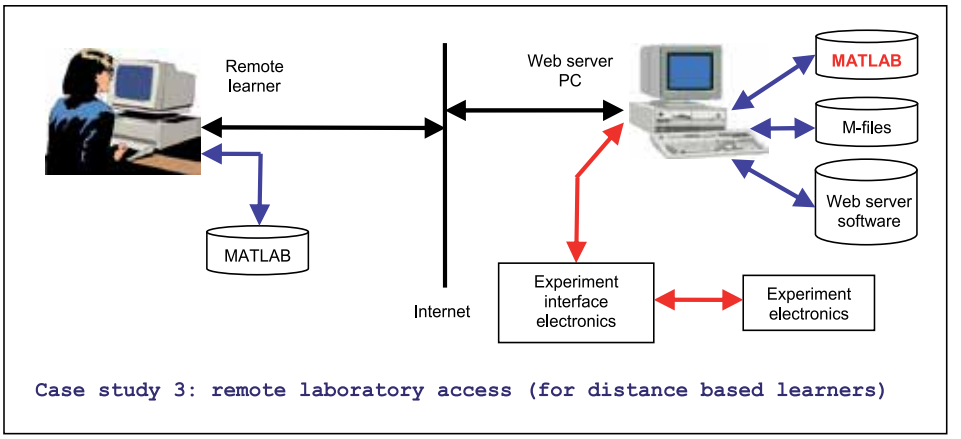


Figure 3. Remote user access via an Internet browser learning scenario

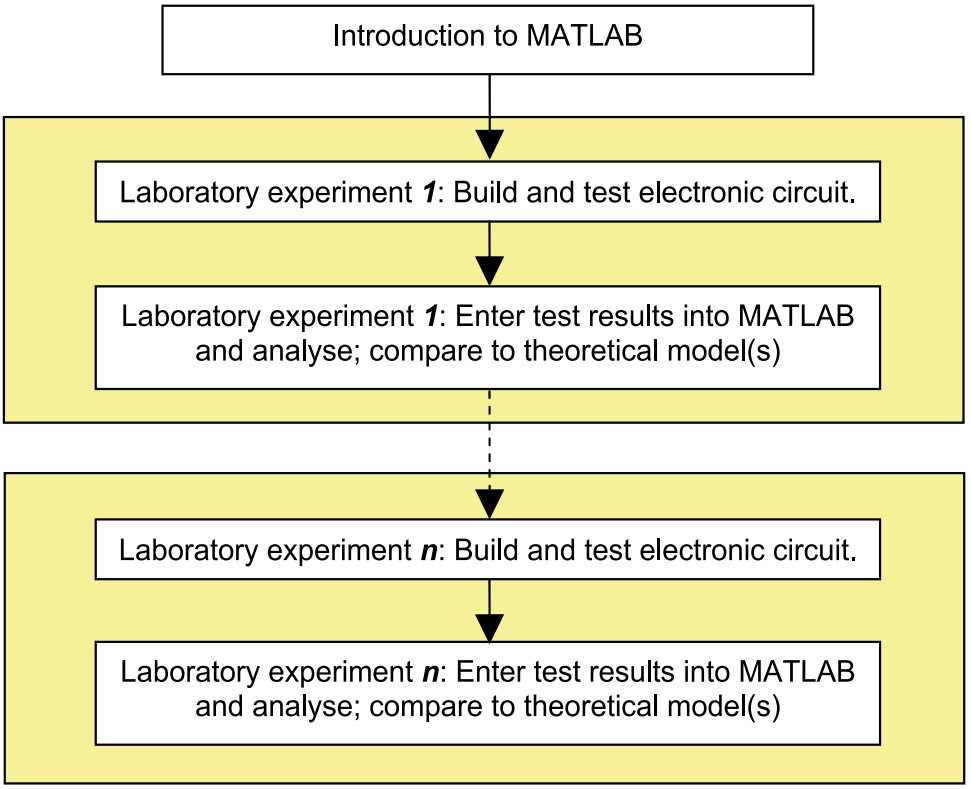


Figure 4. Laboratory experiment "flow"

The development of the hardware-software infrastructure and use of the three above teaching and learning scenarios are introduced here with reference to an experiment consisting of a BAT86 Schottky barrier diode [2] and are presented in this chapter. The remainder of the chapter is structured as follows:

Section 2: The use of MATLAB within an education environment

The use of MATLAB as an aid to teaching and learning for a wide range of engineering and scientific applications is presented. A rationale for using MATLAB is provided and how it may be used is identified. Reference is made to the teaching and learning in the computer and electronic engineering disciplines.

Section 3: Teaching and learning semiconductor device fundamentals

The teaching of semiconductor device fundamentals at an introductory level within the university sector is presented with reference to current teaching undertaken by the authors. The need for teaching semiconductor devices and their application in the electronics and microelectronic industries is provided, along with the need to relate the theory to real (practical) devices through the use of suitable laboratory experiments undertaken by the students. The use of MATLAB as an integrated mathematical analysis tool is presented where theory and practice are compared.

Section 4: Case study 1: at presence learning (stand alone experiments)

The use of MATLAB is presented whereby physical circuits with the semiconductor devices of interest are developed and tested by the student. Results are then entered into MATLAB and analysis undertaken, comparing the real devices with their mathematical ideal models. Current and voltage relationships are then identified. The student gains hands-on experience with both electronic hardware and computer based software.

Section 5: Case study 2: at presence learning (computer aided learning (CAL))

The use of MATLAB is presented as in case study 1, but now pre-built experiments are accessed through a custom software application and the experiments are accessed via a PC interfaced electronic hardware arrangement. The student concentrates on the MATLAB and software side of the experiment activity.

Section 6: Case study 3: remote laboratory access (for distance based learners)

The use of MATLAB is presented as in case study 2, but now the interface is via a remote laboratory arrangement, accessed via an Internet browser and web server arrangement. With this arrangement, remote learners are supported.

Section 7: Conclusions

Conclusions to the work that has been undertaken are presented.

Section 8: References

References used in the development of the chapter are provided.

2. The use of MATLAB within an education environment

2.1. Introduction

MATLAB is an invaluable tool for use within the engineering disciplines for education, research and industrial purposes. In this chapter, the use of MATLAB is presented and discussed within a university education context. MATLAB is an almost universal tool for engineering education. It provides a cost-effective *what if* platform where users can manipulate and explore functions to discover and explore the response of a system. Here, the system is an electronic circuit where the focus of the circuit operation is discovering and exploring the behaviour of semiconductor devices.

2.2. Why use MATLAB?

MATLAB is a high-level language and interactive environment that enables a user to perform computationally intensive engineering and scientific calculations tasks faster than with traditional programming languages such as C. It includes a set of integrated graphics and plotting capabilities allowing users to visualise their data and analysis results and which can also be extended by the user to suit his or her own needs. As such, it provides the student and practising engineer with a suite of useful tools for analysing and solving engineering related problems. For semiconductor devices made from semiconductor

materials such as silicon, which the work discussed in this chapter are aim at exploring, MATLAB is the perfect tool to use as it allows the student to undertake directed and self study activities, even outside laboratory. As the calls for innovation and creativity become stronger, students cannot afford to limit their experiments and exploration within physical laboratory. By integrating MATLAB within experiments and the course syllabus, it supports self-directed learning and also does not cost anything to make mistakes!

2.3. Important concepts to introduce

With the integration of MATLAB into a course syllabus, there is a need to identify the key concepts to introduce and for the students to practice. It is therefore important for the course developer to ensure that there is a seamless integration of MATLAB into the course syllabus and for there to be a clear focus on why and how this mathematical analysis tool is used. Therefore the course developer needs to consider a wide range of aspects including:

1. The role of MATLAB
   Why is MATLAB utilised in the course with a focus on the engineering discipline concerned? How would there be a suitable and seamless integration of the analysis tool with the core engineering topics in the course? How much time should be allocated to the teaching and learning of MATLAB core concepts Vs the use of MATLAB to solve engineering problems?

2. What is important for electronic and computer engineering students
   Why utilise mathematical analysis tools in electronic and computer engineering and how can they be used to support the practicing engineer? With MATLAB being introduced to the students for the first time, how can this support more advanced engineering topics? For example, MATLAB with its toolbox Simulink is widely used in control engineering and where students are introduced to control engineering concepts, their knowledge of MATLAB from this introductory course could be used to allow the teaching of the control engineering to concentrate on using MATLAB rather than reintroducing the core MATLAB concepts.

3. Consider a stand-alone module (i.e., just MATLAB) or integrate MATLAB into subject (as considered here)
   The introduction to students of MATLAB can be either the main focus of a course whereby the introduction to MATLAB is the purpose of the course, or MATLAB can be introduced as a tool to use in supporting engineering disciplines. Whilst allocating a complete course to MATLAB would allow students to consider both the introductory concepts and the more advanced concepts (such as the use of the toolboxes), it might not necessarily provide a link to the use of this tool in solving engineering discipline specific problems. It also means that valuable and restricted time within the overall programme of study (the available time needs to be allocated to many different aspects of engineering) which should be focused on the specific engineering discipline is not necessarily allocated to the focus area of the overall programme of study. The alternative approach, as considered here, is to provide a more generic introduction to

the tool before using it to solve problems relating to a specific electronic and computer engineering discipline, namely semiconductor devices.

4. Navigating the MATLAB desktop
   The MATLAB desktop provides the method in which a user interfaces to MATLAB and hence a working knowledge of the desktop must be obtained. However, learning the structure of the desktop should not become the focus of the learning and so detract from learning how to use the tool to solve real engineering problems.

5. Dealing with errors
   When learning how to use any software application and a new language, errors in the use of the software application, along with syntax and semantic errors with the language will inevitably be experienced by the student. How to deal with these errors can be a daunting task for a student and so the prompt correcting of these errors would be important. It would be expected that there would be a common set of errors encountered by many students and so many errors should be readily identifiable and corrected.

6. Matrix manipulation
   Within MATLAB, everything is treated as a matrix. Hence, the students would need to revise their previous learning of matrices and apply the concepts within the MATLAB environment, learning how to create and manipulate matrices using the native syntax. For example, a common problem encountered when learning how to use MATLAB is in the multiplication of matrices (for example, such as determining the square of an $n \times m$ matrix named y if attempting to use the command y^2 directly and y is not a square matrix).

7. Command line entry Vs m-files
   The starting point of learning the tool is how to effectively use the MATLAB command line for data and command entry. Once the basic concepts are learnt, the use of m-files (both script m-files and function m-files [3]) can then be introduced and from there onwards, m-files may become a more convenient manner in which to enter data and commands.

8. Arithmetic operators
   The use of arithmetic operators (addition, subtraction, multiplication, division, left division, power) when considering scalar values (1×1 matrices) and matrices ($n \times m$ matrices). How the arithmetic operations are undertaken – operations undertaken on the complete matrix or matrix element-by-element in turn. This requires a working knowledge of matrix algebra.

9. Logical operators
   The use of logical operators (less than, less than or equal to, greater than, greater than or equal to, equal to, not equal to, logical AND, logical OR) for determining conditions of the variables within the MATLAB workspace.

10. Functions
    A number of mathematical functions are available to create equations (including absolute value, square root, sine (value in radians), cosine (value in radians), tangent

(value in radians), the exponential operator, the logarithmic operator, the value of pi (Π), the imaginary unit (i or j = √(-1))).

11. Program (flow) control
    Program (flow) control allows for the development of scripts that control how the script (program) operates depending on specific conditions. There are three types of control statement in MATLAB, along with a program termination statement: **conditional control**, **loop control**, **error control** and the program termination statement **return**. **Conditional control** statements are *if* (together with *else* and *elseif*) and *switch* (together with *case* and *otherwise*) to execute MATLAB statements based on some logical condition. Loop control statements are *for* to execute MATLAB statements a fixed number of times, *while* to execute MATLAB statements an indefinite time based on some logical condition and *continue* to pass control to the next iteration of the *for loop* or *while loop* in which it appears and skips any remaining statements in the body of the loop. **Error control** (*try* … *catch*) changes the flow control if an error is detected during execution. The program termination statement **return** causes execution to return to the invoking function.

12. 2D and 3D plotting
    MATLAB includes a large number of plotting functions which allow the user to view their data as both two-dimensional (2D) plots and three-dimensional (3D) plots.

13. MATLAB toolboxes
    Whilst probably not appropriate to include in an introductory course, the MATLAB toolboxes such as Simulink and DSP toolbox provide for powerful extensions to the basic MATLAB commands and would typically be used in more advanced courses. For example, Simulink is widely used in the control engineering discipline and, with its block diagram graphical model generation approach, provides for a useful and important tool for the engineer.

14. Integration of other languages
    Whilst probably not appropriate to include in an introductory course, the ability to create MATLAB scripts which interact with code developed in other languages (such as FORTAN, C/C++ and Java) provide for a useful extension and enhanced flexibility in the use of MATLAB for system modelling and analysis.

15. Graphical user interface (GUI) development
    MATLAB provides for the ability to create graphical user interfaces which allow the user to interact with MATLAB through a suitable user interface rather than the command prompt or directly writing m-files.

16. Dealing with terminology
    Finally, as with anything that the engineer is involved in, there is a need to learn and correctly use the terminology specific to the domain that is being worked in.

3. Teaching and learning of semiconductor device fundamentals

For electronic and computer engineers, the use of semiconductor devices is integral to everything that they do, whether they design electronic circuits using semiconductor devices or program processors to control specific electronic circuits. Whilst practising engineers may not necessarily investigate the physics of the devices on a day-to-day basis, it is essential that they understand the behaviour of the devices at the material level (how they behave and why) in order to use these devices effectively within the electronic circuits they design or use. There are therefore two key aspects to the teaching and learning of semiconductor device fundamentals:

1. The **theory underpinning the device operation** – what is happening at the physical material level and how the behaviour can be related to device terminal behaviour in terms of voltages and currents (the development of theoretical mathematical models for idealised and more realistic device models).
2. How the **device terminal behaviour in terms of voltages and currents** (using the developed idealised and more realistic device models) can be used in practical and useful electronic circuits.

For example, consider an introduction to diodes. Both semiconductor (p-n junction) and Schottky barrier (metal-semiconductor contact) diodes are encountered and so must be introduced. The concepts identified in theses devices (such as a.c. signal rectification) would then be extended to more complex devices such as transistors, thyristors, triacs and integrated circuits (ICs).

Consider the Schottky barrier diode. The initial starting point is the structure and underlying mathematical equations (current-voltage (I-V) relationship) for this device. These are summarised in figure 5.

The starting point for understanding the device operation would need to introduce semiconductor materials, what their properties are and how they can be considered to behave (electrons and holes as charge carriers, intrinsic and extrinsic semiconductors, and the effects material doping [4, 5]). Based on these principles then the behaviour of the metal-semiconductor contact can be introduced and developed:

1. The behaviour of the materials around the contact junction and away from the contact junction at thermal equilibrium and non-equilibrium conditions (forward bias and reverse bias conditions).
2. The differences between ohmic and rectifying contacts. It is the rectifying contact (allowing current to flow through the metal-semiconductor contact under forward bias conditions but blocking current flow under reverse bias conditions) that would be of interest in the Schottkty barrier diode discussions.
3. The current-voltage (I-V) relationship at forward and reverse bias conditions at the anode and cathode device terminals would be developed and the use of the diode in

electronic circuits would be introduced and discussed. Reference would initially be made to a rectifier circuit using the diode and a resistor connected in series as shown in figure 6(a) [note that the standard diode symbol is shown here rather than the Schottky barrier diode symbol].

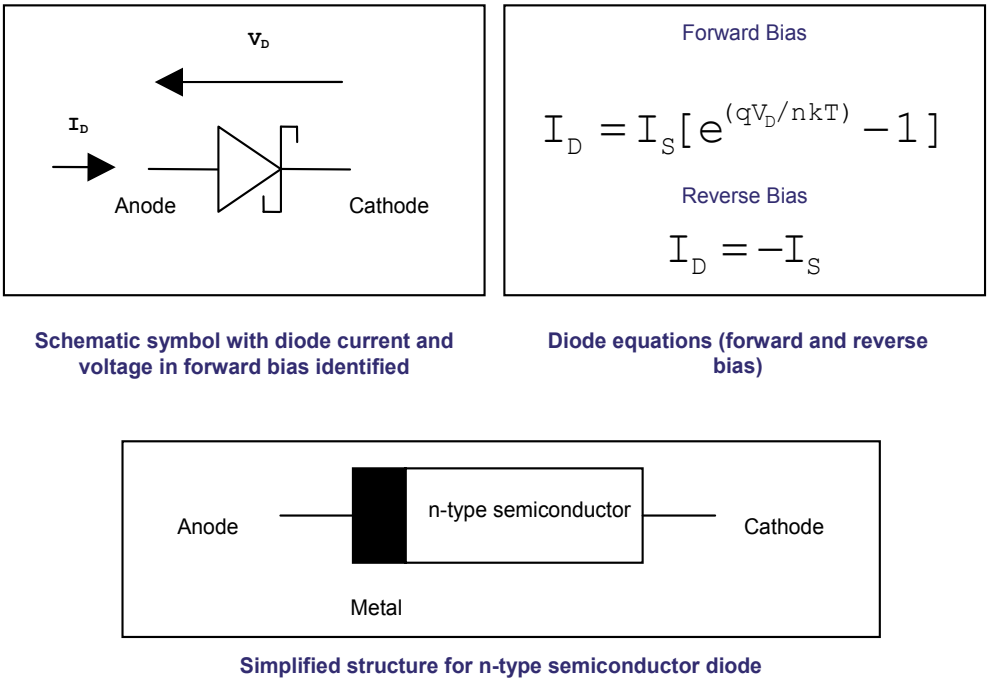


Figure 5. Schottky barrier diode summary (n-type semiconductor type)

With reference to the diode rectifier circuit (figure 6(a)), this can be modelled mathematically for both forward bias and reverse bias in MATLAB to show the principle of operation. A sample m-file for modelling and simulating the Schottky barrier diode is shown in listing 1 and the output plot is shown in figure 7. Note that this code works for versions of MATLAB after v6.5. Here:

1. An idealised diode model is created which has a forward voltage drop of 0.3 V (*Vd*) when conducting (line 14).
2. The resistor (*R*) is set to 2 Ω for illustration purposes (line 15).
3. A sine wave voltage source (*Vin*) is created with an amplitude of 1 V and ten complete cycles of the sine wave are generated (lines 17 to 22).
4. The resistor voltage drop in forward bias and reverse bias is calculated (lines 24 to 30).
5. The resistor current and hence the diode current (*Id*) is calculated (line 32).
6. Four sub-plots are created with time along the x-axis (lines 38 to 73). There is then a delay of five seconds before the script code continues (line 74).
7. The plotted values are scrolled across the sub-plots (lines 80 to 90).

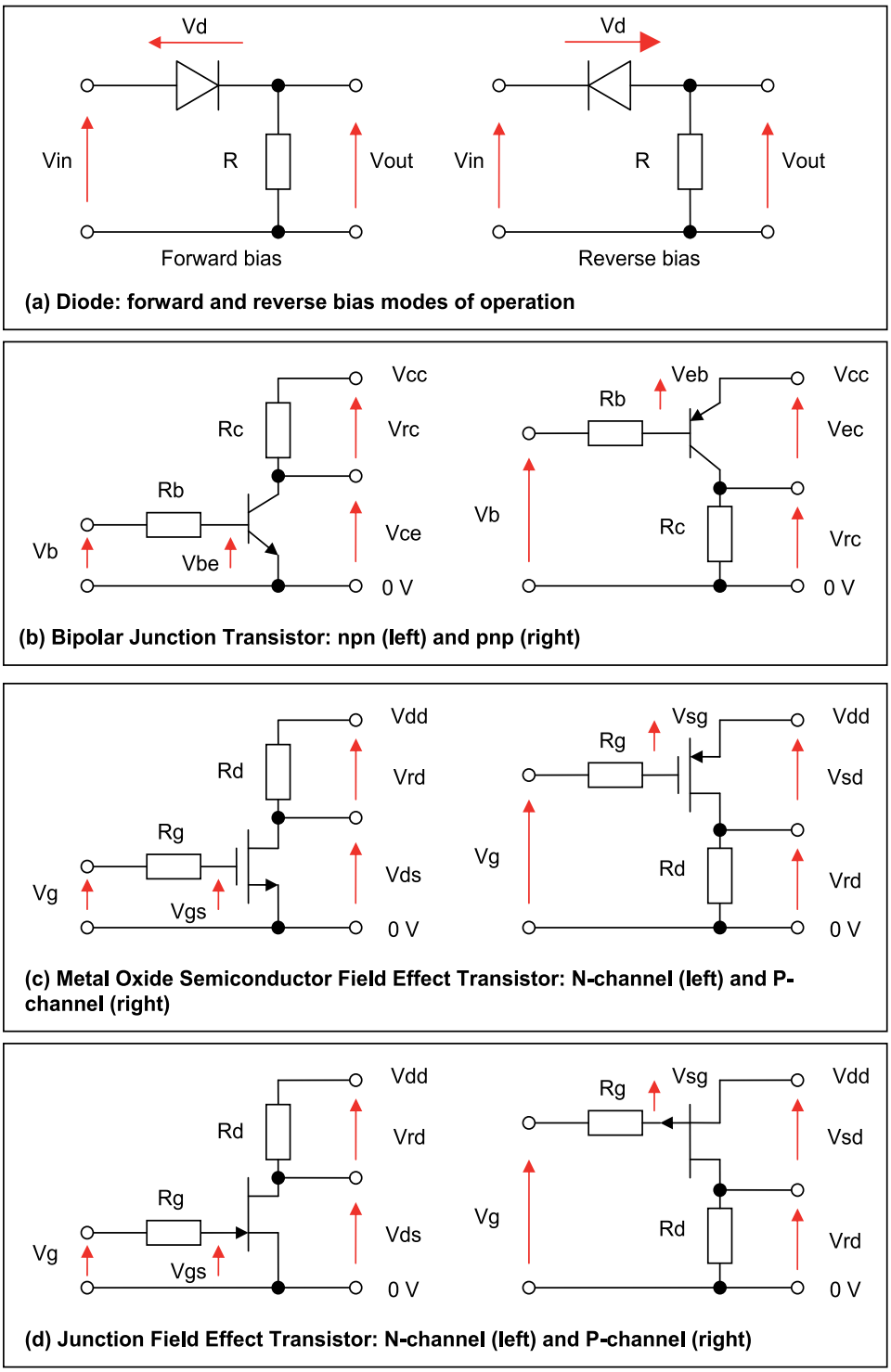


Figure 6. Basic test circuits for diodes and transistors

```
%%-------------------------------------------------
%% Function to create the waveforms and subplots
%% for a Schottky diode model.
%%-------------------------------------------------

function Schottky_animation()

%%-------------------------------------------------
%% Create the variables and equations
%%-------------------------------------------------

clear Degrees Time R Vin Vr Vd Id;

Vd = 0.3;
R = 2;

for (j=0:1:9)
    for (k=1:1:360)
        Vin(k + (j* 360)) = 1.0 * sind(k - 1);
        Time(k + (j* 360)) = (k + (j * 360)) / 360;
    end
end

for (i=1:1:length(Vin))
    if (Vin(i) > 0.3)
        Vr(i) = (Vin(i) - Vd);
    else
        Vr(i) = 0;
    end
end

Id = (Vr / R);

%%-------------------------------------------------
%% Create the subplots and wait for 5 seconds
%%-------------------------------------------------

scrsz = get(0,'ScreenSize');
figure('Name', 'Ideal Schottky Diode Half-Wave Rectifier Circuit', ...
   'OuterPosition',[scrsz(3)/8 scrsz(4)/16 (3 * (scrsz(3)/4)) (15 * (scrsz(4)/16))])

ha(1) = subplot(4,1,1);
plot(Time, Vin,'YDataSource', 'Vin');
hold on;
grid;
plot(Time, Vr, 'r', 'YDataSource', 'Vr');
title ('\it{Time Vs Vin & Vr}', 'Color','k', 'FontWeight', 'bold');
xlabel('Time (seconds)');
ylabel('Vin & Vr (V)');

ha(2) = subplot(4,1,2);
plot(Time, Vin,'YDataSource','Vin');
grid;
```

```
title ('\it{Time Vs Vin}', 'Color','b', 'FontWeight', 'bold');
xlabel('Time (seconds)');
ylabel('Vin (V)');

ha(3) = subplot(4,1,3);
plot(Time, Vr, 'r', 'YDataSource', 'Vr');
grid;
title ('\it{Time Vs Vr}', 'Color','r', 'FontWeight', 'bold');
xlabel('Time (seconds)');
ylabel('Vr (V)');

ha(4) = subplot(4,1,4);
plot(Time, Id, 'm', 'YDataSource', 'Id');
grid;
title ('\it{Time Vs Id}', 'Color','m', 'FontWeight', 'bold');
xlabel('Time (seconds)');
ylabel('Id (A)');

linkaxes(ha, 'xy');
axis([0 3 -1.1 1.1]);
pause(5);

%%-------------------------------------------------
%% Create the scrolling plot
%%-------------------------------------------------

for k = 0:0.1:7

    refreshdata(ha(1),'caller');
    refreshdata(ha(2),'caller');
    refreshdata(ha(3),'caller');
    refreshdata(ha(4),'caller');
    axis([k (k + 3) -1.1 1.1]);
    drawnow;
    pause(0.5);

end

end

%%-------------------------------------------------
%% End of code
%%-------------------------------------------------
```

Listing 1. Schottky barrier diode model – MATLAB m-file code to show signal rectification

In figure 7 then:

1. Time is plotted on the horizontal axis.
2. The top plot shows the input sine wave voltage (*Vin*) and the half-wave rectified voltage across the resistor (*Vr*) are plotted on the vertical axis.
3. The second plot shows the input voltage (*Vin*).
4. The third plot shows the resistor voltage (*Vr*).
5. The bottom plot shows the diode current (*Id*).

When the plot scrolls, after an initial delay of five seconds, it shows a similar waveform to that which would be seen on an oscilloscope display; it would be seen here that as time increments, the viewed waveforms scrolls across the screen. This shows how theoretical models can be created and analysed in MATLAB. Simulation however is only part of the overall story. Relating theory to the "real world" requires suitable the *design*, *build* and *test* of real circuits. Then, MATLAB could be used to analyse the results from a physical circuit prototype, and the theoretical model and practical circuits could be compared.

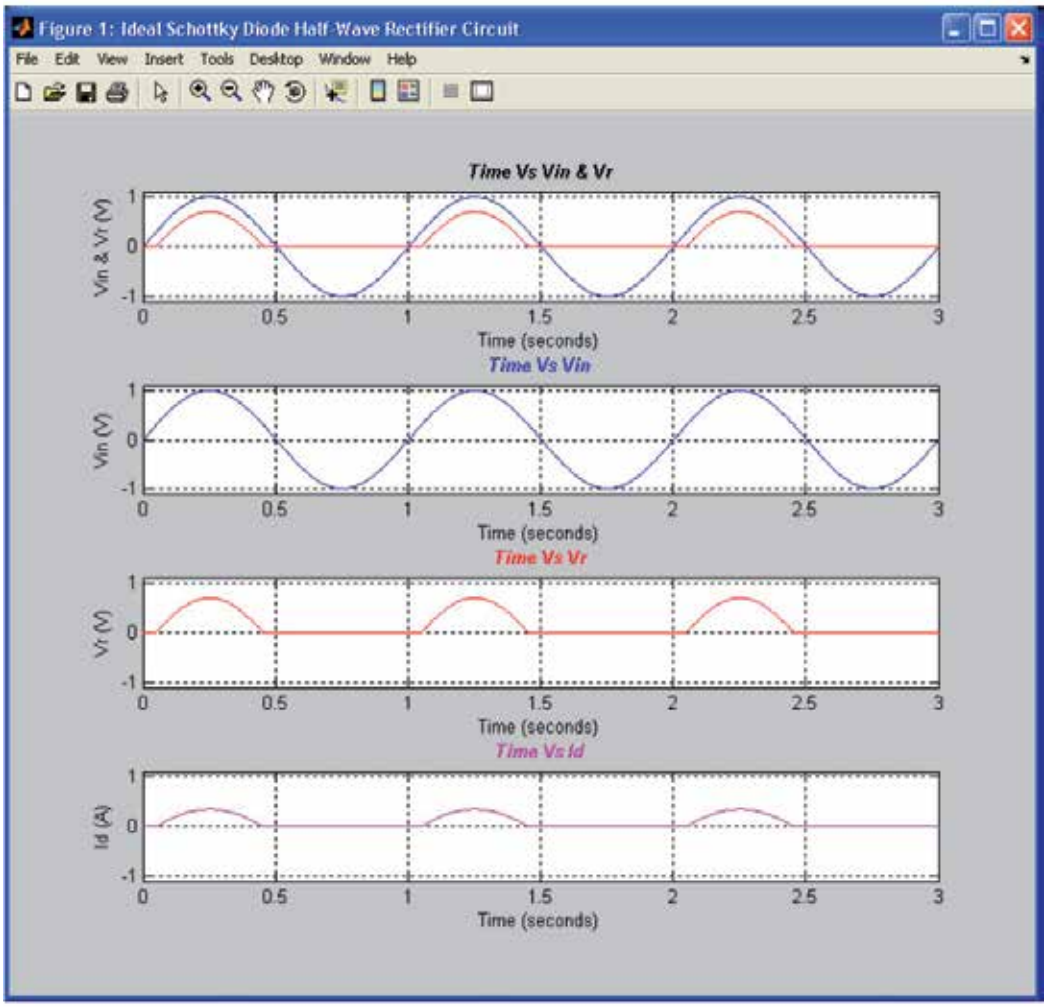


Figure 7. Schottky barrier diode model – MATLAB plot from listing 1

In more general terms, the above discussion flow would be extended to any semiconductor device, given the ability to create and measure the required range of voltage levels. For example, figure 6(b) shows a test circuit for a bipolar junction transistor (both npn and pnp types), figure 6(c) shows a test circuit for a metal oxide semiconductor field effect transistor

(both N-channel an P-channel types) and finally, figure 6(d) shows a test circuit for a junction field effect transistor (both N-channel an P-channel types). These test circuits are suitable for the experimentation arrangements considered here. However, it is possible to create different test set-ups given the availability of suitable test and measurement equipment. Care however has to be taken in order to ensure that:

1. Suitable device types are used.
2. The applied voltages operate the devices in the intended modes of operation.
3. Damage to the devices would not occur in the physical test set-up if specified and used correctly (maximum device ratings are never encountered or exceeded).

Device damage would not occur in a simulated mathematical model, but could occur in a real device. For example, in N-channel JFET circuits, the gate-source voltage is to be zero or negative for correct device operation with the drain-source voltage zero or positive. The gate resistor (Rg) is included here to ensure that if a positive gate-source voltage were to be applied then the current through the JFET gate node would be limited to a safe value by suitable choice of the resistor value. When gate-source voltage is to be zero or negative, no current flows through the transistor gate (the p-n junction created is reverse biased) and for d.c. gate-source voltages, the resistor has no effect (although for a.c. signals the value of the resistor would affect the circuit operation).

4. Case study 1: The traditional learning scenario

This section will describe the use of the experiment via a traditional laboratory scenario. In this arrangement, the student builds a test circuit (either using a suitable solderless prototyping board or physically soldering the components to a suitable printed circuit board (PCB)) and runs a number of electrical tests on the electronic circuit. The tests are chosen to operate the particular device under the modes of operation that are of interest for the student to investigate. Once the tests have been completed, the student would plot the results (by-hand) on graph paper and then import the results into MATLAB for analysis and computer based graphing of the results. Here, it would then be possible to consider the physical process of setting-up an experiment, running the experiment and taking results before utilising a suitable mathematics tool for analysis purposes. However, the traditional *manual* way of plotting the graph from experiment data is slow and sometimes not convenient. Both the idealised (theoretical mathematical equation model) and the operation of an actual semiconductor device could then be analysed and compared, with differences between the practical device test results and idealised models analysed using MATLAB. Using the analysis tool to analyse the behaviour of and to plot the experiment data means that various analyses can be performed on the data and the results quickly plotted.

Within the teaching and learning of semiconductor device fundamentals, the basic devices to initially introduce to the student are the diode and transistor. To illustrate this, figure 8 shows the device to discuss here, the Schottky barrier diode. The circuit here shows the BAT86 Schottky barrier diode in a forward bias mode of operation. In this mode of operation, when the input voltage applied is positive, the diode will allow the flow of

current through the load resistor and the diode will have a voltage drop of approximately 0.3 V when conducting (the actual device voltage drop being dependent on the level of current flowing through the diode). If the diode is connected in the reverse direction, the reverse bias mode of operation will be encountered and the diode will block the flow of current until a reverse bias junction breakdown voltage is encountered at which point the diode will conduct current. In reverse bias junction breakdown, if the current flow is not limited then damage to the diode will occur.

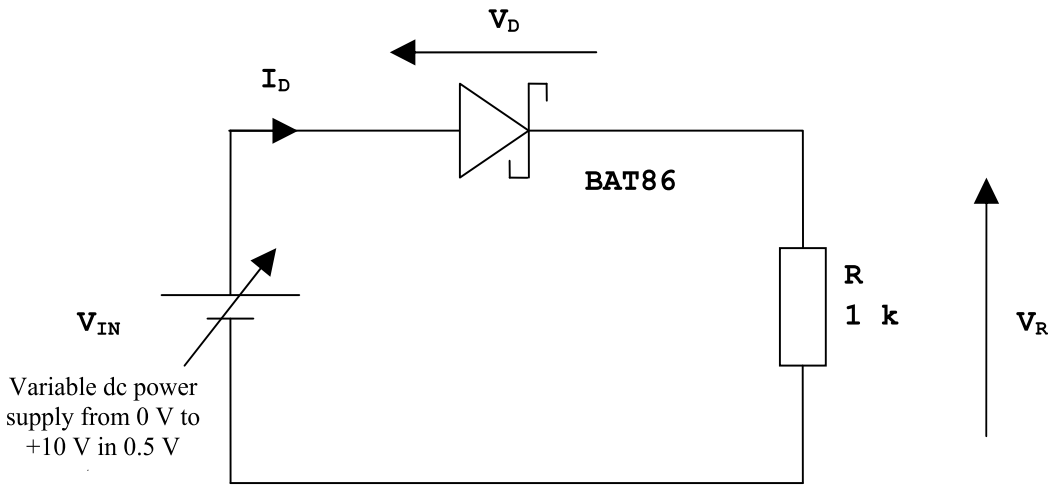


Figure 8. BAT86 Schottky diode experiment (forward bias)

The I-V mathematical model characteristic of the diode in figure 9 shows both the expected forward and reverse bias modes of operation and the ideal device equation are also noted:

Forward bias:

$$I_D = I_S\left[e^{(qV_D/nkT)} - 1\right]$$

Reverse bias (prior to breakdown):

$$I_D = -I_S$$

Here:

- I_D is the diode current.
- I_S is the diode saturation current.
- q is the charge on an electron.
- V_D is the forward bias diode voltage drop.
- n is the ideality factor and is set to 1.
- k is Botzmann's constant.
- T is the temperature in degrees Kelvin.

The current-voltage (I-V) relationship that should be encountered during an experiment is that as shown in figure 9. The regions of operation of interest are the forward bias (to the right of the I_D-axis) and the reverse bias (to the left of the I_D-axis) prior to reverse bias junction breakdown.

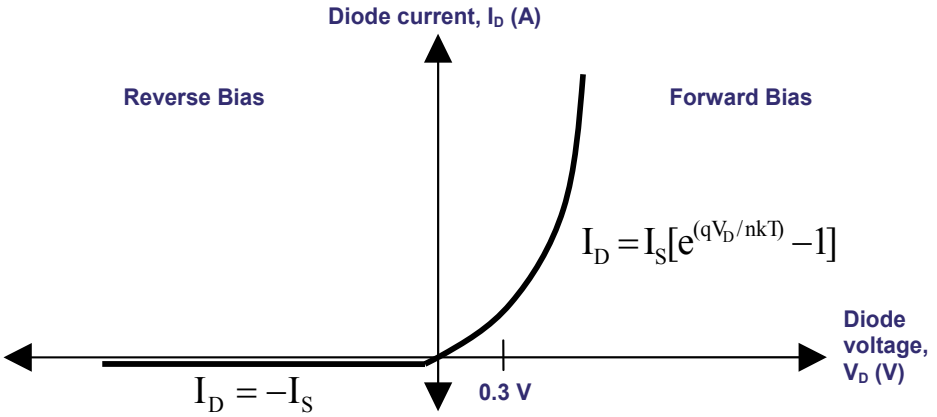


Figure 9. Schottky diode I-V characteristic (before reverse bias junction breakdown is encountered)

In forward bias, the diode current increases in an exponential manner with a linear increase in diode voltage. The diode voltage is around 0.3 V when current flows through the device, the exact value of diode voltage dependent on the value of the diode current. In reverse bias and prior to reverse bias breakdown occurring, the diode current is essentially independent of the diode voltage and is approximately the value of the saturation current (I_S). This effect can readily be modelled in MATLAB as shown in listing 2, here using the *for loop* in the calculation of the diode current for set values of diode voltage.

For comparison purposes, from the BAT86 Schottky barrier diode datasheet, the diode parameters can be identified. These are summarised in table 1.

Parameter	Conditions	Maximum value
Forward bias		
Forward bias voltage drop	Forward current = 0.1 mA	300 mV
	Forward current = 1 mA	380 mV
	Forward current = 10 mA	450 mV
	Forward current = 30 mA	600 mV
	Forward current = 100 mA	900 mV
Reverse bias		
Reverse bias current	Reverse bias voltage = 40 V	5 μA

Table 1. BAT86 Schottky barrier diode [2] datasheet forward and reverse bias parameters

In listing 2, both the forward bias mode of operation and the reverse bias mode of operation are modelled and plotted, where:

- The forward bias voltage is *Vd_forward*.
- The forward bias current is *Id_forward*.
- The reverse bias voltage is *Vd_reverse*.
- The reverse bias current is *Id_ reverse*.

One way in which the ideal device equations can be modelled in MATLAB is shown in listing 2:

```
%%-------------------------------------------------------------------
%%-- Schottky barrier diode forward bias equation
%%-- x-axis scaling (voltage) from 0 V to +0.8 V
%%-------------------------------------------------------------------

        Vd_forward = (0:0.01:0.8)

        T = 300
        k = 1.38066e-23
        q = 1.60218e-19

        Is_sch = 1e-9
        n = 1.0

        for i=(1:1:length(Vd_forward))

            Id_forward = Is_sch * (exp((q * Vd_forward)/(n * k * T)) - 1)

        end

%%-------------------------------------------------------------------
%%-- Schottky barrier diode reverse bias equation
%%-- x-axis scaling (voltage) from 0 V to -10.0 V
%%-------------------------------------------------------------------

        Vd_reverse = (0:-0.01:-10.0)

        Is_sch = 1e-9

        for i = (1:1:length(Vd_reverse))

            Id_reverse = -Is_sch

        end

%%-------------------------------------------------------------------
%% End of code
%%-------------------------------------------------------------------
```

Listing 2. Calculating the forward and reverse bias operation of the Schottky diode

These values and equations can be entered into MATLAB and plots of the I-V characteristic can be produced (by adding code for plotting the results to the code shown in listing 2).

Figure 10 shows a figure where the forward and reverse bias diode characteristics are plotted. The top subplot shows the forward and reverse bias characteristic (V_D shown from -10 V to +0.6 V) and the bottom two subplots show the forward bias characteristic zooming in on different ranges of V_D. Hence, specific areas of device operation can easily be identified from the overall set of data and individual plots created for understanding and analysis purposes.

The experiment can then be prototyped on a solderless prototyping board as shown in figure 11. Connecting this circuit to a dual d.c. power supply and digital voltmeter will provide all the necessary circuitry and equipment to undertake the experiment shown in figure 8. Table 2 shows the forward bias results (measuring *Vin* and *Vr*, and calculating *Vd* and *Id*). Table 3 shows the reverse bias results. Both sets of diode currents are calculated using the calculated resistor current and the actual measured resistor value ($Rm = 997\ \Omega$)

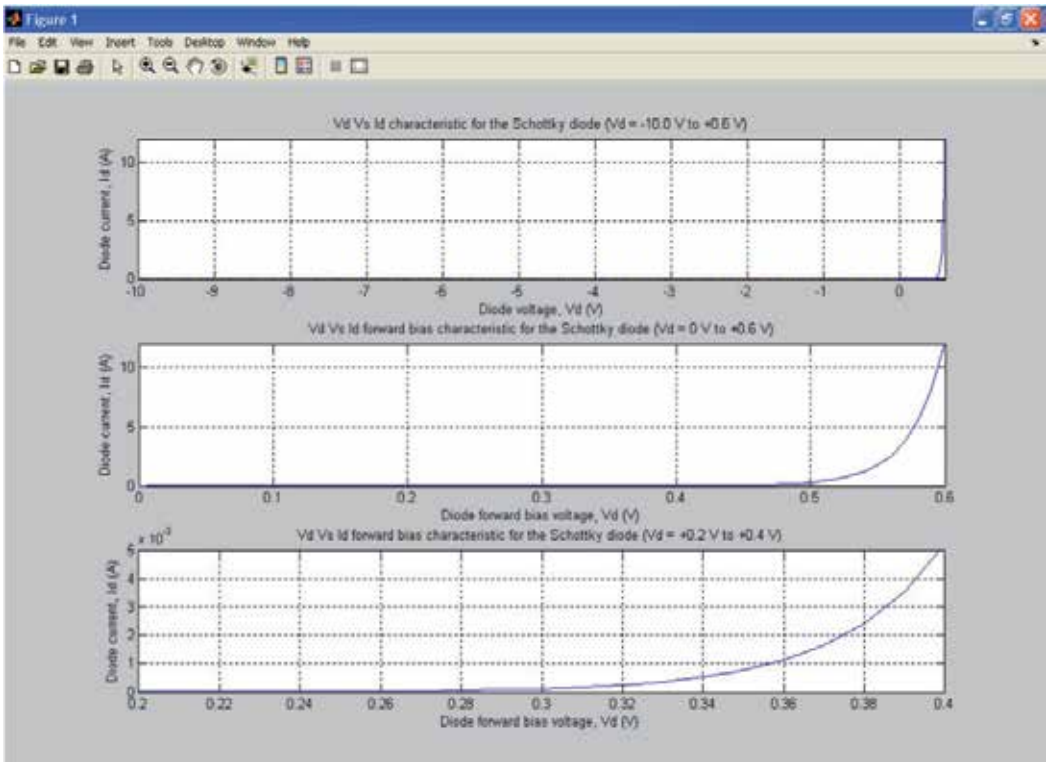


Figure 10. Ideal Schottky barrier diode equation plots

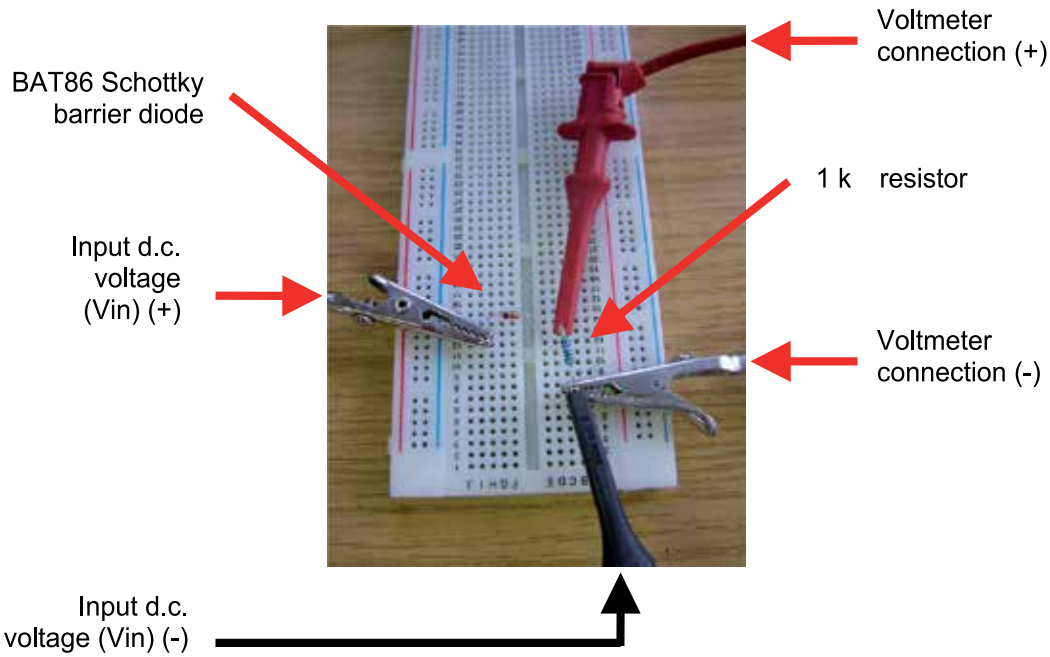


Figure 11. Prototyping the experiment

The measured values of voltage (*Vin* and *Vr*) can be entered into MATLAB and the diode voltage and current (*Vd* and *Id*) can then be calculated. The results can be entered into MATLAB and plotted with the MATLAB m-file code as shown in listing 3. Note also that the actual resistance value (*Rm*) of the 1 kΩ resistor was used in the current calculation in order to account for the tolerance of the resistor used. Figure 12 shows the resulting MATLAB plot with the forward bias shown on the top sub-plot and the both forward and reverse bias shown on the bottom sub-plot.

Vin (V)	Vr (V)	Vd = (Vin – Vr) (V)	Id = Ir = (Vr / Rm) (A)
Measured	**Measured** (to nearest 1 mV)	**Calculated** (to nearest 1 mV)	**Calculated** (to nearest 1 μA)
0	0	0	0
0.5	0.268	0.232	269 μA
1.0	0.739	0.261	741 μA
1.5	1.223	0.277	1.227 mA
2.0	1.712	0.288	1.717 mA
2.5	2.204	0.296	2.211 mA
3.0	2.697	0.303	2.705 mA
3.5	3.190	0.310	3.200 mA
4.0	3.685	0.315	3.696 mA
4.5	4.180	0.320	4.193 mA
5.0	4.675	0.325	4.689 mA
5.5	5.170	0.330	5.186 mA
6.0	5.666	0.334	5.683 mA
6.5	6.162	0.338	6.181 mA
7.0	6.659	0.341	6.679 mA
7.5	7.155	0.345	7.177 mA
8.0	7.651	0.349	7.674 mA
8.5	8.148	0.352	8.173 mA
9.0	8.645	0.355	8.671 mA
9.5	9.141	0.349	9.169 mA
10.0	9.637	0.363	9.666 mA

Table 2. Diode forward bias test results

Vin (V)	Vr (V)	Vd = -(Vin – Vr) (V)	Id = -Ir = -(Vr / Rm) (A)
Measured	**Measured** (to nearest 1 mV)	**Calculated** (to nearest 1 mV)	**Calculated** (to nearest 1 μA)
0	0	-0	0
0.5	0	-0.5	0
1.0	0	-1.0	0
1.5	0	-1.5	0
2.0	0	-2.0	0
2.5	0	-2.5	0
3.0	0	-3.0	0
3.5	0	-3.5	0
4.0	0	-4.0	0
4.5	0	-4.5	0
5.0	0	-5.0	0
5.5	0	-5.5	0
6.0	0	-6.0	0
6.5	0	-6.5	0
7.0	0	-7.0	0
7.5	0	-7.5	0
8.0	0	-8.0	0
8.5	0	-8.5	0
9.0	0	-9.0	0
9.5	0	-9.5	0
10.0	0	-10.0	0

Table 3. Diode reverse bias test results

```
%%-----------------------------------------------------------------------
%%-- BAT86 Schottky barrier diode test
%%-----------------------------------------------------------------------

%%-------------------------------------
%% Test conditions
%%-------------------------------------

Rm = 997
```

```
Vin = [0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10]

%%-------------------------------------
%% Diode in forward bias
%%-------------------------------------

Vr_forward = [0 0.268 0.739 1.223 1.712 2.204 2.697 3.190 3.685 4.180 ...
    4.675 5.170 5.666 6.162 6.659 7.155 7.651 8.148 8.645 9.141 9.637]

Vd_forward = (Vin - Vr_forward)
Id_forward = (Vr_forward / Rm)

%%-------------------------------------
%% Diode in reverse bias
%%-------------------------------------

Vr_reverse = [0 0 0 0 0 0 0 0 0 0 0 0 ...
     0 0 0 0 0 0 0 0 0]

Vd_reverse = -(Vin - Vr_reverse)
Id_reverse = -(Vr_reverse / Rm)

%%-------------------------------------
%% Plot results
%%-------------------------------------

subplot(2, 1, 1)
plot(Vd_forward, Id_forward, 'k')
hold on
grid
plot(Vd_forward, Id_forward, 'o')

title('Vd Vs Id for measured BAT86 Schottky Barrier Diode (forward bias)')
xlabel('Vd (V)')
ylabel('Id (A)')

subplot(2, 1, 2)
plot(Vd_forward, Id_forward, 'k')
hold on
grid
plot(Vd_forward, Id_forward, 'o')

plot(Vd_reverse, Id_reverse, 'r')
plot(Vd_reverse, Id_reverse, 'o')

title('Vd Vs Id for measured BAT86 Schottky Barrier Diode')
xlabel('Vd (V)')
ylabel('Id (A)')

%%-----------------------------------------------------------------------
%%-- End of BAT86 Schottky barrier diode test
%%-----------------------------------------------------------------------
```

Listing 3. M-file code for entering and plotting the test results for the BAT86 Schottky barrier diode

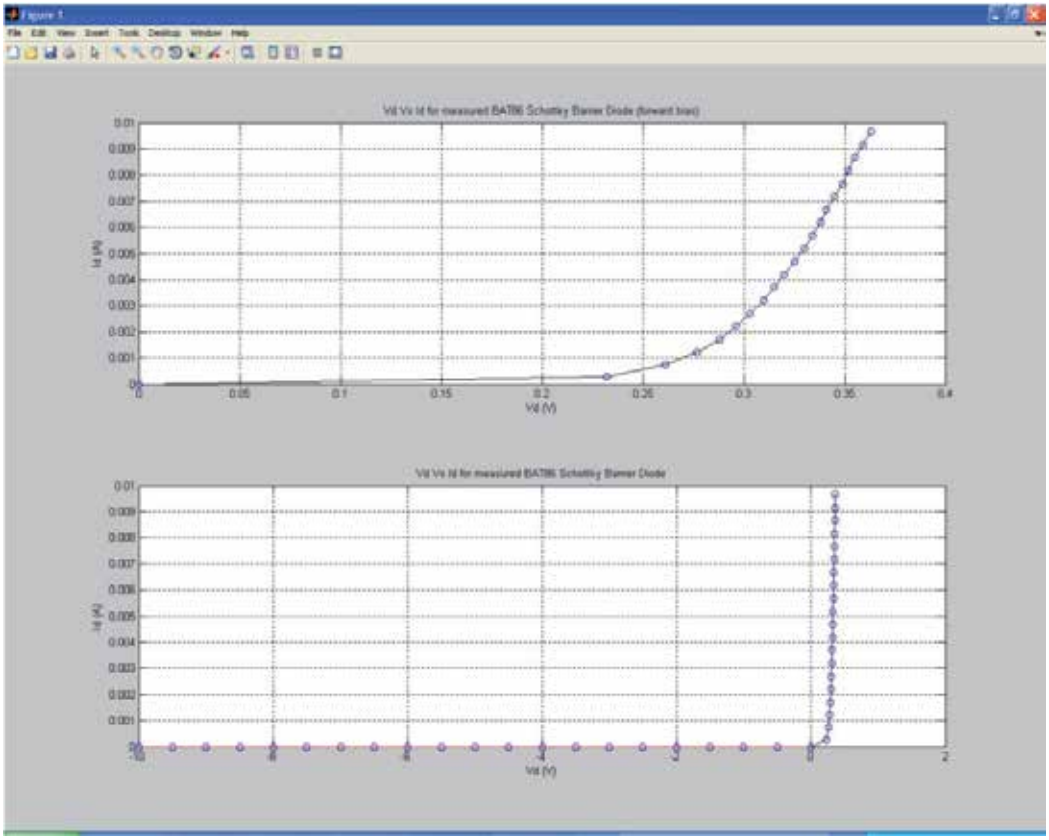


Figure 12. BAT86 Schottky barrier diode test results (forward and reverse bias)

In order to create the plots then there are three main parts to the m-file:

1. Entering the device test data (measured values) as scalar values and arrays:
   a. The measured resistor value (*Rm*) (line 9).
   b. The input voltage values (*Vin*) (line 11).
   c. The measured resistor voltage with the diode in forward bias (*Vr_forward*) (lines 17 and 18).
   d. The measured resistor voltage with the diode in reverse bias (*Vr_reverse*) (lines 27 and 28).
2. Creating the equations to determine the diode voltage and current values:
   a. The diode voltage in forward bias (*Vd_forward*) (line 20).
   b. The diode current in forward bias (*Id_forward*) (line 21).
   c. The diode voltage in reverse bias (*Vd_reverse*) (line 30).
   d. The diode current in reverse bias (*Id_reverse*) (line 31).
3. Plotting the currents and voltages on a single figure using subplots (lines 37 to 58).

5. Case study 2: Computer aided learning scenario

This section will describe the use of the experiment via a computer interface. In this arrangement, the student does not build the circuit – the circuit experiment is pre-built and connected to a computer via a suitable computer serial port. In this discussion, the RS-232 serial port is used and this interfaces to the diode experiment via an interface circuit consisting of a suitably configured Spartan-3 field programmable gate array (FPGA) [6, 7] and digital-to-analogue converter (DAC) and analogue-to-digital converter (ADC) arrangement.

This structure of the hardware and software interface is shown in figure 13.

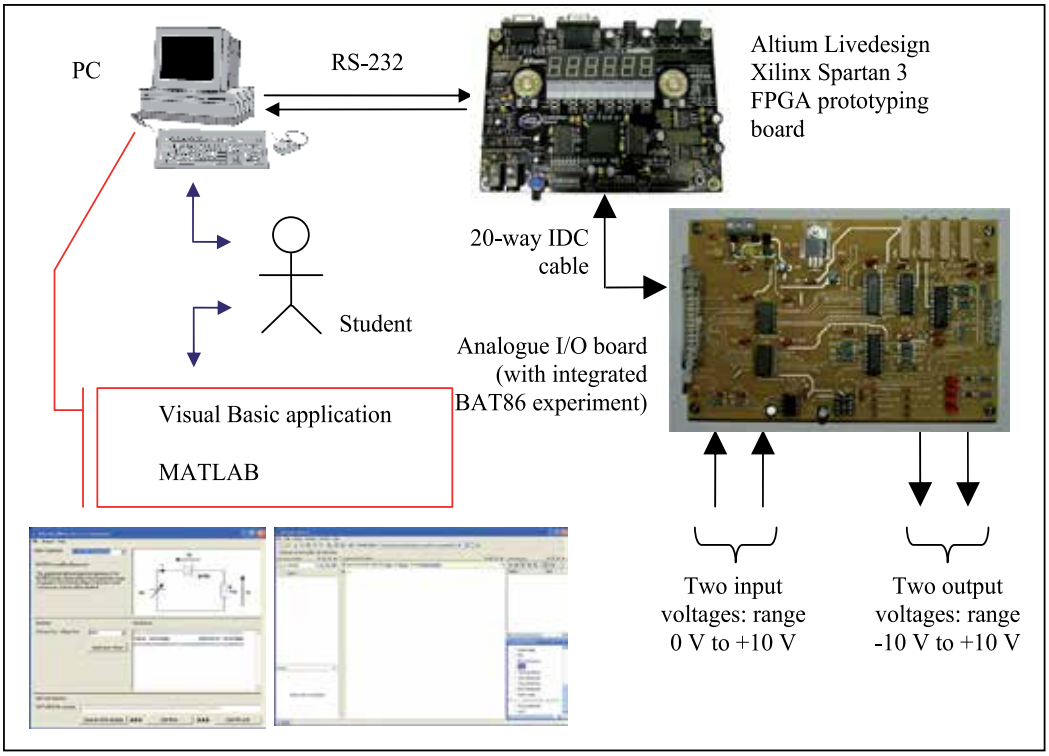


Figure 13. Tester hardware interface set-up

Here, the PC connects to an FPGA prototyping board via an RS-232 interface. The FPGA prototyping board also houses a voltage level shifter circuit (the MAX3232 IC) to interface the RS-232 voltage levels to the FPGA +3.3 V power supply voltage levels. Digital inputs and outputs (I/O) of the FPGA then connect to an analogue I/O board. This houses a power supply, two DACs (providing two independent voltage outputs in the range -10 V to +10 V) and two ADCs (receiving two input voltages in the range 0 V to +10 V). The

analogue I/O board consists of a custom made printed circuit board (PCB) with an on-board BAT86 Schottky barrier diode experiment and a connector to interface to additional experiments.

Figure 14 shows the manufactured PCB in more detail. It is of course possible to utilise an existing hardware arrangement rather than designing a custom made solution.

In addition, the computer runs a suitably designed software application which gives the user access to the tester hardware. It is possible to use any suitable programming language (including the MATLAB GUI (Graphical User Interface) builder) to provide for a user interface and suitable software applications to access to the computer RS-232 port. However, here, a Visual Basic [8] application was used with Visual Basic here being the language of preference. Figure 15 shows the GUI as designed. The user selects the experiment (diode in forward bias or reverse bias mode of operation) and then runs a test by setting the input voltage (*Vin*) to apply to the circuit and then sending this to the experiment. The results from the experiment then are displayed on the GUI.

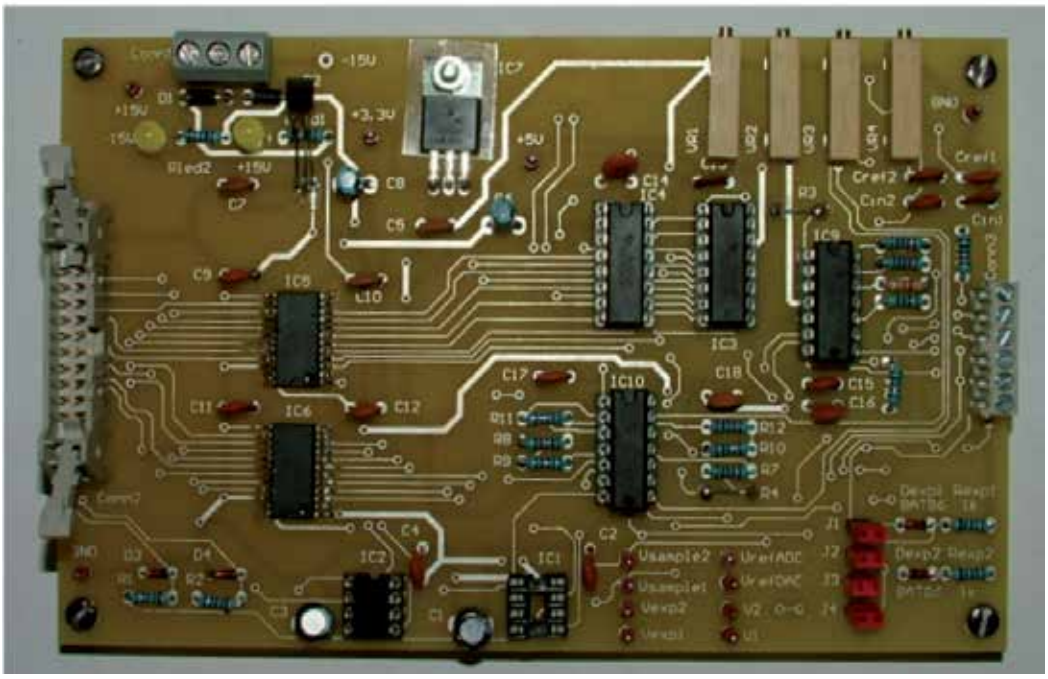


Figure 14. Analogue I/O board PCB

On completion of the experiment, the experiment results are saved into an automatically generated MATLAB m-file template (essentially the input and output voltages are saved in arrays in the m-file). The user then edits the m-file (adding his or her own results analysis and plotting commands), and runs MATLAB to analyse the test results. Listing 4 shows an example m-file template automatically generated by the software application.

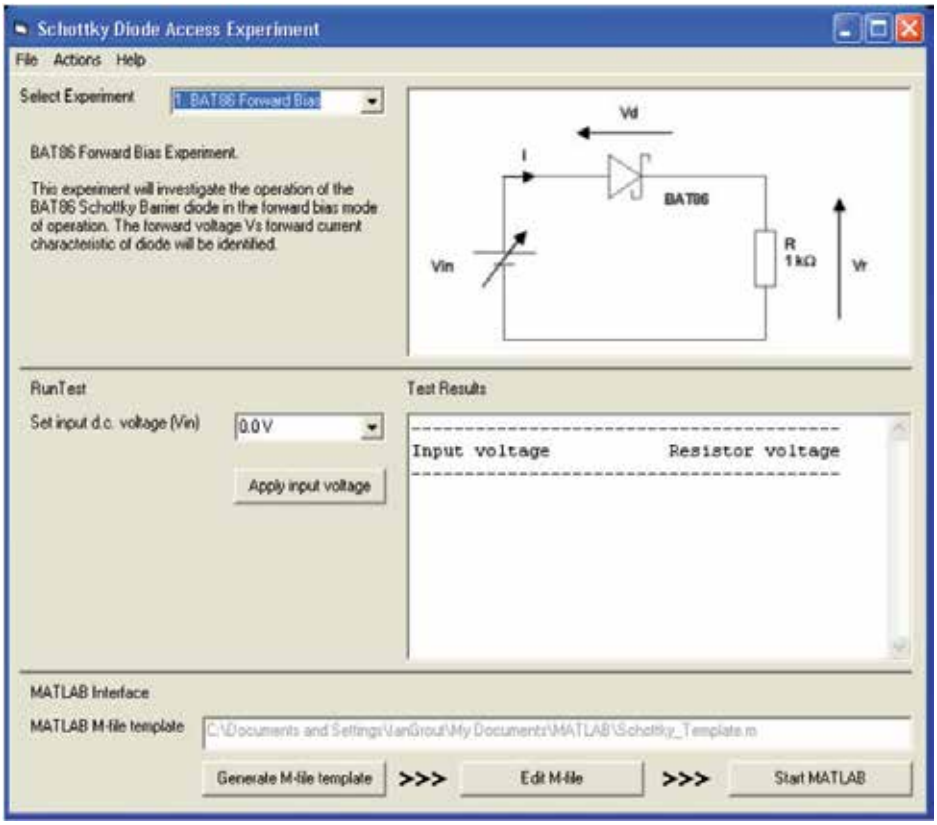


Figure 15. User interface GUI

```
%-----------------------------------------------------------------%
% MATLAB template file for BAT86 Schottky barrier diode experiment.
% BAT86 Forward Bias mode of operation.
%-----------------------------------------------------------------%
% Experiment undertaken on 13/01/2012 14:25:43
%-----------------------------------------------------------------%

%%-------------------
%% Test Results
%%-------------------

R = 1e+3

Vin = [0, 0, 0]
Vr  = [0.00, 0.00, 0.00]

%%-------------------
%% Analysis section
%%-------------------



%-----------------------------------------------------------------%
```

Listing 4. Automatically generated m-file template

The equipment set-up is shown in figure 16. Here, a laptop PC runs the software application and interfaces to the FPGA prototyping board via an RS-232 serial data link. The analogue I/O board operates on a +/-15 V d.c. power supply and test points are provided on the board to allow for the measurement of specific circuit voltages. The BAT86 Schottky barrier diode experiment is also incorporated on the analogue I/O (experiment) board.

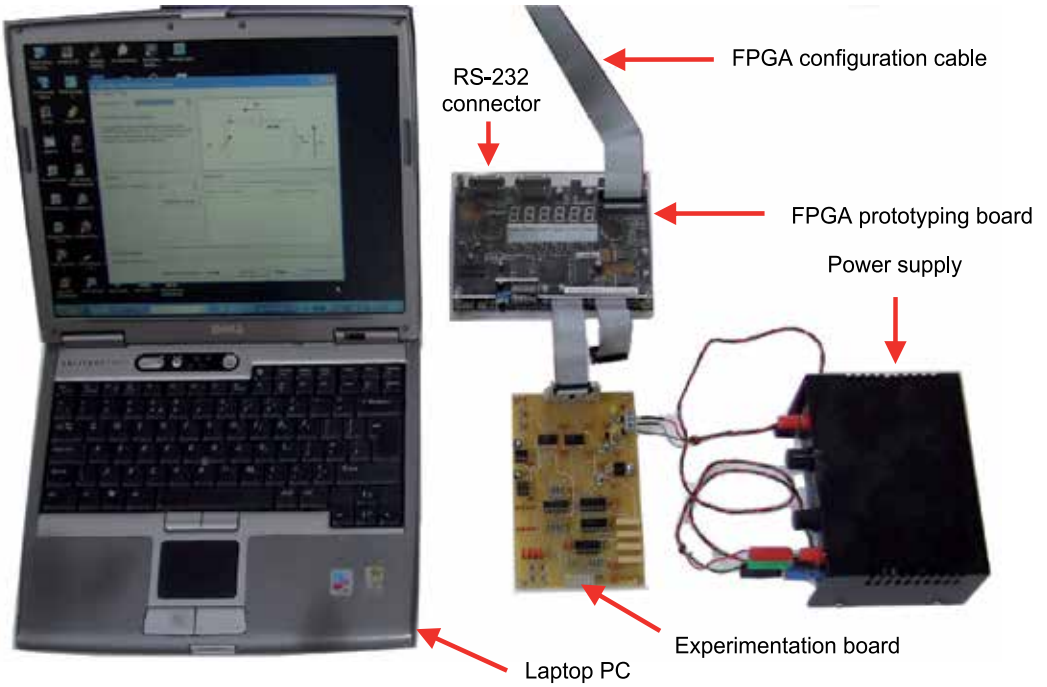


Figure 16. Computer set-up

6. Case study 3: Distance education learning scenario

Increasingly, universities are providing access to courses in a distance mode of operation – that is students are not physically located within the institution, but learn from an alternative (remote) location. In the simplest terms, students are registered to study for a qualification within the institution but access course material (lecture notes and assignments) via a suitable Internet connection and learning management system (LMS) such as Moodle [9] or a custom LMS solution. However, such a simplistic statement does not tell the whole story.

In engineering and science, there is a need to undertake experiments and analyse experiment results. This is traditionally undertaken *at-presence* where the learner undertakes the experiment within the laboratory facilities hosted by the institution. This approach might not be possible for distance learners and so alternative approaches to providing access to experimentation have been developed – distance learning utilising remote experimentation accessed through a remote laboratory [10]. The role that distance learning now undertakes within the teaching and learning environments on a global scale has gained widespread acceptance over the last number of years. Textural based teaching material, enhanced with graphics and animation, is now supported through the use of remote experimentation. Remote experimentation is essentially physical laboratory experiments that are set-up to be accessible via the Internet. Many institutions and organisations now provide for their laboratory experimentation to be Internet-enabled, so providing access via a web browser for remote users who may be in a location in the world that provides the user with an Internet access capability. This has been shown to be of high value from university education through to E-Science applications [11].

Industry can also benefit from the concept of distance access and learning scenario. Imagine for example, a parent company which has invested heavily in expensive equipment, for example integrated circuit (IC) testers. When the company extends its operations into other locations, it does not then need not to invest in the same equipment again. This is where the remote logging into the tester for device testing and data collection purposes can be done, and the analysis can carried out using MATLAB. In another scenario, the after sales service of a product can be made more friendly, easier and cost effective. In traditional approach, when a machine is down (i.e., not operational due to a fault), the customer has to wait for the service engineer to arrive from another part of the world to repair the machine. Using the concept of distance access, the service engineer needs only to be remotely connected to the down machine and MATLAB can once again be useful to analysis the symptoms and suggest possible causes, along with solutions. Of course the symptom has to be first modelled accurately just like the diode and other devices are modelled.

Here and in this distance based learning scenario, the basic experimentation set-up and discussed in section 5 (Case study 2: computer aided learning scenario) can be modified to be accessible via the Internet. Essentially, a web-server arrangement (a WAMP (Windows, Apache [12], MySQL [13] and PHP [14])) system is set-up here and the experiment user

interface is via a series of Internet browser pages (web pages). Figure 17 shows the home page set-up for the experiment.

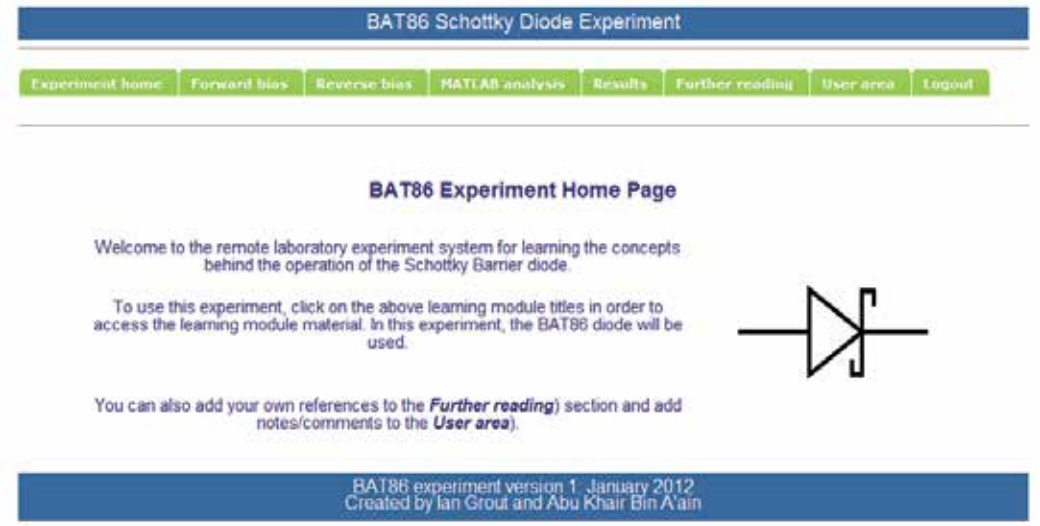


Figure 17. User interface (home page)

In the user interface, the user is considered to already have successfully logged in to the remote laboratory via a username/password arrangement and is then given access to a diode experiment menu system (top of the page) for accessing different parts of the experiment. These options are identified in table 4. The user is prompted to access the different web pages in order to access different aspects of the experiment, including access to MATLAB via the web server arrangement.

Option	Option description
Experiment home	Return to this (home) page.
Forward bias	Run the diode forward bias experiment.
Reverse bias	Run the diode reverse bias experiment.
MATLAB analysis	Create a MATLAB function to enter and analyse the experiment results.
Results	View the results from previously run experiments and MATLAB analyses.
Further reading	List of references (which can be added to by users).
User area	Area to allow users to add notes for all users to access.
Logout	Log out from the overall remote laboratory arrangement.

Table 4. User interface options

In order to run an experiment, the user chooses the diode in forward bias or reverse bias mode of operation via the top menu system. Figure 18 shows the forward bias experiment web page. This provides for an introduction to the experiment and an area to enter the values of the input voltage (*Vin*) to apply to the test circuit. This is via an HTML form seen to the right of the page. All values are entered into the form and then submitted to the web server (and hence the experiment) using the *Submit* button.

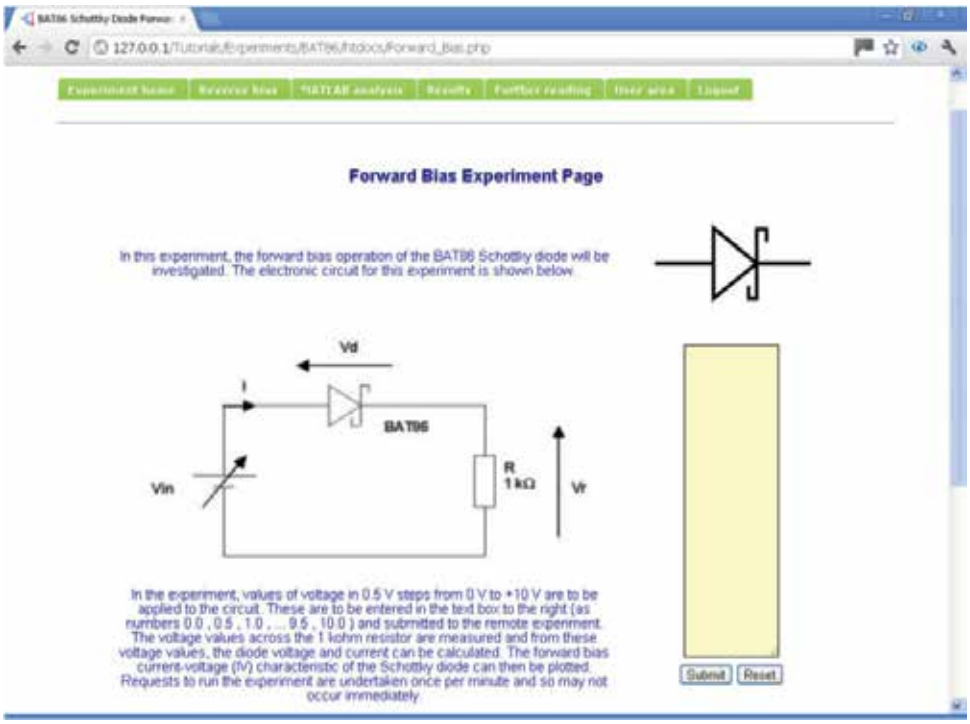


Figure 18. Remote submission of diode forward bias experiment input voltage values

Figure 19 shows the reverse bias experiment web page. This page has the same form as the forward bias experiment, except now on submission of the input voltage values, the reverse bias experiment is selected rather than the forward bias experiment.

As the experiments are performed remotely with the experiment electronic hardware connected to the web server PC, the user has a much more restricted and controlled access to the experiment. In this arrangement, the experiment is run on the remote web server and the results are accessible via a web page. Here, the user does not have interactive control of the experiment, rather they submit their values and the web server treats this as a job to complete, allowing the same experiment to be used by multiple users without the need for the user to pre-book the experiment.

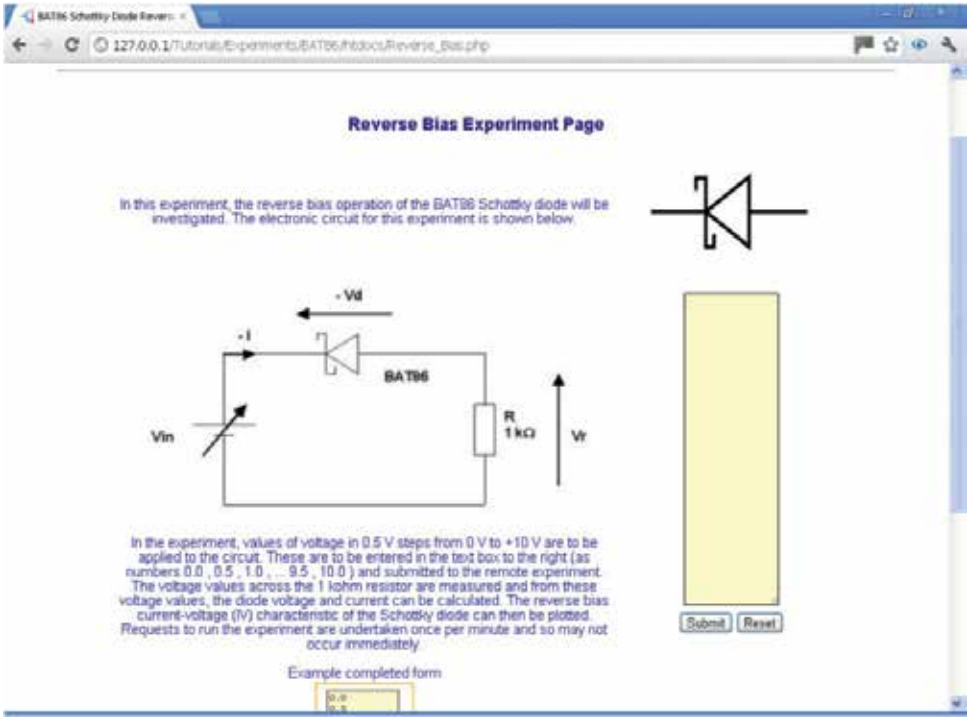


Figure 19. Remote submission of diode reverse bias experiment input voltage values

Figure 20 shows the experiment results page that the user sees. Their submitted test values and the experiment results are available as hypertext links via this web page. Hence, the user can see the text based experiment results which would then be used in MATLAB to analyse the results.

The MATLAB analysis is undertaken by entering the code for a MATLAB *function* into a form on the web page and then submitting the *function* to the web server for remote processing. On completion of the processing, the results are available for the user to access.

In figure 21, a template form for the MATLAB code as a function is automatically presented to the student and they complete the function with their own comments. This is shown in more detail in figure 22 and listing 5. The user does not modify this code structure, but inserts their own code between the "% Start of user commands" and "% End of user commands" comments.

On submission of the completed form, the code is automatically saved in a function m-file, MATLAB automatically executes the m-file commands and the results are saved to suitable results files.

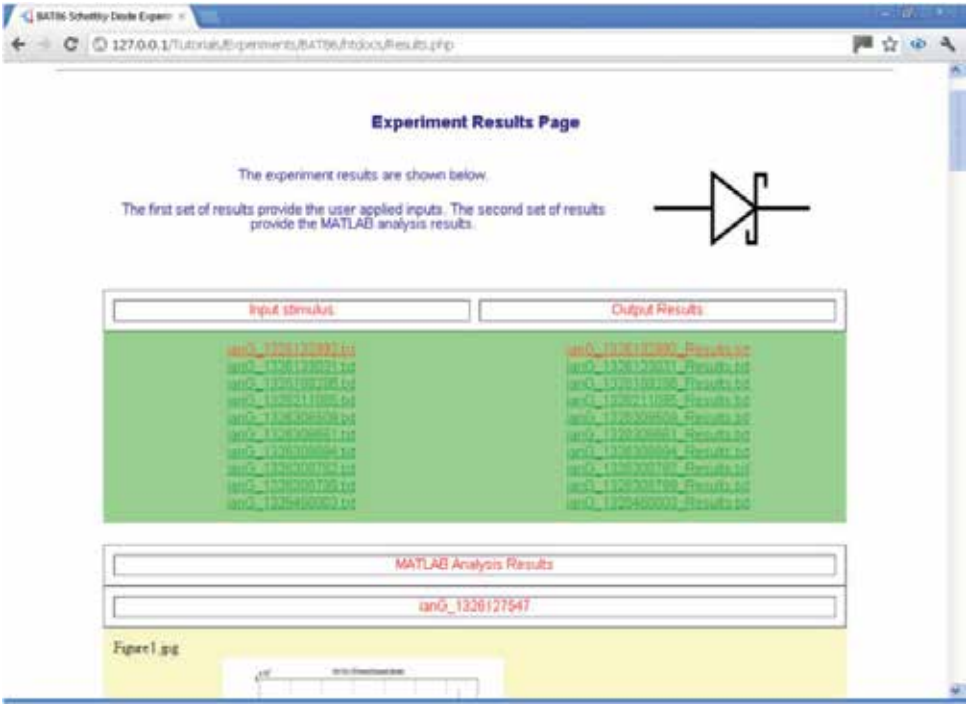


Figure 20. Viewing experiment results

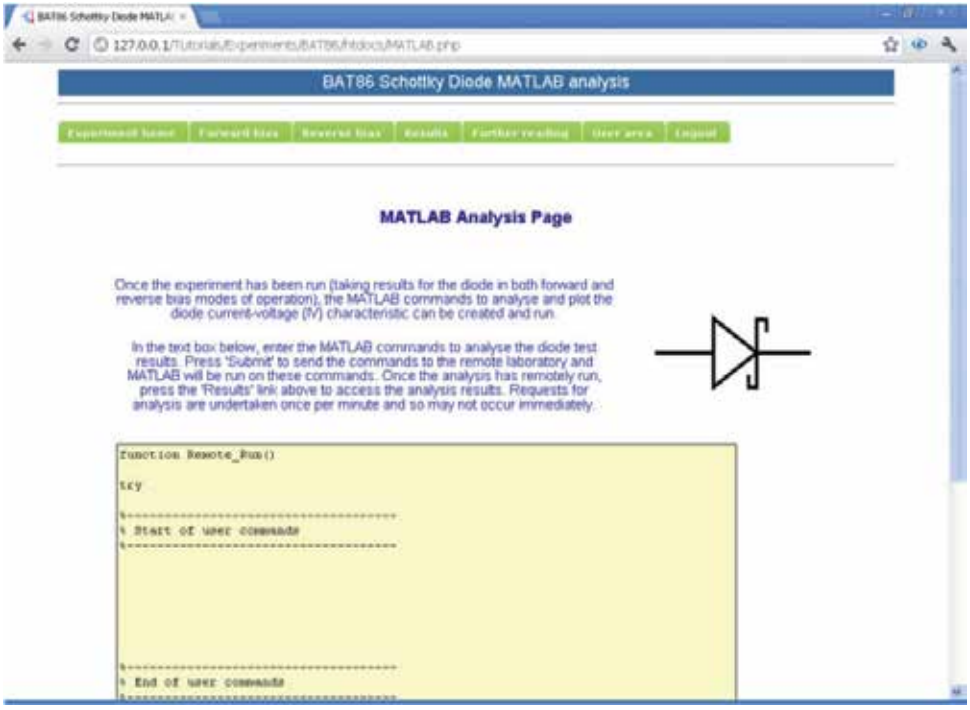


Figure 21. Remote submission of MATLAB commands to web server for remote processing

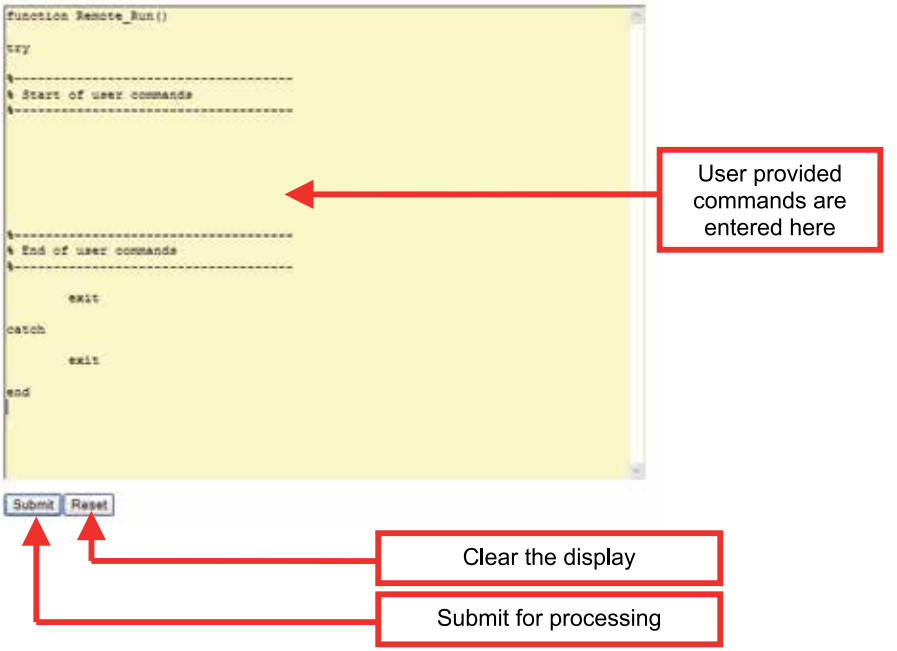


Figure 22. Remote submission of MATLAB commands to web server for remote processing

The function template code is shown in listing 5. On submission, this function is saved into the m-file and MATLAB is executed on the web server PC. In order to handle possible errors in the submitted code, then the *try – catch* error control is used.

```
function Remote_Run()

try

%-----------------------------------
% Start of user commands
%-----------------------------------









%-----------------------------------
% End of user commands
%-----------------------------------

        exit

catch

        exit

end
```

Listing 5. MATLAB function template

Two important aspects of using MATLAB in this manner are:

1. MATLAB is run remotely on a different computer and so any instant visual feedback that a user would see when running MATLAB on his or her own computer is not immediately available.
2. If any errors are encountered in the user provided code, MATLAB must exit "gracefully" and not simply crash. It must also be able to provide suitable error feedback to the user.

To this extent, the above function template captures and reports errors, and the text feedback that would normally be seen in the MATLAB Command Window is automatically outputted to a text log file (and this file is available via the Results Page).

A third aspect of using MATLAB in this manner is that any figures to plot the results would not be seen if simply the plot(x, y) command was used.

To this extent, whenever a figure plot is to be produced, this will need to be saved as an image file (using the commands of the form shown in listing 6) and if the image file is in JPEG format, it will be displayed in the Results page along with the MATLAB log file and the uploaded analysis code.

```
figH = figure('visible', 'off')                    Create figure


      plot(Vd_forward, Id_forward)                 Plot to figure
      grid                                         Add grid
      title('Vin Vs I (Forward biased diode)')     Add title
      xlabel('Vin (volts)')                        Add x-axis label
      ylabel('I (amps)')                           Add y-axis label


      print(figH, '-djpeg', 'Figure1.jpg')         Print figure to JPG file


close(figH)                                        Close figure
```

Listing 6. MATLAB: plotting the figure to a JPEG image file

Here in listing 6, a figure is created and plotted to (along with the annotation required by the user). Here, a figure called *figH* is created and plots the variables **Vd_forward** and **Id_forward** along with a figure title and axis labels. This is then saved to an image file in JPEG format with the file name Figure1.jpg.

Once an analysis has been undertaken, the results are available for viewing on the results page in the format as shown in figure 23. Here, the analysis run name (a unique run name), the image files produced, the generated MATLAB m-file and the MATLAB log file are available for viewing.

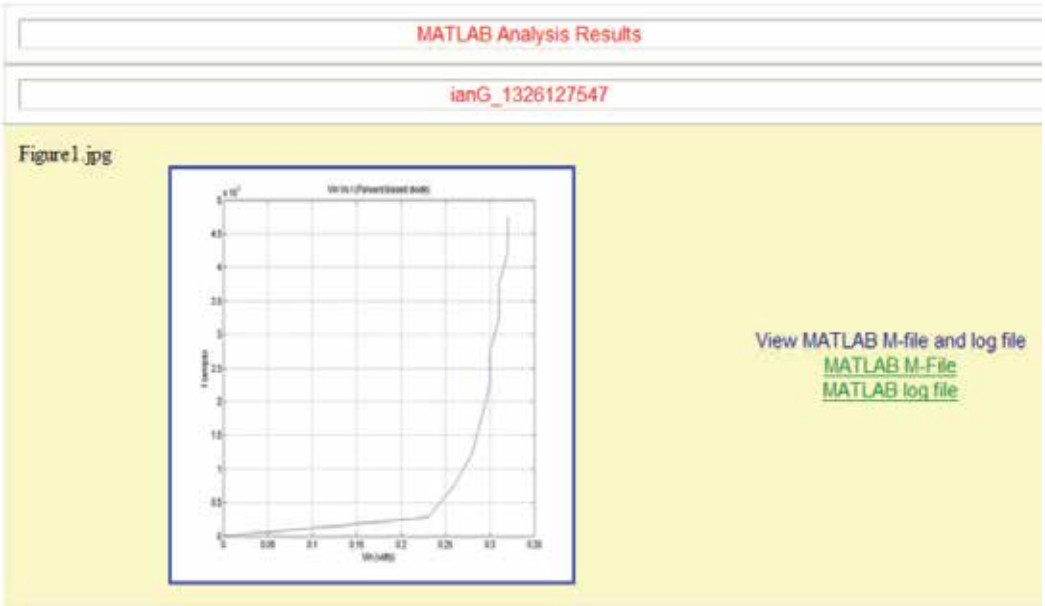


Figure 23. MATLAB analysis results access

The user can click on the computer mouse on the image and this shows the full-size figure in a new browser window. Where the user runs an m-file to create multiple figures and save these as image files, each generated image file is shown in the results window and is therefore accessible. The image file can then be saved to the student's own computer for later inclusion in experiment reports.

In this work, the infrastructure to support the use of MATLAB to be accessed via an Internet browser page is developed and discussed.

A final comment to make however relates to the use of the MATLAB software license where MATLAB runs on a particular computer, but is accessed remotely from a separate computer. It is not the intention to discuss specifics about the use of the software license and if there is any doubt about the use of the license in this mode then the provider of the license should be consulted.

7. Conclusions

This chapter has presented and discussed the use of MATLAB within an education environment with reference to the teaching and learning of semiconductor device fundamentals. Specifically, MATLAB can be integrated into the education curriculum as a tool to provide specific analysis and results presentation operations, these being (i) physical electronic circuit test results data entry, analysis and graphical plotting; (ii) idealised device characteristic equation modelling and graphical plotting; (iii) comparisons between idealised and actual device performance; and (iv) documentation preparation purposes. In

this work, consideration was given to the use of MATLAB in three teaching and learning scenarios; (i) at-presence "traditional" laboratory experiments; (ii) at-presence computer aided learning laboratories; and (iii), distance based remote access to laboratory experiments. Each scenario was introduced and the development of the laboratory experiments discussed. The physical infrastructure (electronic hardware and software) was identified and the role in which technology is utilised in the education environment was presented. In particular, the way in which MATLAB was considered to be used and how it was integrated into custom developed education technology tools were highlighted. The discussions were based on the evaluation of the Schottky barrier diode, specifically the BAT86 Schottky barrier diode. However, the discussions, arguments and experimentation hardware and software can be readily adapted to other forms of semiconductor devices.

Author details

Ian Grout
Department of Electronic and Computer Engineering, University of Limerick, Limerick, Ireland

Abu Khari Bin A'ain
Faculty of Electrical Engineering, Universiti Teknologi Malaysia (UTM), Skudai, Johor, Malaysia

8. References

[1] The Mathworks Inc., MATLAB®, 2012, www.themathworks.com
[2] NXP, BAT86 Schottky barrier diode product datasheet, 2004, www.nxp.com/documents/data_sheet/BAT86.pdf
[3] Duane Hanselman and Bruce R. Littlefield, Mastering MATLAB 6, 6th Ed., Prentice Hall; 1 edition (14 Dec 2000), ISBN 0130194689
[4] Pierret R.F., Semiconductor Device Fundamentals, 1996, Addison-Wesley Publishing Company, ISBN 0-13-178459-5x
[5] Sze S.M., Semiconductor Devices Physics and Technology, 1985, John Wiley & Sons. Inc., ISBN 0-471-83704-0
[6] Xilinx Inc., Spartan-3 FPGA Family Data Sheet, 2009, www.xilinx.com/support/documentation/data_sheets/ds099.pdf
[7] Altium, TR0104 LiveDesign Evaluation Board Technical Reference Manual, 2004
[8] Microsoft Corporation, 2012, Visual Basic 6.0, msdn.microsoft.com/en-us/vstudio/ms788229
[9] Moodle, 2012, moodle.org/
[10] Ian Andrew Grout and Alexandre César Rodrigues da Silva, "Analysis Tool for Remote Laboratory Structures", Proceedings of the Remote Experimentation and Virtual Instrumentation conference (REV 2010), Stockholm, June 29 - July 2, 2010, www.rev-conference.org/REV2010/
[11] R. Ratering et al., "GridBeans: Supporting e-Science and Grid Applications", Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing", 4-6 Dec. 2006, Amsterdam, The Netherlands

[12] The Apache Software Foundation, Apache HTTP Server Project, 2012, httpd.apache.org/
[13] MySQL, 2012, dev.mysql.com/
[14] PHP, PHP Hypertext Preprocessor, 2012, www.php.net/

Chapter 16

An Interactive Tool for Servo Systems Learning

Nourdine Aliane, Rafael Pastor Vargas and Javier Fernández Andrés

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46459

1. Introduction

Servo systems play an important role in industry. They are found in most automated manufacturing systems, machine tools, and robotic systems to cite but a few examples. Servo systems fundamentals have become an integral part of industrial electronics and other related fields in engineering, such as electrical engineering or computer disciplines. Servo systems topics should be extended to all engineering curricula [1]. It is therefore a major challenge to ensure that future engineers should be familiar with servo systems, and be able to analyze and control them.

A direct current controlled motor (DC motor) is considered to be the simplest form of servo system, and is used as a starting point for understanding all other electric machines. Control of DC motors is widely taught in control engineering and robotics courses, and is commonly used in laboratory experiments as providing an excellent case study. The importance of the material is well-evidenced in many textbooks [2]-[5]. However, in most of these texts, servo system problems are highly simplified for pedagogical purposes, and the given examples focus on the linear parts, and do not take into account practical issues such as compliance coupling, trajectory generation, the wind-up effect, feed-forward compensation, or torque limitation.

This paper presents an interactive learning module focused exclusively on servo systems. It is aimed at bridging the gap between theoretical background and experimentation, providing insight into fundamental concepts. The tool is based on exploiting interactivity as a pedagogical basis in teaching and learning activities. Although many interactive tools have been developed for general topics in control education [6]-[9], interactive tools focusing on servo systems are practically nonexistent.

There are several works in the literature which deal with the use of Matlab-Simulink for servo systems simulations for educational purposes [10]-[13], but the tools presented in these works lack interactivity. The Matlab Central File Exchange [14] is also used as a source

to search for servo systems educational tools. Several excellent demonstrations can be found, but none of them meet the criteria for simulating practical aspects of servo systems. It is also worth mentioning the use of remote laboratories, where DC motors are successfully integrated in remote experimentation platforms [15]-[16]. However, remote experimentation is currently limited to performing simple experiments, and topics such as the effects of disturbances, anti-windup, or feed-forward, are not considered.

The development of the module presented in this paper is based on Simulink models used in combination with the Matlab graphical user interface (GUI). The choice of the Matlab platform is due to the fact that it is by far more productive than many other high level programming languages. Simulink, on the other hand, is used to easily model and simulate a variety of systems using intuitive block diagrams. Wrapping Simulink models within Matlab-GUI is advantageous and improves interactivity. This feature allows users to change parameters and view simulations without having to deal directly with Simulink blocks. Indeed, Matlab includes a built-in tool called GUIDE, which permits developers to design GUIs for Matlab applications.

The remainder of the paper is organized as follows: Section 2 presents the objectives and scope of the tool. Section 3 gives some theoretical background on servo systems. Section 4 describes the interactive module and shows how Simulink models can be integrated within an interactive Matalb-GUI. Section 5 describes usage experiences and gives some classroom examples. Section 6 describes the methodology used in evaluation and discusses the results obtained. Finally, Section 7 concludes the paper.

2. Objectives and scope

In the author's institution, servo systems topics are an integral part of undergraduate control engineering and robotics courses. This part of the material examines servo systems principles, providing students with modeling techniques for DC motors, the formulation of the control problem, and an introduction to the basic treatment of feedback design. The theoretical background is supplemented with hands-on laboratory work. The laboratory activities are aimed at showing the students some qualitative aspects of servo systems control and giving them a broad picture of axis control. However, there is neither enough time nor sufficient equipment to experiment with all the practical aspects of servo systems.

To remedy this situation, the methodology was recently modified to place greater emphasis on fundamental concepts, because the tuning of servo systems can be confusing due rather to unfamiliar principles than to their complexity. Thus, to help students develop their knowledge of, and intuition for, servo systems and their control, an interactive module, dedicated exclusively to this topic, was developed. The tool is aimed at speeding up learning by reducing the time needed to design simulations. It is also intended to be interactive in the sense that it allows students to see immediately the effects of changing parameters on the behavior of systems. Finally, the tool is intended to complement rather than replace laboratory work.

The developed tool simulates the behavior of a separately-excited DC motor, and covers several practical issues of servo systems. Students can, among other capabilities, configure the model, perform simulations of position and speed control, interactively tune the controller, specify trajectories, and compare different control algorithms. Lastly, the user can perform comparisons of simulations by evaluating some performance measures.

3. Servo system background

3.1. Servo system modelling

Servo systems topics, as treated in many textbooks, consider simple models and assume that the motor shaft and its load are rigidly coupled. In practice, however, most servomechanisms contain flexible modes in the actuator or the structure being controlled. This is particularly true in systems such as disk drives, antennae and radar pointing or robotics axes, where coupling might have significant compliance. For a model to be useful, it must be realistic and yet simple enough to understand and manipulate. In general, flexible coupling is modeled as a rotational spring, which is sufficient to yield models of high order.

The interactive tool presented in this paper simulates a DC motor coupled to its load through a shaft with variable compliance (see Figure. 1). The user can set the model to what s/he considers relevant (a compliantly or rigidly coupled model).

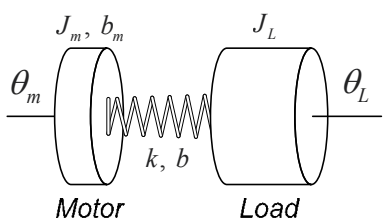


Figure 1. Model of motor and load compliantly coupled.

In the case of an armature-controlled DC motor, the control input is the armature voltage U. The torque T provided by the motor is proportional to the armature current i, and the back emf e is proportional to the motor's velocity ω, by the following equations

$$\begin{aligned} T &= k_m i \\ e &= k_m \omega \end{aligned} \tag{1}$$

The electrical equation is

$$U - e = L\frac{di}{dt} + Ri \tag{2}$$

where R is the armature resistance and L is the armature winding inductance.

The torque provided by the motor is transmitted to the load through the compliant shaft, which is modeled as a rotary spring with constant k and damping constant b. The mechanical equations of the motion are:

$$\begin{aligned} &J_m\ddot{\theta}_m + b_m\dot{\theta}_m + b\left(\dot{\theta}_m - \dot{\theta}_L\right) + k\left(\theta_m - \theta_L\right) = k_m i \\ &J_L\ddot{\theta}_L - b\left(\dot{\theta}_m - \dot{\theta}_L\right) - k\left(\theta_m - \theta_L\right) = 0 \end{aligned} \tag{3}$$

where subscript m refers to the motor, subscript L refers to the load, J is an inertia, and b is a damping constant.

Defining the sate vector as $x = \left[i, \theta_m, \dot{\theta}_m, \theta_L, \dot{\theta}_L\right]^T$, the state space representation of the DC motor with compliant coupling

$$\dot{x} = \begin{bmatrix} -R/L & 0 & -k'_m/L & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ k_m/J_m & -k/J_m & -(b+b_m)/J_m & k_m/J_m & b/J_m \\ 0 & 0 & 0 & 0 & 1 \\ 0 & k/J_L & b/J_L & -k/J_L & -b/J_L \end{bmatrix} x + \begin{bmatrix} 1/L \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u \tag{4}$$

If the coupling shaft is rigid, then $k \to \infty$, and $\dot{\theta}_m = \dot{\theta}_L$. Adding the two mechanical subsystem equations (3) yields

$$J_m\ddot{\theta}_m + J_L\ddot{\theta}_L + b_m\dot{\theta}_m = k_m i \tag{5}$$

Thus, the mechanical subsystem becomes

$$\left(J_m + J_L\right)\ddot{\theta}_m + b_m\dot{\theta}_m = k_m i \tag{6}$$

Defining the total moment of inertia as $J = J_m + J_L$, and the state as $x = \left[i, \theta_m, \dot{\theta}_m\right]^T$, the state space equation is now simplified as

$$\dot{x} = \begin{bmatrix} -R/L & 0 & -k'_m/L \\ 0 & 0 & 1 \\ k_m/J & 0 & -b_m/J \end{bmatrix} x + \begin{bmatrix} 1/L \\ 0 \\ 0 \end{bmatrix} u \tag{7}$$

3.2. Servo system control

Despite the development of more advanced control techniques, the PID controller is still the most common algorithm used in servo systems applications [17]. The attractiveness of PID resides in the fact that an accurate model is not required and its control capabilities have been proven to be adequate for controlling servo systems. Indeed, PID controllers are used in many dedicated motion-control, such as LM628/629 [18], Magellan [19], Galil boards [20], or the Quanser DC motor model for training and education [21].

The mathematical expression in the time domain is given by the equation (8).

$$u(t) = K_p\left(r(t) - y(t)\right) + K_i\int\left(r(t) - y(t)\right)dt + K_d\frac{d\left(r(t) - y(t)\right)}{dt} \tag{8}$$

where u(t) is the controller output, r(t) is the reference, y(t) is the measured system output, and *Kp, Ki* and *Kp* are the PID adjustable parameters. In real servo systems, PID controllers can perform poorly when used alone. To enhance the system performance, a number of additional mechanisms such as handling the wind-up effect properly, combining the feedback with feed-forward control, or using a trajectory generator, are adopted as technical solutions.

3.2.1. Anti-windup correction

The elimination of the steady state error in servo systems has long been performed using the integrator action. However, this action has the disadvantage of causing the wind-up effect, which occurs when the calculated control signal exceeds its saturation limits and the controller is unable to respond immediately to changes in the error signal. To prevent the windup effect, the operating range of the control signal should be limited to the range of the voltage input of the servo [22]. This ad-hoc solution provides instant recovery when the error signal changes signs. There are many way to ovoid the integrator wind-up and one of these possibilities is illustrated in the following Simulink diagram.

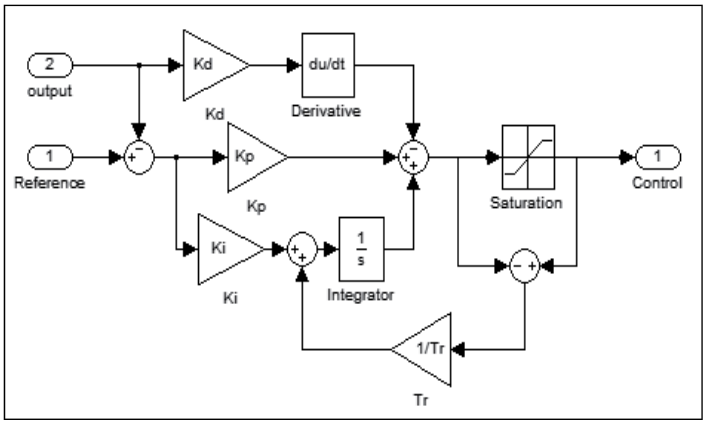


Figure 2. PID controller with Anti-Windup correction.

3.2.2. Feed-forward control

Feed-forward is another technique used to improve servo systems performance and is essentially used to reduce the tracking error in high performance motion control problems. Theoretical developments show that the feed-forward transfer function is the inverse dynamic of the servo. In general, feed-forward compensation is performed through the required acceleration and velocity.

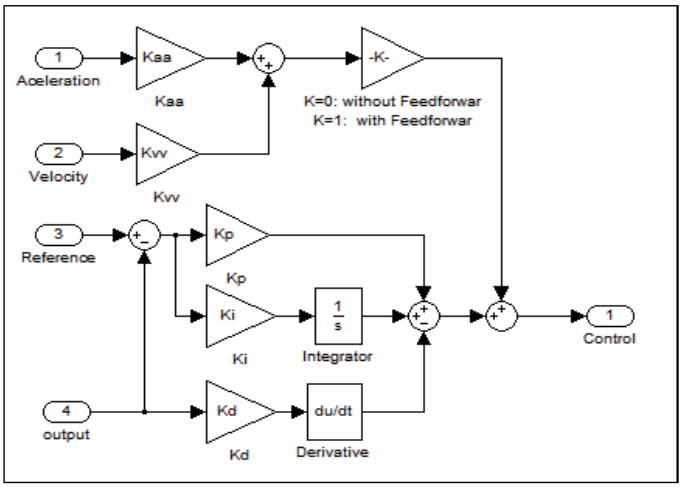


Figure 3. PID controller with feed-forward control.

3.2.3. Motion control

In real servo systems control, the position step references can cause controller saturation and lead to significant overshoot, and indeed, this kind of references are hardly used. The step response is actually used as a measure of system performance. To overcome these problems, references known as S-curves, such as parabolic or trapezoidal profiles, are used instead. These S-curves are provided by a trajectory generator, which is an algorithm at the top of control hierarchy.

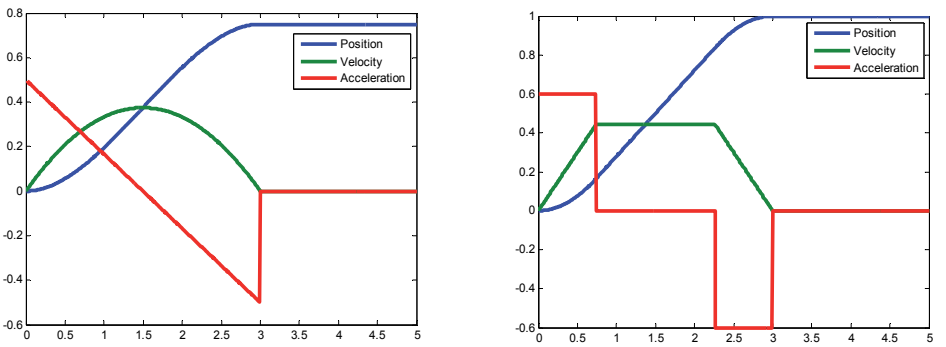


Figure 4. Parabolic and trapezoidal profiles for motion control.

4. Interactive tool description

As mentioned above, the interactive tool is composed of two parts: a Matlab-GUI application and a Simulink model. The Simulink model is completely transparent for end users, and is automatically opened and runs in the background. The tool is freely available

on the World Wide Web [23]. The layout of the GUI of the interactive module is shown in Figure 5.

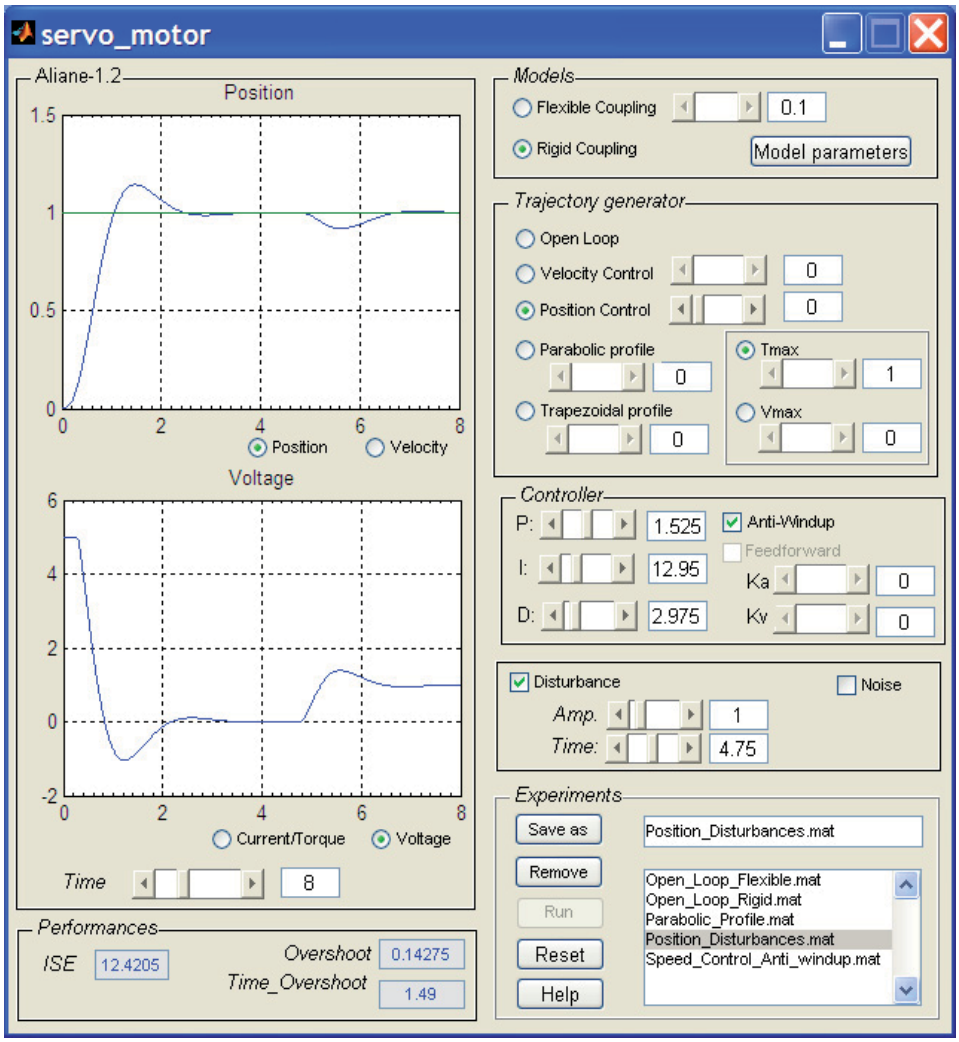
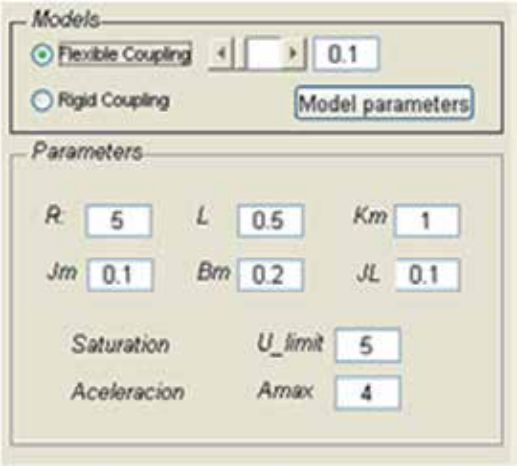


Figure 5. The servo systems learning module main user interface.

Its interface visually displays all the actions that users may execute. In the first group of controls, users can select between a rigid or flexible coupling model. The difference between these models can be visualized by their open-loop step response. Furthermore, the compliance of the flexible coupling can be changed interactively which allows students visualize the significance of coupling stiffness better. Users can also customize the model by setting the model parameters in the "model parameters" button, which displays a window for capturing model parameters.



R: Electric resistance
L: Electric inductance
Km: Electromotive force constant.
Jm: Moment of inertia of the rotor
Bm: Damping ratio factor
JL: Moment of inertia of the load
U_limit: Maximal input source voltage
Amax: Maximal motor acceleration

Figure 6. DC-Motor model parameters.

This is suitable for motor modeling, and it is particularly useful for investigating the effects of some DC-motor's parameters on the speed and how the induced current is modified.

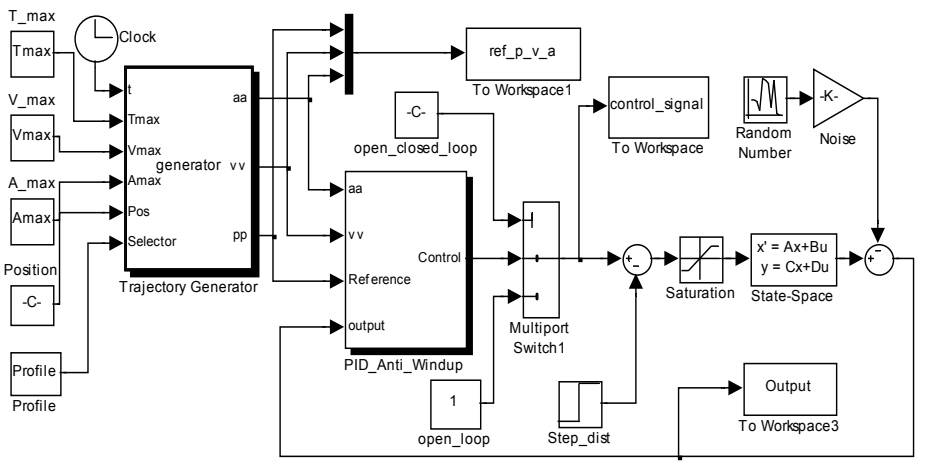
In the second group, users can select between simulating a velocity or position control. In the position besides the step input reference, users can choose parabolic or trapezoidal profiles as a trajectory reference. The third group deals with control, where users can interactively modify the PID tuning parameters and immediately see their effect on the system behavior. In addition, the feedback control can be augmented with a feed-forward and an anti-windup correction. The fourth group allows the simulation of the effect of exogenous factors such as external disturbances and noise measurements. The fifth group lets users save and retrieve simulations, which is particularly useful for drawing straightforward comparisons between different scenarios. Finally, as well as visualizing velocity, position, induced current, and voltage input, some performance measures, such as the integral of squared error (ISE) and the overshoot, are also displayed.

The design of the Matlab-GUI is easy using the Matlab GUIDE. This toolkit is opened by entering the command ">>guide" in the command line. Then, images of all the elements (sliders, axes plotting, etc) are dragged out, and then the user writes their respective callback functions. The reader can go further with the help of the Matlab GUI manuals.

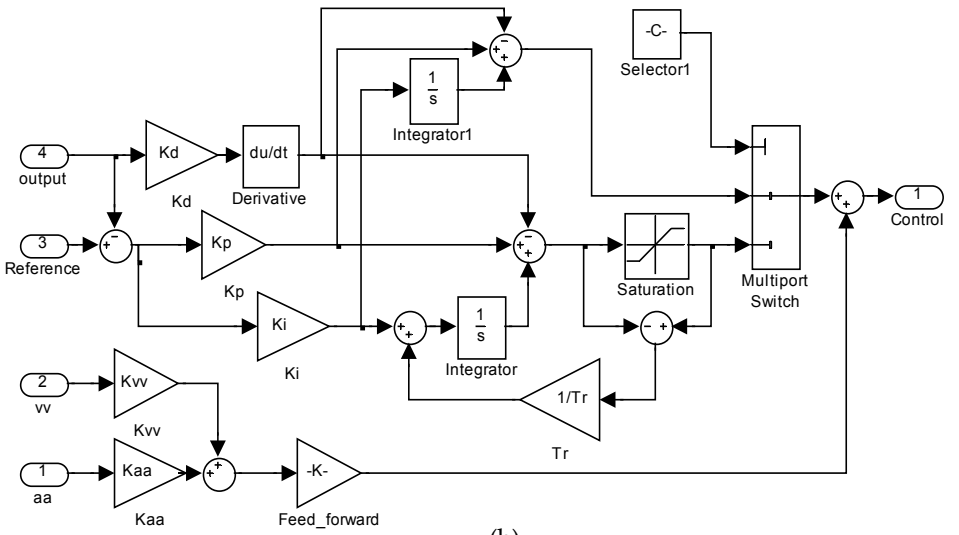
The Simulink block diagram implementing the whole servo systems simulation is shown in Figure 7. Interfacing Simulink models with Matlab GUI applications is easy, but not trivial operation. The control of Simulink models can be performed through suitable Matlab commands. Simulink blocks can be accessed and their parameters changed by using the "set_param()" function. Alternatively, the communication between Simulink models and Matlab-GUI can be performed by using variables defined in the main workspace. These two

mechanisms are also valid for specifying other Simulink parameters such as sample time or time simulation through the GUI. Lastly, a Simulink simulation can be run by using the "sim()" function.

As far as the trajectory generator is concerned, this is implemented as an "embedded Matlab function block," which is a Simulink block that contains a compiled Matlab code. Unlike "Matlab function blocks," which accept multiple inputs and support only a single scalar output, the functions in "embedded Matlab function blocks" allow multiple inputs and return multiple outputs.



(a)



(b)

Figure 7. a) Simulink block diagrams: General structure for servo systems simulation; b) The controller subsystem: the PID, anti-windup, and feed-forward control.

5. Classroom experience

The tool was used during the 2006-07 and 2007-08 academic years at Universidad Europea de Madrid (Spain), in control engineering and robotics subjects, providing the instructor with a valuable supplement to lectures by projecting the module screenshots. The first advantage of the tool presented here is that it allows the instructor to set up simulations with minimum effort, which obviously helps him to maintain the flow of the lecture, and allows him to pose different cases quickly in response to the students' questions. The module proved to be an effective instrument to facilitate teaching, since the interactive nature of the module helps to draw comparisons between simulations, explore the effect of varying parameters, and give explanations through "what-if" scenarios.

The learning module was also used by approximately 35 students in the classroom, following guided exercises. These exercises were focused on students getting a feel for the qualitative aspects of servo systems, having them explore what would happen if a given parameter were increased or decreased. The quantitative aspects were not emphasized as much. For example, students are asked to undertake the problem of position control and tracking trajectories. After a review of the PID controller, students are asked to tune the controller through a trial and error process, taking advantage of the interactivity of the tool. As a first step, students use proportional action (P) and examine steady-state error. Next, they introduce the integral action (I) to see how the steady-state error is eliminated. Finally, the derivative action (D) is activated to increase the dumping of the system. Robustness is also an important aspect in servo systems. In another exercise, simulations are performed to assess control strategy with respect to external disturbances, considering cases where a motor is holding a final position and a disturbance is applied to its shaft. Thanks to the module's interactivity, many other concepts such as flexible coupling, anti-windup and feed-forward corrections, or the use of S-curve references are presented with animated graphs examining the effects of some parameters. These exercises are solved in group, and the use of the interactive tool provides an excellent context for provoking discussions and reflections.

To foster interest in the use of the tool, a comprehensive tutorial, providing a number of handouts on classroom activities, was also developed [23]. This tutorial starts with an introductory level on modeling techniques, and goes into more detail on using the PID controller, anti-windup, feed-forward and trajectory generation. It provides a total of ten guided examples; some are projected as demonstrations during lectures while others are developed as exercises and as a basis for group discussion on more in-depth theoretical topics.

5.1. Classroom examples

Many are the classroom activities that can be performed by the tool presented. This subsection shows three classroom examples that illustrate the module's capabilities. Although these examples do not fully exploit all of the tool's features, they illustrate some important aspects of servo systems.

Example-1: Modeling and open-loop simulation

Models are an essential element of servo system analysis and design, and serve to understand how systems behave with and without control. This aspect is covered through demonstrations shown during a lecture, carrying out comparisons of a motor's open-loop responses with rigid and flexible coupling, as well as observing dynamically how changes in the compliance parameter affects the shapes of the responses. Furthermore, different parameters of the model are taken into account and are changed dynamically in order to assess their effect in light of the modeling concept. Step responses of a DC motor with rigid and flexible coupling are shown in Figure. 8.

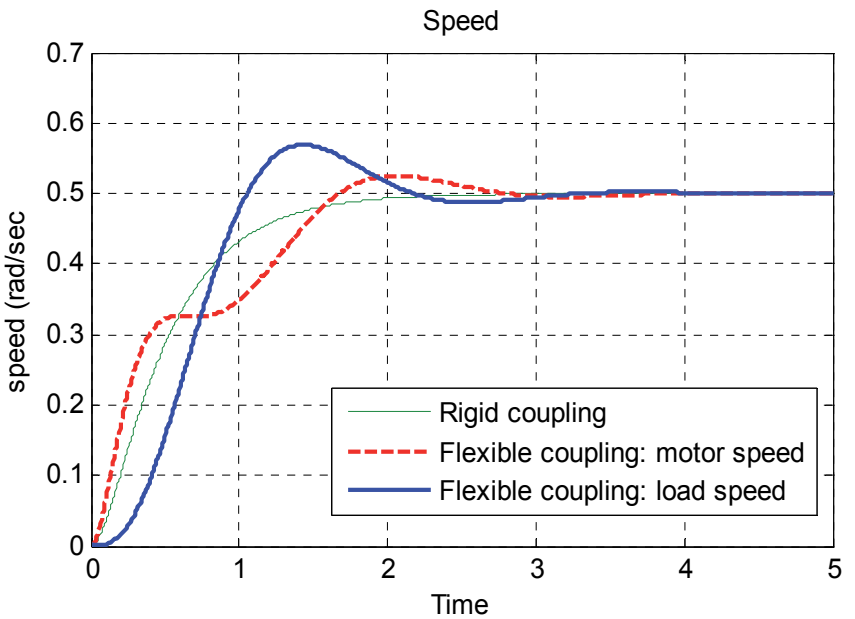


Figure 8. Step responses of a DC motor with rigid and flexible coupling (k = 0.7).

Example-2: Speed control and anti-windup correction

The tool is used to study velocity control and its characteristics. For example, one exercise is devoted to simulating PI control, applying poles placement strategy. The tuning parameters are adjusted analytically assuming a simplified model. Afterwards, students are required to compare simulation outputs applying the same controller tuning to the model with flexible coupling. In the context of speed control, another exercise is dedicated to demonstrating the benefits of anti-windup correction. In this section, students are required to compare the system's output under different conditions: namely, a simulation with no saturation limits, a simulation with saturation limits, and a simulation with anti-windup compensation. This comparison is illustrated in Figure. 9.

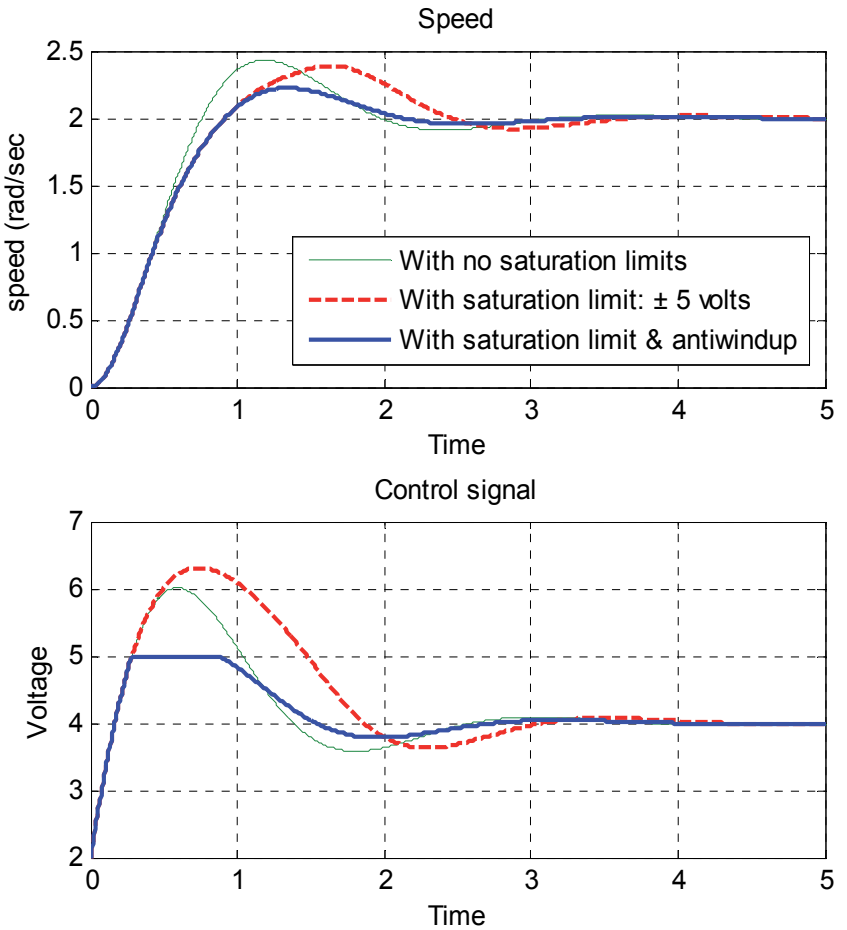


Figure 9. Speed control using a PI controller with and without anti-windup effect (P=1 I=7).

Example-3: Parabolic velocity profiling and feed-forward

Servo systems behavior is often characterized by their step responses. However, it is also useful to see how servo systems behave under S-curves, such as parabolic or trapezoidal profiles. Students are asked to investigate the effect of these trajectories and how input voltage and current are influenced. Lastly, activities concerning feed-forward compensation are also possible, demonstrating how near-zero tracking error can be achieved. Figure. 10 shows a comparison of position control through a parabolic profile, and illustrates the effect of feed-forward.

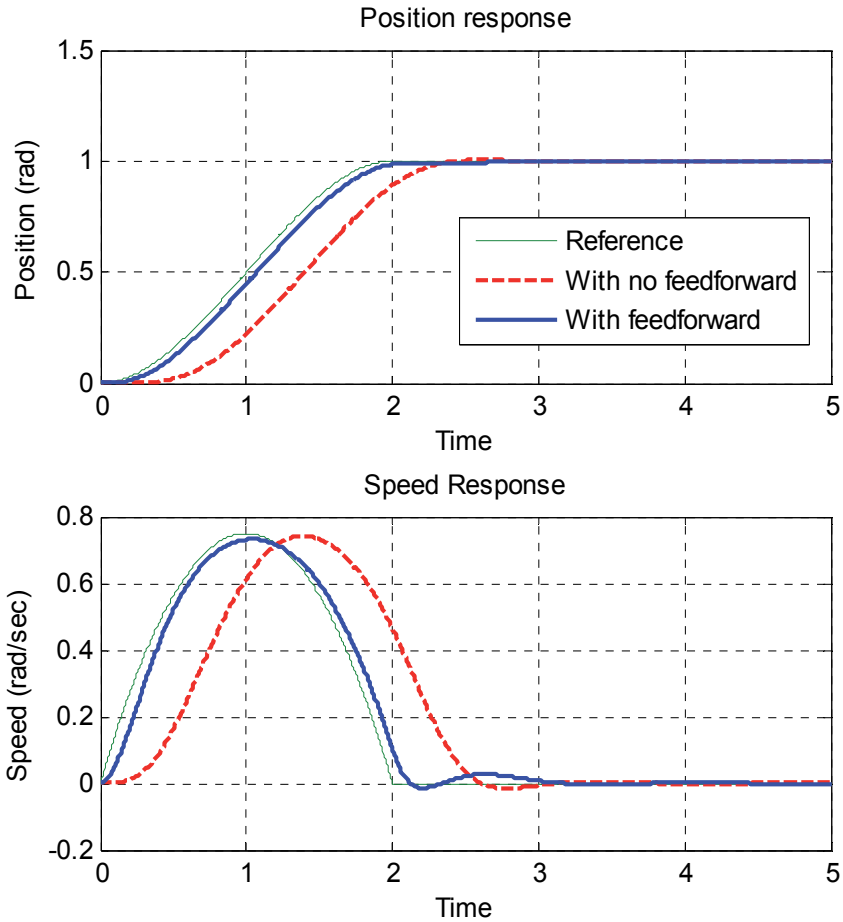


Figure 10. Illustration of a parabolic velocity move profile with and without feed-forward. (P = 12.6, I = 0.5, D = 4.25, Kv = 4.95, Ka = 0.85, T = 2).

6. Assessment and evaluation

The use of the interactive module, from the students' point of view, and how this contributes effectively to enhancing the learning of servo systems are important issues for the instructor. To this end, a qualitative assessment based on a focus group interview [24] and a traditional course survey were conducted by the instructor. Interviewing students constitutes an excellent way of assessing the use of the new learning module as well as learning outcomes. The interview consisted of discussing a set of open-ended questions with the students, raising two primary issues: the usability of the module and its effectiveness in learning. After the discussion, students were asked to give an evaluation (from 1 to 5) for each of the discussed questions. The assessment was conducted with two groups (a control engineering and robotics courses) during the second 2007 semester. In all, 34 students took

part in the interview. The students' questionnaire and corresponding evaluations are summarized in Table 1.

Nº.	Questions	score
1	The module is easy to use.	3.91 (0.79)
2	The module is productive and allows setting up simulations quickly.	3.56 (0.93)
3	The module is useful for complementing lectures.	3.88 (1.01)
4	The module gave me more motivation to learn.	3.79 (0.77)
5	Overall, the module helped me to learn servo systems.	3.88 (0.88)

Table 1. Student questionnaire. The table gives the mean and the standard deviation of students' responses. The grading scale is from 1 (low) to 5 (high).

Concerning the usability of the module, students were receptive to using the new tool. The majority agreed that it is easy to use and does not add cognitive load. They also agreed that the use of a simulation module with a GUI is much better than directly using Simulink blocks, since it helps them to proceed quickly. Students agreed that its most valuable feature is the sliders provided as a means to explore the effects of varying parameters on system behavior. The companion tutorial, with its guided exercises, is also considered important for the correct use of the module. In this aspect, students presumably limit their instructional material to the tutorials designed by the teacher.

The results from the question on whether the module is useful for complementing lectures showed that students consider that interactive tools represent a convenient way of enhancing in-class sessions and bringing new dimensions to traditional teaching methods. The majority considered that the tool presented constitutes a good complement for lectures, and many of them advocated extending the use of such modules to more topics and subjects.

Concerning learning outcomes, students uniformly agreed that the module allowed them to pay more attention to simulation results, rather than to designing Simulink diagrams. Some students declared that the interactive feature of the module activated their curiosity to discover its contents. As the interaction with the module is driven mainly by sliders, students can explore the effects of a full range of parameters by adjusting the values from low to high, or vice versa. In this aspect, the majority believed that this routine enables them to perceive clearly the qualitative aspects of servo systems. Finally, many students agreed that the tool helped them to understand servo systems problems, and many of them felt that the interactive module was an effective learning medium.

7. Conclusion

In this paper, the development of a Matlab/Simulink-GUI application for servo systems learning is described, and its use in the classroom is also addressed. The tool exposes students to servo systems concepts and allows them to experience a variety of practical scenarios. Its principal feature is that it dramatically reduces the effort needed to specify

simulations. It also allows students to explore the effects of varying parameters and instantly observe their influence on the system's behavior. Interaction with the module is straightforward and is done essentially using sliders. This sort of interaction proved to be helpful since it stimulates the students, and it is found that the interactive tool provides a stronger motivation to learn than does the traditional use of Simulink blocks. Finally, extensive use of the module can give students an enhanced intuition and help to them to gain better understanding of the qualitative aspects of servo systems.

Author details

Nourdine Aliane* and Javier Fernández Andrés
Dpto: Sistemas Informáticos Automática y Comunicaciones, Universidad Europea de Madrid (UEM), Villaviciosa de Odón, Madrid, Spain

Rafael Pastor Vargas
Dpto. Sistemas de Comunicación y Control, Escuela Técnica Superior Ingeniería Informática, UNED, Spain

Acknowledgement

The authors wish to thank the Spanish Ministry of Science and Innovation and the National Plan R&D TIN2008-06083-C03/TSI "s-Labs: Integration of open services for remote and distributed virtual laboratories, reusable and safe".

8. References

[1] D. S. Bernstein, The Quanser DC motor Control Trainer (2005), *IEEE Cont. Syst. Mag.*, vol. 25, no. 3, pp. 90-94.
[2] G. F Franklin, J. D. Powell, and A. Emami-Naeini, (2006), *A. Feedback Control of Dynamic* Systems. 5th ed, New Jersey: Prentice-Hall.
[3] F. L. Lewis, (1992), *Applied Optimal Control & Estimation: Digital Design & Implementation.* 1st ed, New Jersey: Prentice Hall.
[4] G. F. Franklin, J. D. Powell and M. L. Workman, (1997) *Digital Control of Dynamic Systems.* 3rd ed, Boston: Addison Wesley.
[5] R. N. Clark, (1997) *Control System Dynamics.* Cambridge: Cambridge University.
[6] B. Wittenmark, H. Hägglund, and M. Johansson, (1998), Dynamic pictures and interactive learning, *IEEE Cont. Syst. Mag.*, vol. 18, no. 3, pp. 26-32.
[7] M. Johansson, M. Gäfvert, and K. J. Åström, (1998), Interactive tools for education in automatic control, *IEEE Cont. Syst. Mag.*, vol. 18, no. 3, pp. 33-40.
[8] S. Dormido, S. Dormido-Canto, R. Dormido-Canto, J. Sanchez, and N. Duro, (2005) The role of interactivity in control learning," *Int. J. Eng. Educ.*, vol. 21, no. 6, pp. 1122-1133.

* Corresponding Author

[9] J. L. Guzmán, K. J. Åström, S. Dormido, T. Hägglund, and Y. Piguet, (2008), Interactive learning modules for PID control, *IEEE Cont. Syst. Mag.*, vol. 28, no. 5, pp. 118-134.
[10] T. Kikuchi, T. Kenjo and S. Fukuda, (2002), "Developing an educational simulation program for the PM stepping motor," *IEEE Trans. Educ.*, vol 45, no 1, pp-70-78.
[11] S. Ayasun and C. O. Nwankpa, (2005), Induction motor tests using Matlab/Simulink and their integration into undergraduate electric machinery courses, *IEEE Trans. Educ.*, vol 48, no 1, pp. 37–46.
[12] N. Patrascoiu, (2005), Modeling and Simulation of the DC Motor Using Matlab and LabVIEW," *Int. J. Eng. Educ.*, vol 21, no 1, pp. 49-54.
[13] D. Hercog and K. Jezernik, (2005), Rapid Control Prototyping using Matlab/Simulink and DSP-based Motor Control," *Int. J. Eng. Educ.*, vol. 21 no. 4, pp.596-605.
[14] MATLAB-Central-File-Exchange [on-line], Retrieved March 2012 from: http://www.mathworks.com/matlabcentral/fileexchange
[15] Casini, M. y D. Prattichizzo, (2003), The automatic control Telelab: A user-friendly interface for distance learning," *IEEE Trans. Educ.*, vol.46, no. 2, pp. 252-257.
[16] R. Pastor, C. Martín, J. Sánchez, and S. Dormido, (2005), Development of a XML-based lab for remote control experiments on a servo motor, *Int. J. Elect. Eng. Educ.*, vol. 42, nº 2, pp. 173-184.
[17] R. Kelly and J. Moreno, (2001), Learning PID structures in an introductory course of automatic control," *IEEE Trans. Educ.*, vol. 44, no. 4, pp. 373-376.
[18] LM628/LM629 Precision Motion Controller, [on-line] Retrieved March 2012 from: http://cache.national.com/ds/LM/LM628.pdf
[19] MAGELLAN Motion Control ICs, [on-line] Retrieved March 2012 from: http://www.pmdcorp.com/downloads/Magellan_PUG_v23_Feb_2008.pdf
[20] GALIL Controllers, [on-line] Retrieved March 2012 from: http://www.galilmc.com/about/index.html
[21] J. Apkarian and K. J. Astrom, (2004), A laptop servo for control education, *IEEE Cont. Syst. Mag.*, vol. 24, no. 5, pp-70-73.
[22] K. J. Åström and T. Hägglund, (2005), Advanced PID Control, ISA-Society.
[23] The Matlab interactive module for servo systems learning: [on-line]: http://www.esp.uem.es/aliane/servosystems/ss.zip
[24] B. M. Olds, B. M. Moskal and R. L. Miller, (2005), Assessment in Engineering Education: Evolution, Approaches, and Future Collaborations," *J. Eng. Educ.*, vol. 94, no. 1, pp. 13-25.

Chapter 17

Illustrating Amazing Effects of Modern Physics with Numerical Simulations Conducted in the Classroom

Eric Anterrieu

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46457

1. Introduction

Real experiments are sometimes missing or highly expensive to implement when teaching modern physics. This is an important issue, especially when the theories are not intuitive. However, with the incorporation of computers into the classroom it has become much easier to illustrate some amazing effects of modern physics that cannot be brought to the attention of students without the aid of numerical simulations. Thanks to the performance of MATLAB langage, as well as to the easiness to code a theoretical problem, the students can really conduct numerical expriments and play with them almost in real time. This computer-based teaching approach is illustrate here with the capabilities to use MATLAB for solving initial values problems of ordinary differential equations encountered in relativistic electrodynamics [2] as well as in non-linear optics [3].

2. Ordinary differential equations in Matlab

In mathematics, an ordinary differential equation (or ODE) is a relation f that contains one or more derivatives of a dependent variable y with respect to a single independent variable t, usually referred to as time. Many studies have been devoted to the solution of ODEs. In the case where the equation is linear, it can be solved by analytical methods. Unfortunately, most of the interesting ODEs are non-linear and, with a few exceptions, cannot be solved analytically. However, in science and in engineering, a numeric approximation to the solution is often good enough to solve a problem. Numerical integration is that part of numerical analysis which studies the numerical solutions of ODEs with the aid of iterative solvers.

MATLAB offers several numerical algorithms to solve a wide variety of ODEs. These solvers are designed to handle only explicit ODEs of the form $y' = f(t, y)$, linearly implicit ODEs of the form $M(t, y) \cdot y' = f(t, y)$ as well as fully implicit ones of the form $f(t, y, y') = 0$. Generally

there are many functions that satisfy a given ODE, and additional information is necessary to specify the solution of interest. In an initial value problem, the solution of interest satisfies a specific initial condition $y(t_o) = y_o$ at a given initial time t_o. Finally, the ODE solvers available in MATLAB accept only first-order ODEs. However, ODEs often involve derivatives of order higher than one. As a consequence, to use the ODE solvers for solving an ODE of order n, the equation has to be rewritten as an equivalent system of n first-order ODEs.

All the ODE solvers proposed in MATLAB share a common syntax that makes them easy to try if it is not apparent which one is the most appropriate. To apply a different method to the same problem, simply change the name of the ODE solver function. The simplest syntax, common to all the solver functions, is:

```
>> [t,y] = solver(odefun,tspan,y0,options);
```

where `solver` is one of the ODE solver functions available in MATLAB.

The input arguments are:

- `odefun`, a handle to a function that evaluates the ODE.
- `tspan`, a vector specifying the interval of integration.
- `y0`, a vector of initial conditions for the problem.
- `options`, a structure of optional parameters to change the default integration properties of the solver.

The output arguments contain the solution approximated at discrete points:

- `t`, a vector of time points.
- `y`, a solution array with the values of the solution y at the time t returned in `t`.

Beginning with initial condition, the solver steps through the time interval, computing a solution at each time step. If the solution for a time step satisfies the solver's error tolerance criteria, it is a successful step. Otherwise, it is a failed attempt; the solver shrinks the step size and tries again. The solver `ode45` is based on a RUNGE-KUTTA formula [6] and is according to MATLAB documentation "the best function to apply as a first try for most problems."

3. Non-linear optics

Ray optics is the branch of optics [4] in which all the subtle wave effects are neglected: the light is considered as travelling along rays which can only change their direction by refraction or reflection. The usefullness of the computer for studying, or designing, optical systems within an educational frame, is well known and it has been already suggested [10], but only in the paraxial approximation of geometrical optics. Indeed, when light propagates in media with constant refractive index, the SNELL-DESCARTES laws can be applied for implementing a fast numerical ray-tracing procedure based on a geometrical approach of the problem. However, when propagation takes place in media with non-homogeneous refractive index,

the differential equation governing the propagation of light has to be solved with the aid of the computer. This section describes a fast, accurate and easy to use code for illustrating the propagation of light in such media, sometimes with amazing effects that cannot be brought to the attention of students without the aid of numerical simulations, or except at an expensive cost.

3.1. Solving the iconal equation

The iconal equation [4], which describes the path of the light propagating in a medium with refractive index n, is an ODE of the second order:

$$\frac{\mathrm{d}}{\mathrm{d}s}\left(n\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}s}\right) = \nabla n, \tag{1}$$

where $\mathbf{r}$ denotes the position vector of a point on the ray and $\mathrm{d}s$ is an element of length along the path. From the point of view of the numerical implementation of a ray tracing procedure, it is more convenient to set $\mathrm{d}\ell = \mathrm{d}s/n$ so that (1) now reads:

$$\frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}\ell^2} = \frac{1}{2}\nabla n^2. \tag{2}$$

When light is propagating in a plane, the vector equation (2) reduces to a system of two scalar equations of the second order. We therefore have in Cartesian coordinates:

$$\begin{cases} \dfrac{\mathrm{d}^2x}{\mathrm{d}\ell^2} = \dfrac{1}{2}\dfrac{\partial n^2}{\partial x}, \\[2ex] \dfrac{\mathrm{d}^2y}{\mathrm{d}\ell^2} = \dfrac{1}{2}\dfrac{\partial n^2}{\partial y}. \end{cases} \tag{3}$$

As soon as the refractive index does not vary with x and y, (3) is straiforward to integrate and leads to a straight line path. When n is not uniform and according to the dependency of n with respect to x and y, it may not be possible to integrate (3) without the aid of the computer. This is why the capabilities of MATLAB to quickly solve an ODE is used here for studying the propagation of light in non-homogeneous media. However, since MATLAB can only solve ODEs of the first order, (3) has to be rewritten:

$$\begin{cases} \dfrac{\mathrm{d}p_x}{\mathrm{d}\ell} = \dfrac{1}{2}\dfrac{\partial n^2}{\partial x}, \\[2ex] \dfrac{\mathrm{d}p_y}{\mathrm{d}\ell} = \dfrac{1}{2}\dfrac{\partial n^2}{\partial y}, \end{cases} \quad \text{with} \quad \begin{aligned} \dfrac{\mathrm{d}x}{\mathrm{d}\ell} &= p_x, \\[2ex] \dfrac{\mathrm{d}y}{\mathrm{d}\ell} &= p_y. \end{aligned} \tag{4}$$

The MATLAB function `iconalODE` contains the final definition of the problem: in vector `f` are stored the values of x, y, p_x and p_y for a given value of ℓ, whereas the vector `df` returns the values of p_x, p_y, $\mathrm{d}p_x/\mathrm{d}\ell$ and $\mathrm{d}p_y/\mathrm{d}\ell$ for the same value of ℓ.

```
function df=iconalODE(l,f)
x = f(1);  px = f(3);
y = f(2);  py = f(4);
% gradient of n^2
[dn2dx, dn2dy] = dn2(x,y,param);
% ode
dpxdl = 0.5*dn2dx;
dpydl = 0.5*dn2dy;
df = [px; py; dpxdl; dpydl];
```

The students have to supply a function `dn2` which should return the Cartesian components of the gradient of n^2 at any point (x, y), the expression of which may depend on parameters stored in the `param` vector. The equation described in `iconalODE` is solved with the aid of the built-in function `ode45` provided by MATLAB:

```
>> [l,f] = ode45('iconalODE',l,fi);
>> X = f(:,1);
>> Y = f(:,2);
>> dYdX = f(:,4)./f(:,3);
```

The solver is fed with a vector `fi` which contains the initial settings of the problem, namely x_i, y_i, $(\mathrm{d}x/\mathrm{d}\ell)_i$ and $(\mathrm{d}y/\mathrm{d}\ell)_i$ at a given starting point M_i. The values of the solution, namely x, y, $\mathrm{d}x/\mathrm{d}\ell$ et $\mathrm{d}y/\mathrm{d}\ell$ along the ray path, are returned in the array `f`: here, for convenience reasons, vectors `X`, `Y` and `dYdX` are used for storing the values of x, y and dy/dx.

3.2. Application in non-homogeneous media

In this study, the refractive index of the non-homogeneous media satisfies the law:

$$n^2(\rho) = 1 + \frac{\rho_o^2}{\rho^2} \quad \text{with} \quad \rho = \sqrt{x^2 + y^2}. \tag{5}$$

Another choice would have been to study the propagation of light in LUNEBURG lens [9], however contrary to expression (5) the refractive index would have not present a singularity and consequently the example would have been less constraining from the computational point of view and less illustrative from the point of view of the propagation of light in non-homogeneous media. However, according to the approach adopted here, moving from one study to another just requires to change the lines of code written in the function `dn2` supplied by the students: this is exactly what is done in a classroom.

Coming back to (5), the region where the media is non-homogeneous is restricted to a disk $\mathcal{D}$ with radius R. Outside $\mathcal{D}$, n is supposed to be uniform and equal to

$$n_i = \sqrt{1 + \rho_o^2/R^2}, \tag{6}$$

so that no discontinuity occurs at the boundary. According to (5), within $\mathcal{D}$ the components of the gradient of n^2 in Cartesian coordinates are:

$$\begin{aligned} \frac{\partial n^2}{\partial x} &= -\frac{2x\rho_0^2}{(x^2+y^2)^2}, \\ \frac{\partial n^2}{\partial y} &= -\frac{2y\rho_0^2}{(x^2+y^2)^2}, \end{aligned} \tag{7}$$

whereas they are null outside $\mathcal{D}$ since n is there uniform and equal to n_i given by (6). The function `dn2` is therefore reduced to:

```
function [dn2dx,dn2dy]=dn2(x,y,param)
Ro = param(1);
R  = param(2);
if (sqrt(x^2+y^2) > R)
    dn2dx = 0;
    dn2dy = 0;
else
    dn2dx = -2*x*Ro^2/(x^2+y^2)^2;
    dn2dy = -2*y*Ro^2/(x^2+y^2)^2;
end
```

where `Ro` is used for storing the value of parameter ρ_o and `R` that of the radius R of $\mathcal{D}$.

Before solving the ODE described in functions `iconalODE` and `dn2` it is necessary to set the initial conditions of the problem. Let us consider an incident ray starting at a point $M_i(x_i, y_i)$ and making an angle α with the Ox axis. In the neighbourhood of M_i, we have $\mathrm{d}x = \mathrm{d}s\cos\alpha$ and $\mathrm{d}y = \mathrm{d}s\sin\alpha$. Since at any point on the ray path $\mathrm{d}\ell = \mathrm{d}s/n$, we therefore have:

$$(\mathrm{d}x/\mathrm{d}\ell)_i = n_i\cos\alpha \quad \text{and} \quad (\mathrm{d}y/\mathrm{d}\ell)_i = n_i\sin\alpha.$$

Finally, the radius of $\mathcal{D}$ is set, for example, to $R = 4\rho_o$ with $\rho_o = 1$. Since the system exhibits a central symmetry, the study is restricted here to incidents rays parallel to the Ox axis, so that $\alpha = \pi$, and starting from M_i located at a distance $y_i = h = 2\rho_o$ from the Ox axis and with $x_i = 6\rho_o$. The following lines are the MATLAB code corresponding to all these initial settings.

```
>> Ro = 1;
>> R = 4*Ro;
>> h = 2*Ro;
>> Xi = 6*Ro;
>> Yi = h;
>> alpha = pi;
>> Ni = sqrt(1 + (Ro/R)^2);
>> dXidl = Ni*cos(alpha);
>> dYidl = Ni*sin(alpha);
>> fi = [Xi Yi dXidl dYidl];
```

After point M_i, the light travels according to the ODE described in the functions `iconalODE` and `dn2`. The complete path of the ray returned by the `ode45` solver is represented on Figure 1. It corresponds to the following lines:

```
>> figure(1);
>> plot(X/Ro,Y/Ro,'r-');
```

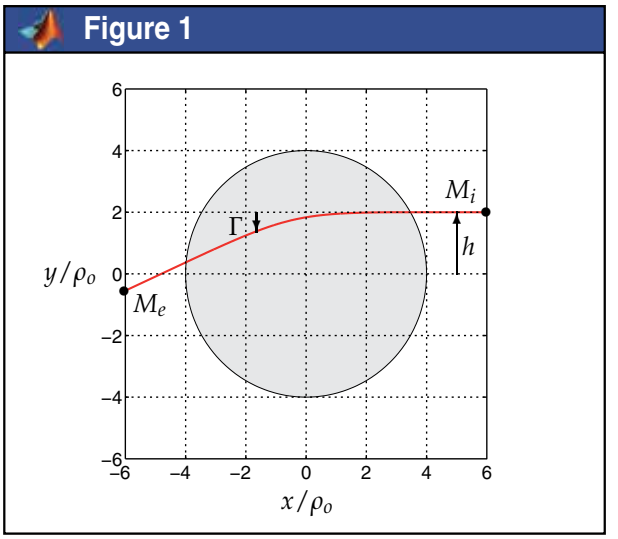


Figure 1. An example of a ray path (in red) between points M_i and M_e. Outside the disk $\mathcal{D}$ (in grey) the media is homogeneous with a constant refractive index: the light is travelling in straight line. On the contrary, the media in $\mathcal{D}$ is non-homogeneous with spatially varying refractive index: the path of the light is curved.

As expected, outside the disk $\mathcal{D}$, the light is travelling in straight line since the media is homogeneous with uniform refractive index given by (6). On the contrary, within $\mathcal{D}$ the path is curved since the light is here travelling in a non-homogeneous media with spatially varying refractive index according to (5). The curvature of the path is oriented towards the center of the disk $\mathcal{D}$, that is to say towards the direction of the gradient of n. The ray continuously tends towards the center without reaching it. According to the inverse return of light, the path of the light obtained when solving the iconal equation (1) from M_i to M_e and that obtained when solving it from M_e to M_i in the reverse direction of propagation are nothing but the same. In addition, owing to the central symmetry of the system, the path of the light is symmetrical with respect to the location where the distance from the center of $\mathcal{D}$ is minimal.

Coming back to the situation of Figure 1, the deviation angle Γ after travelling from M_i through the non-homogeneous media can be computed at any point M_e in the homogeneous media according to:

$$\Gamma = \arctan(\mathrm{d}y/\mathrm{d}x)_e + \pi$$

that is to say in MATLAB language:

```
>> gamma = atan(dYdX(end))+pi
```

As observed by the students, h is playing the role of an impact parameter with respect to the singularity of the refractive index at the center of the disk $\mathcal{D}$ and the angle Γ that of a scattering angle with regards to the direction of the incoming ray. Shown in Figure 1, for $h = 2\rho_o$, the deviation is about 25° with respect to the incident direction.

An interesting and convenient aspect of conducting numerical simulations with the aid of the computer in a classroom with MATLAB is the capability to re-run the code with different initial settings and to obtain the new path of the light almost immediately. This is exactly what is done by the students for investigating the influence of the parameter h on the deviation angle Γ. As shown in Figure 2, Γ varies with h in a non-linear manner. Values of Γ greater than $\pi/2$ correspond to rays which make a half turn, or even more than a complete turn, before leaving the non-homogeneous media.

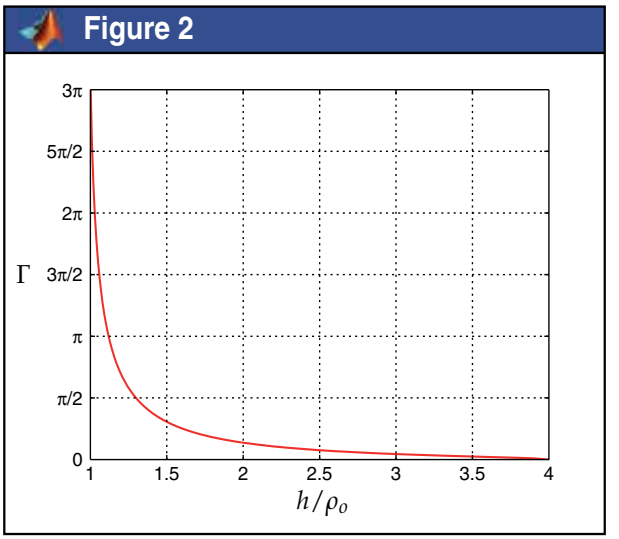


Figure 2. Variations of the deviation (or scattering) angle Γ with the (impact) parameter h.

Some particular situations are shown in Figure 3 for different values of the ratio h/ρ_o. Although trajectories for large values of Γ are not without surprise for the students, they usually accept that the light can make a large number of turns before leaving the disk $\mathcal{D}$. However, their reaction is simply incredible when they discover, even with numerical experiences, that below a limit for the ratio h/ρ_o the path of the light identifies to that of a spiral: the ray seems to be attracted by the singularity of the refractive index of the non-homgeneous media for $\rho = 0$ and does not emerge from the disk $\mathcal{D}$. Such an amazing situation is represented in Figure 4 for $h = 0.965\,\rho_o$.

Coming back to the analogy between h and an impact parameter and between Γ and a scattering angle, these trajectories remind those of a particule captured by a KEPLER or COULOMB potential [5].

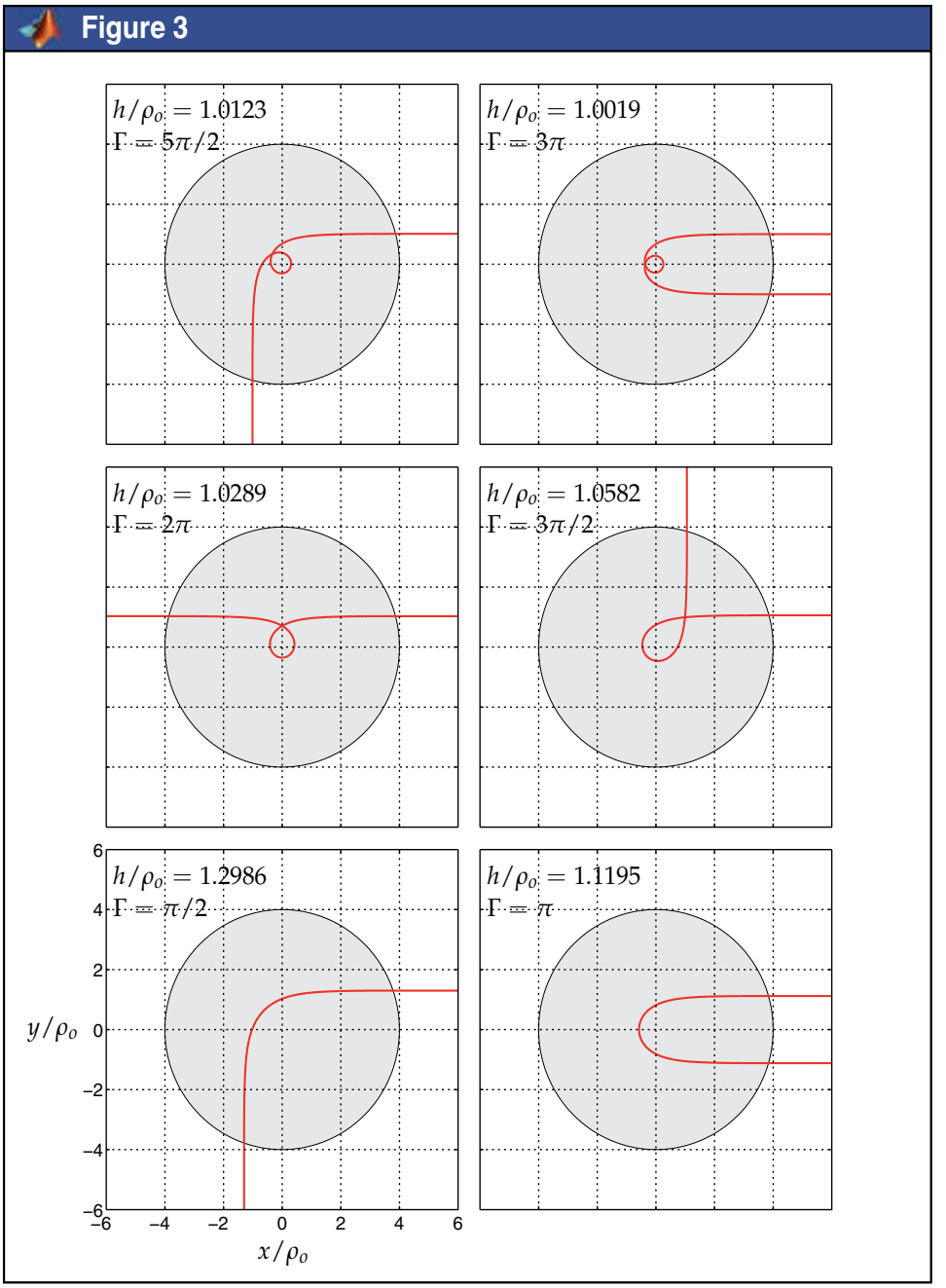


Figure 3. Some light trajectories for various values of h. According to the inverse return of light, all the trajectories are valid solutions for propagation in both directions.

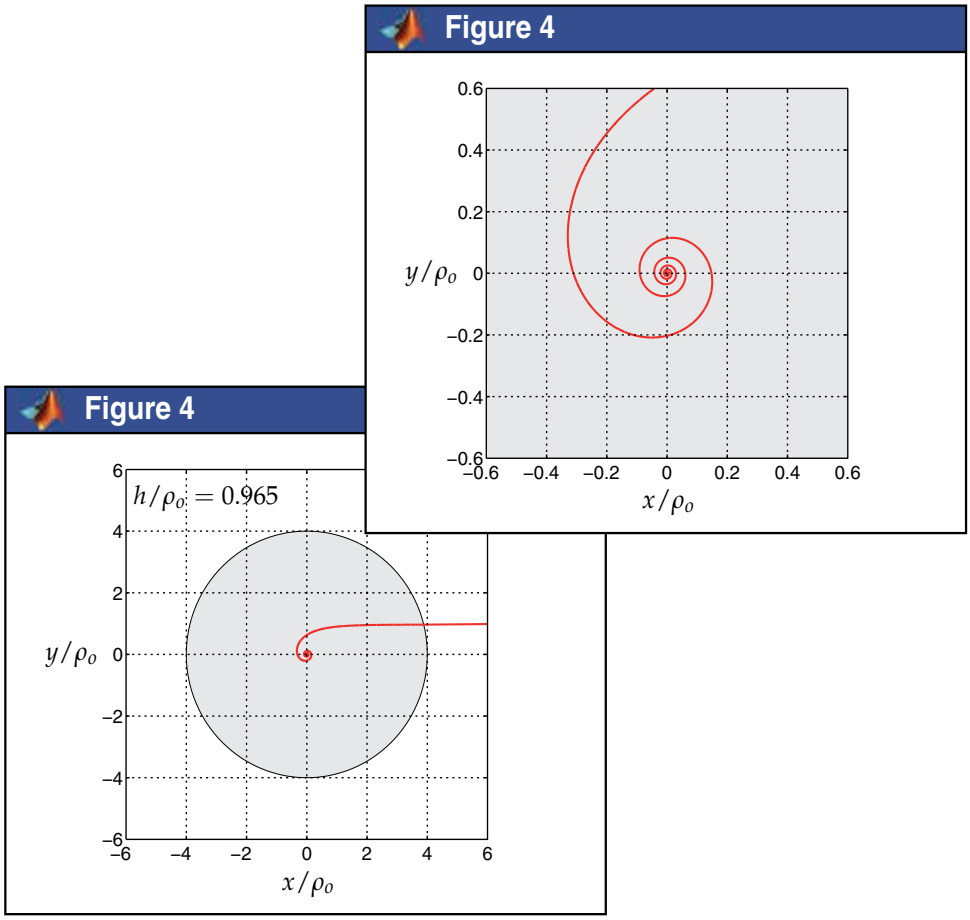


Figure 4. Ray path for $h = 0.965\,\rho_o$ (central part is enlarged on the top-left).

4. Relativistic electrodynamics

Classical electrodynamics is an important branch of the theoretical physics [7], which has numerous applications in physical engineering. On one hand, the usefulness of the computer for studying the movement of particles in electromagnetic fields is now well known and it has been already suggested [11], but only in the Newtonian approximation. However, the trajectory becomes complicated to establish when the electric and magnetic fields are acting together, whatever the initial velocity. On the other hand, the teaching of relativity to undergraduate electrical engineers appears also to be a necessity [8]. A stimulated way to gain the attention and the interest of students for relativity, which is not an intuitive theory, is precisely to begin by studying the movement of fast particles in electromagnetic fields because the applications are numerous and surprising. This section describes a fast, accurate and easy to use code for computing the trajectories of charged relativistic particles under the action of the LORENTZ force and illustrating some effects of relativity that cannot be brought to the attention of students without the aid of numerical simulations, or except at an expensive cost.

4.1. Solving the movement equation

The equation which describes the trajectory of a relativistic particle of mass m and charge q under the simultaneous actions of an electric field $\mathbf{E}$ and a magnetic one $\mathbf{B}$ is an ordinary differential equation (ODE) of the first order [7]:

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}), \tag{8}$$

where $\mathbf{v}$ is the velocity of the particle, $\mathbf{p} = \gamma\, m\mathbf{v}$ is the relativistic expression of the momentum with $\gamma = 1/\sqrt{1 - \mathbf{v}^2/c^2}$ the relativistic factor and c is the speed of light. In Cartesian coordinates, the previous vector equation can be put in the form of a system of three scalar ODEs of the first order:

$$\begin{cases} \dfrac{\mathrm{d}p_x}{\mathrm{d}t} = q(E_x + v_y B_z - v_z B_y) \\[2ex] \dfrac{\mathrm{d}p_y}{\mathrm{d}t} = q(E_y + v_z B_x - v_x B_z) \\[2ex] \dfrac{\mathrm{d}p_z}{\mathrm{d}t} = q(E_z + v_x B_y - v_y B_x) \end{cases} \tag{9}$$

Owing to the factor γ, solving this system of ODEs is harder in the relativistic case than in the Newtonian one. Indeed, expression of the first derivatives of the components p_x, p_y et p_z of the momentum $\mathbf{p}$ is not simple because of the expression of the first derivative of γ itself. It is necessary to express the speed $\mathbf{v}$ as a function of the momentum $\mathbf{p}$ without involving the LORENTZ factor γ. This is done by introducing the energy $\mathcal{E}$ of the free particle:

$$\mathbf{v} = \frac{\mathbf{p}}{\gamma\, m} = \frac{\mathbf{p}c^2}{\mathcal{E}} = \mathbf{p}\frac{c^2}{\sqrt{\mathbf{p}^2c^2 + m^2c^4}}. \tag{10}$$

This vector equation reduces again to three scalar equations in Cartesian coordinates:

$$\begin{cases} v_x = p_x c^2 / \sqrt{(p_x^2 + p_y^2 + p_z^2)c^2 + m^2c^4} \\[2ex] v_y = p_y c^2 / \sqrt{(p_x^2 + p_y^2 + p_z^2)c^2 + m^2c^4} \\[2ex] v_z = p_z c^2 / \sqrt{(p_x^2 + p_y^2 + p_z^2)c^2 + m^2c^4} \end{cases} \tag{11}$$

Finally, solving (8) in Cartesian coordinates in the relativistic case is equivalent to solve the system of ODEs (9) together with (11). However, owing to the coupling between these equations, it may not be possible to integrate them without the aid of the computer. This is why the capabilities of MATLAB to quickly solve ODEs is used here for tackling this problem.

The MATLAB function `odeEB` contains the definition of the system to solve: in vector `f` are stored the coordinates (x, y, z) of the particle as well as the components (p_x, p_y, p_z) of the momentum $\mathbf{p}$ at a given time t, whereas the vector `df` returns the components $\mathrm{d}x/\mathrm{d}t$, $\mathrm{d}y/\mathrm{d}t$ and $\mathrm{d}z/\mathrm{d}t$ of the speed vector $\mathbf{v}$ as well as the components of the first derivative of the momentum $\mathrm{d}p_x/\mathrm{d}t$, $\mathrm{d}p_y/\mathrm{d}t$ and $\mathrm{d}p_z/\mathrm{d}t$ at the same time.

```
function df=odeEB(t,f,q,m)
% particle coordinates
x  = f(1);
y  = f(2);
z  = f(3);
% particle momentum
Px = f(4);
Py = f(5);
Pz = f(6);
% particle energy
P = sqrt(Px.^2+Py.^2+Pz.^2);
E = sqrt(P^2*c^2+m^2*c^4);
% particle velocity
Vx = Px*c^2/E;
Vy = Py*c^2/E;
Vz = Pz*c^2/E;
% electric and magnetic fields
[Ex,Ey,Ez]=fieldE(x,y,z,t);
[Bx,By,Bz]=fieldB(x,y,z,t);
% ode
dPxdt = q*(Ex + Vy*Bz - Vz*By);
dPydt = q*(Ey + Vz*Bx - Vx*Bz);
dPzdt = q*(Ez + Vx*By - Vy*Bx);
df = [Vx; Vy; Vz; dPxdt; dPydt; dPzdt];
```

This function is general enough for solving various problems without re-writing any code since the mass m and the electric charge q of the particle are given to the function. The students have only to supply the two functions `fieldE` and `fieldB` which should return the Cartesian components of $\mathbf{E}$ and $\mathbf{B}$ at any point (x, y, z) and at any time t. The equation described in `odeEB` is solved again with the aid of the built-in function `ode45`:

```
>> [t,f] = ode45('odeEB',t,fi);
```

The solver is fed with a vector `fi` which contains the initial settings of the problem, namely the initial position of the particle $(x, y, z)_i$ as well as the initial components of the momentum $(p_x, p_y, p_z)_i$ at a given starting point M_i. The values of the solution, namely the successive positions of the particle (x, y, z) along the trajectory as well as the components (p_x, p_y, p_z) of the momentum $\mathbf{p}$ at the corresponding time, are returned in the array `f`.

```
>> x = f(:,1);  Px = f(:,4);
>> y = f(:,2);  Py = f(:,5);
>> z = f(:,3);  Pz = f(:,6);
```

Here, for convenience reasons, vectors `x`, `y` and `z` are used for storing the values of x, y and z. Likewise, vectors `Px`, `Py` and `Pz` are used for storing the values of p_x, p_y and p_z.

```
>> P = sqrt(Px.^2+Py.^2+Pz.^2);
>> E = sqrt(P.^2*c^2+m^2*c^4);
>> Vx = Px*c^2./E;
>> Vy = Py*c^2./E;
>> Vz = Pz*c^2./E;
```

The final step performed outside the solver is the computation of the components (v_x, v_y, v_z) of the speed vector **v**, according to (10). The corresponding values are written in vectors `Vx`, `Vy` and `Vz`.

4.2. Applications in uniform fields

The applications presented in this section are concerned with the movement of a relativistic electron under the simultaneous action of an electric field **E** and a magnetic one **B**.

```
>> c =  2.99792458E+08; % m/s
>> q = -1.60217649E-19; % C
>> m =  9.10938215E-31; % kg
```

Depending on the orientation of **E** with respect to that of **B**, the trajectory of the electron may be very different.

4.2.1. **E** *and* **B** *collinear*

In this first case, **E** and **B** are collinear, oriented along the Oz axis, $\mathbf{E} = E\,\mathbf{e}_z$ and $\mathbf{B} = B\,\mathbf{e}_z$, and uniform with intensities E and B set to $3 \cdot 10^6$ V/m and 0.05 T, respectively. The functions `fieldE` and `fieldB` supplied by the students are therefore reduced to:

```
function [Ex,Ey,Ez]=fieldE(x,y,z,t)
% cartesian components of electric field
Ex = 0;
Ey = 0;
Ez = 3.00E+06;
```

and

```
function [Bx,By,Bz]=fieldB(x,y,z,t)
% cartesian components of magnetic field
Bx = 0;
By = 0;
Bz = 0.05;
```

The electron is injected in the region where the electromagnetic field is applied from a point M_i with an initial speed vector $\mathbf{v}_i = v_i\mathbf{e}_x$ perpendicular to **E** and **B** and a velocity $v_i = 2\,c/3$. The coordinates of M_i are chosen in such a way that the trajectory rolls up along the Oz axis: $x_i = 0$, $y_i = \rho_i$ and $z_i = 0$, for example, where $\rho_i = \gamma_i m v_i / qB$ is the initial radius of curvature. We therefore have the following initial conditions:

```
>> Vi = 2*c/3;
>> Pxi = gamma(Vi)*m*Vi;
>> Pyi = 0;
>> Pzi = 0;
>> Xi = 0;
>> Yi = Pxi/(q*0.05);
>> Zi = 0;
```

After point M_i, the electron is under the influence of **E** and **B**: its trajectory is the solution of the ODE described in function `odeEB` with the previous initial conditions. The complete path returned by the `ode45` solver is represented on Figure 5:

```
>> figure(5);
>> plot3(x,y,z,'r-');
```

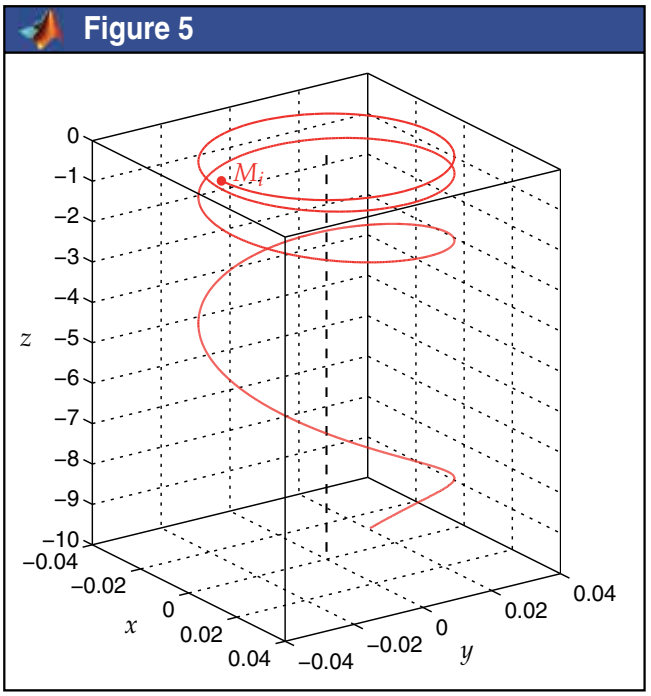


Figure 5. Trajectory of a relativistic electron in an electromagnetic field when **E** and **B** are collinear: $\mathbf{E} = E\,\mathbf{e}_z$, $\mathbf{B} = B\,\mathbf{e}_z$ and $\mathbf{v}_i = v_i\mathbf{e}_x$ with $v_i = 2\,c/3$.

The trajectory is that of a circular helix with a variable step which rolls anticlockwise (from Ox to Oy) along the Oz axis (in the opposite direction of the **E** because here $q < 0$).

According to the LORENTZ force law (8), a charged particle can gain (or lose) energy from an electric field **E**, but not from a magnetic field **B**. Indeed, by definition the magnetic force $q\,\mathbf{v} \times \mathbf{B}$ is always perpendicular to the particle's direction of motion and therefore does not

work on the particle. Consequently, as expected from the theory of relativity, the students can easily observed on Figure 6 that the velocity of the particle $v = (v_x^2 + v_y^2 + v_z^2)^{1/2}$ along the trajectory increases rapidly (under the only action of the electric field **E**) from $v_i = 2\,c/3$ up to the very limit c:

```
>> figure(6);
>> plot(t/1E-09,sqrt(Vx.^2+Vy.^2+Vz.^2)/c,'r-');
```

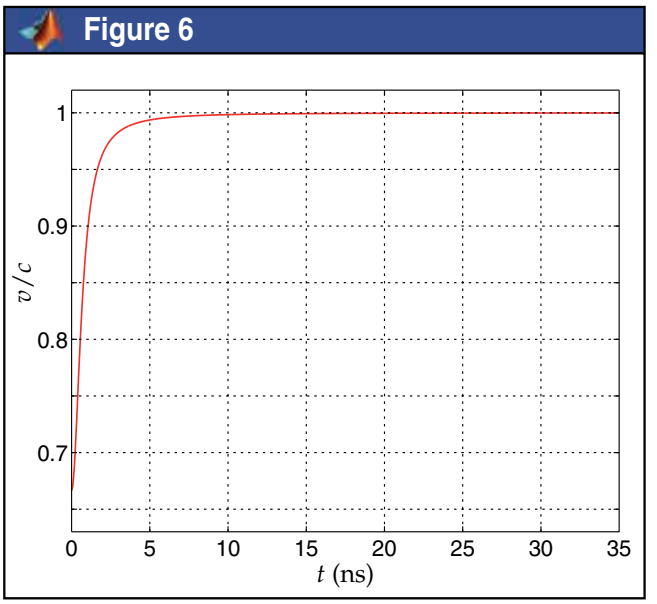


Figure 6. Relative velocity v/c of a relativistic electron along the trajectory shown in Figure 5.

4.2.2. **E** *and* **B** *perpendicular*

Now **E** and **B** are perpendicular to each other, $\mathbf{E} = E\,\mathbf{e}_x$ and $\mathbf{B} = B\,\mathbf{e}_y$, and uniform with intensities E and B set to $3 \cdot 10^6$ V/m and 0.02 T, respectively. The functions `fieldE` and `fieldB` are modified accordingly by the students:

```
function [Ex,Ey,Ez]=fieldE(x,y,z,t)
% cartesian components of electric field
Ex = 3.00E+06;
Ey = 0;
Ez = 0;
```

and

```
function [Bx,By,Bz]=fieldB(x,y,z,t)
% cartesian components of magnetic field
```

```
Bx = 0;
By = 0.02;
Bz = 0;
```

The electron is injected in the region where the electromagnetic field is applied from a point M_i with a initial speed vector $\mathbf{v}_i = v_i \mathbf{e}_z$ perpendicular to both $\mathbf{E}$ and $\mathbf{B}$ and a velocity $v_i = 2c/3$. The complete path returned by the `ode45` solver is represented on Figure 7. The trajectory is now confined to a plan perpendicular to $\mathbf{B}$ and containing both $\mathbf{E}$ and the speed vector $\mathbf{v}$. More exactly it is a sinusoidal curve with amplitude and period depending on $\mathbf{v}_i$, $\mathbf{E}$ and $\mathbf{B}$. Indeed, only the magnetic force can periodically return the particle since $q\,\mathbf{v} \times \mathbf{B}$ operates in the $0xz$ plan sometimes in a direction, sometimes in the other one, depending on the direction of $\mathbf{v}$. On the other hand, the electric force $q\,\mathbf{E}$ also operates in this plan but always in the same direction, namely $-\mathbf{e}_x$ because here $q < 0$.

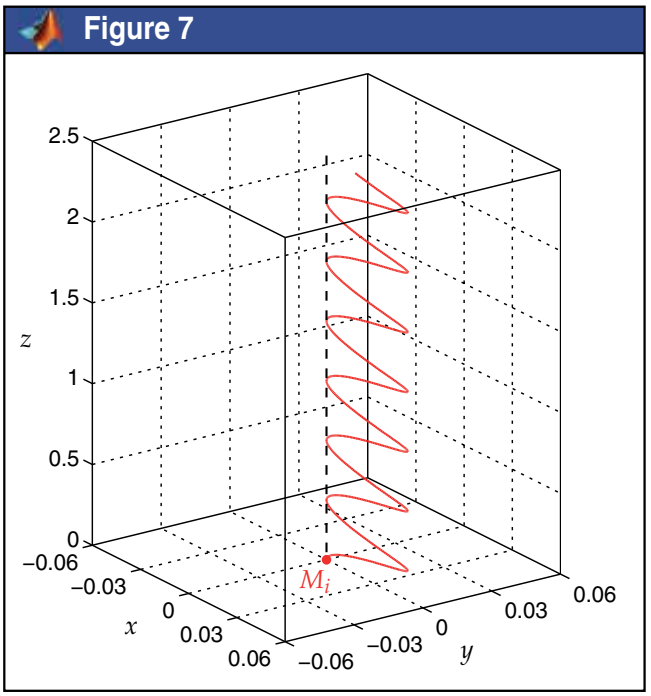


Figure 7. Trajectory of a relativistic electron in an electromagnetic field when $\mathbf{E}$ and $\mathbf{B}$ are perpendicular: $\mathbf{E} = E\,\mathbf{e}_x$, $\mathbf{B} = B\,\mathbf{e}_y$ and $\mathbf{v}_i = v_i \mathbf{e}_z$ with here $v_i = 2c/3$.

An interesting and convenient aspect of conducting numerical simulations with the aid of the computer in a classroom with MATLAB is the capability to re-run the code with different initial settings and to obtain the new trajectory almost immediately. As observed by the students in such a situation, as soon as the value of v_i becomes closer to the ratio E/B, the amplitude of the oscillations decreases as illustrated in Figure 8. When $v_i = E/B$, the actions of the electric and magnetic forces offset each other so that the trajectory reduces to a straight line. Such

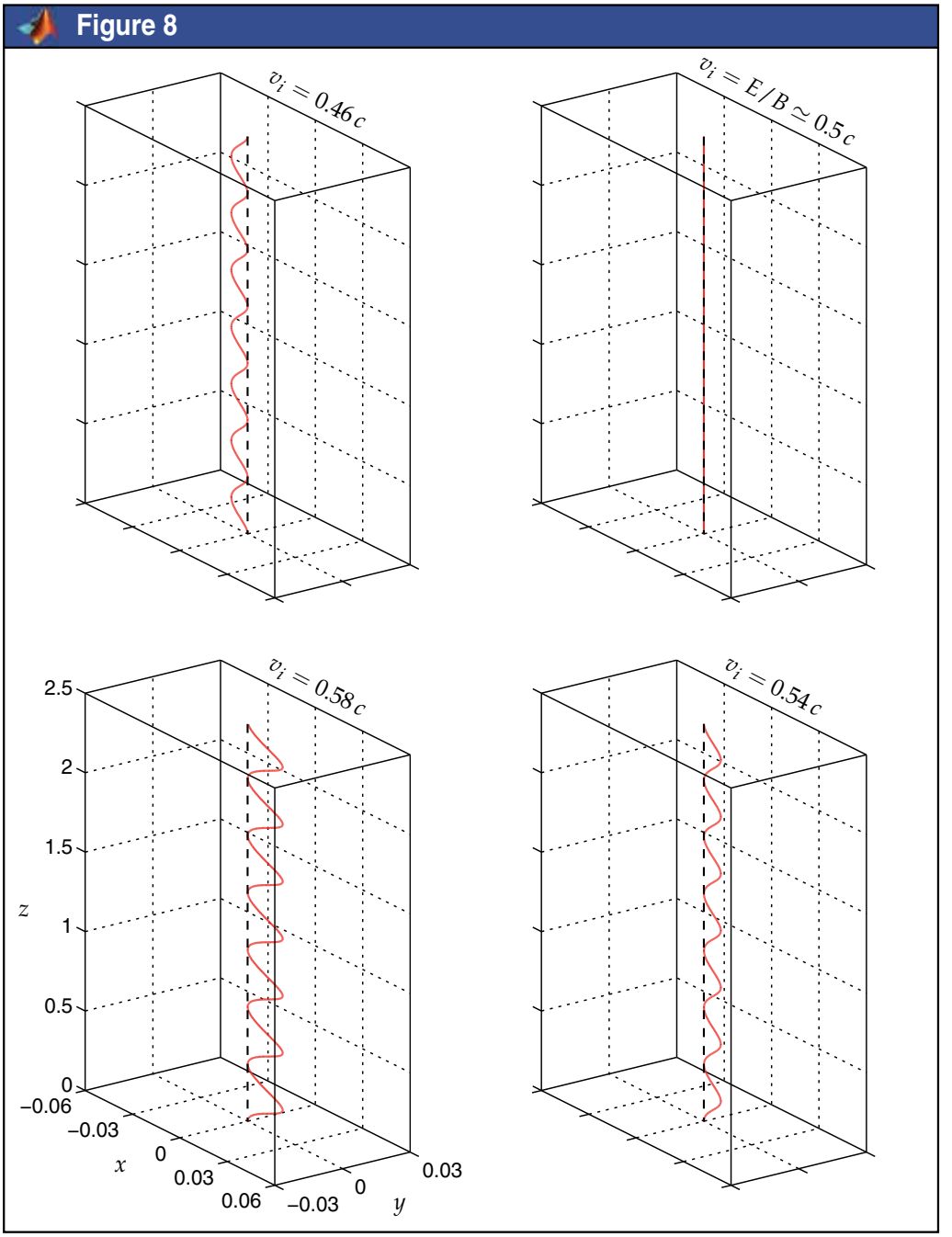


Figure 8. Same as Figure 7 for various initial velocities v_i: when $v_i = E/B$, the trajectory of the electron identifies to that of a straight line.

an amazing situation is encountered in a WIEN filter [12] which is a device used for filtering charged particles with a given velocity (or energy), for example in electron microscopes or in spectrometers.

4.3. Applications in time/space varying fields

In the previous applications, the students dealt with uniform electromagnetic fields. Since the time and space coordinates are given to the functions `fieldE` and `fieldB`, some time and/or space variations in the intensity E and B of the electric and magnetic fields can be easily introduced. This is exactly what is done by the students in the classroom.

Here, the particle is under the lonely action of a static but non-uniform magnetic field **B**, the components of which are:

$$\begin{cases} B_x = -B_o \dfrac{xz}{z_o^2} \\ B_y = -B_o \dfrac{yz}{z_o^2} \\ B_z = B_o(1 + \dfrac{z^2}{z_o^2}) \end{cases} \tag{12}$$

where B_o and z_o are taken equal to 0.05 T and 1 m, respectively. The functions `fieldE` and `fieldB` are modified accordingly by the students to reflect this new experimental setup.

The initial conditions of the problem are now the following ones. The electron is injected in the region where the magnetic field is applied from a point M_i with an initial speed vector $\mathbf{v}_i = v_i(\sin\alpha_i \mathbf{e}_x + \cos\alpha_i \mathbf{e}_z)$ with a velocity $v_i = 2\,c/3$ and $\alpha_i = 20°$, for example. The coordinates of M_i are chosen in such a way that the trajectory rolls along the Oz axis. The complete path returned by the `ode45` solver is shown in Figure 9. The trajectory reminds that of a circular helix with a fixed step but with a variable radius of curvature ρ which raises its maximum for $z = 0$ where B_z raises its minimum and where $B_x = B_y = 0$.

Shown in Figure 10 are the variations of the velocity of the particle $v = (v_x^2 + v_y^2 + v_z^2)^{1/2}$ along the trajectory as well as those of the axial and radial components $|v_z|$ and $v_\rho = (v_x^2 + v_y^2)^{1/2}$ of the speed vector **v**:

```
>> figure(10);  hold('on');
>> plot(t/1E-09,abs(Vz)/c,'b-');
>> plot(t/1E-09,sqrt(Vx.^2+Vy.^2)/c,'g-');
>> plot(t/1E-09,sqrt(Vx.^2+Vy.^2+Vz.^2)/c,'r-');
```

Contrary to what is observed in Figure 6 where the particule is moving under the simultaneous action of an electric field **E** and a magnetic one **B**, here the velocity v of the particle is constant and equal to v_i since it is under the lonely action of a (non-uniform) magnetic field **B** which cannot give (or take) energy to (or from) the particle because it does not work.

Coming back to the trajectory observed in Figure 9 in the light of Figure 10, as soon as B_z grows from $z = 0$, the particle draws circles around the Oz axis which become continually

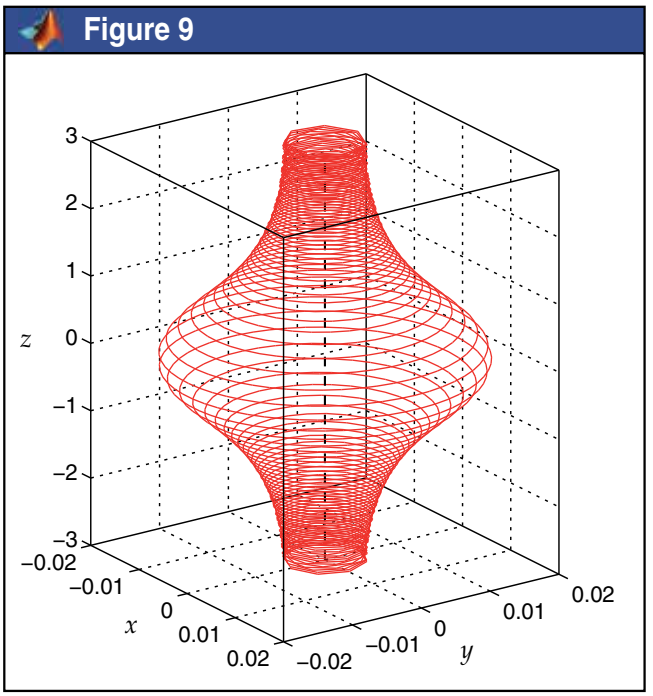


Figure 9. Trajectory of a relativistic electron in the non-uniform magnetic field described in (12).

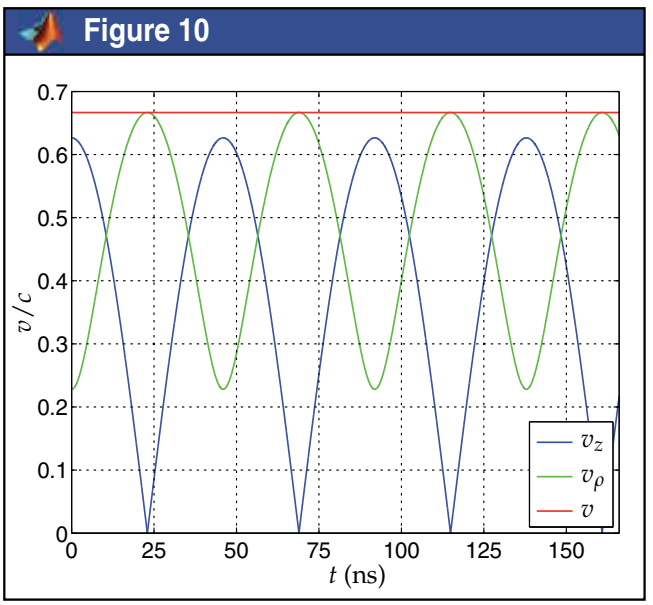


Figure 10. Relative velocity v/c of a relativistic electron along the trajectory shown in Figure 9.

smaller and closer. At the same time, the axial contribution to the kinetic energy is converted into a rotational one up to v_z is cancelled out and v_ρ reach its maximum (v remaining constant and equal to v_i). Then, the particle turns and goes back, drawing again circles in the opposite direction which become larger and more distant up to $z = 0$. The particle has been reflected by the magnetic field $\mathbf{B}$. Such an experimental setup is called a magnetic mirror [1]. It is encountered, for example, in a magnetic bottle which is the superposition of two magnetic mirrors used together for confinement purpose.

5. Conclusion

This contribution has described MATLAB codes for solving ordinary differential equations that cannot be solved without the aid of the computer in the field of non-linear optics as well as in relativistic electrodynamics. Since these two domains of modern physics are associated to non intuitive theories, concrete illustrations are welcomed to gain the attention and the interest of the students. The code is fast, accurate and easy to use. It has to be fed with few functions supplied by the students for switching from one study to another, the initial conditions of the problem being changed accordingly. Since the results are obtained almost in real time, these initial settings can be changed at will so that an interactive study is often conceivable with the computers in a classroom. Numerical examples have demonstrated the capabilities of these codes for illustrating in teaching conditions some amazing effects of modern physics that cannot be brought to the attention of students without the aid of the computer, or at an expensive cost. In a classroom setting, the time to be allocated between the underground physics, the modeling issues and the freewheel experimentation is of course left to the teacher and it may vary from one student to another.

Author details

Eric Anterrieu
National de la Recherche Scientifique & Université de Toulouse, France

6. References

[1] Allen R.J. (1962). A demonstration of the magnetic mirror effect, *American Journal of Physics*, Vol. 30, No. 12, pages 867-869, 1962.

[2] Anterrieu E. & Pérez J.-P. (2011). How to make attractive the teaching of relativistic electrodynamics?, *Proceedings of IEEE International Conference on Computer as a Tool*, pages 1-4, Lisbon (Portugal), 27-29 April 2011.

[3] Anterrieu E. & Pérez J.-P. (2010). Illustrating amazing effects of optics with the computer, *Proceedings of IEEE Engineering Education Conference*, pages 180-185, Madrid (Spain), 14-16 April 2010.

[4] Born M. & Wolf E. (1999). *Principles of optics*, 7th edition, Cambridge University Press, Cambridge.

[5] Boyer T.H. (2004). Unfamiliar trajectories for a relativistic particle in a Kepler or Coulomb potential, *American Journal of Physics*, Vol. 72, No. 8, pages 992-997, 2004.

[6] Dormand J.R. & Prince P.J. (1980). A family of embedded Runge-Kutta formulae, *Journal of Computational and Applied Mathematics*, Vol. 6, No. 1, pages 19-26, 1980.
[7] Feynman R.P. (2005). *The Feynman lectures on physics: the definitive and extended edition*, 2nd edition, Addison Wesley, Boston.
[8] Hoefer W.J.R. (1973). On teaching relativity to undergraduate electrical engineers, *IEEE Transactions on Education*, Vol. 16, No. 3, pages 152-156, 1973.
[9] Luneburg R.K. (1964). *Mathematical theory of optics*, 1st edition, University of California Press, Berkeley.
[10] Petouris A., Humphries J., Russell P., Vourdas A. & Jones G.R (1994). Computer aided design for teaching modern optics to electronic engineering undergraduates, *IEE Proceedings Science, Measurement & Technology*, Vol. 141, No. 3, pages 171-176, 1994.
[11] Sader U. & Jodl H.J. (1987). Computer-aided physics-particles in electromagnetic fields, *European Journal of Physics*, Vol. 8, No. 2, pages 88-92, 1987.
[12] Wien W. (1898). Untersuchungen über die electrische entladung in verdünnten gasen, *Annalen der Physik*, Vol. 301, No. 6, pages 440-452, 1898.

Chapter 18

The MatLab™ Software Application for Electrical Engineering Simulations and Power System

Leonardo da S. Lessa, Afonso J. Prado, Rodrigo Cleber Silva, Sérgio Kurokawa, Luiz F. Bovolato and José Pissolato Filho

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/48555

1. Introduction

Transmission lines are the elements of the electric power system that connect the load to the generation joining the production facilities of energy over large geographic areas. You could say that the transmission of electricity is one of the most important contributions that engineering has offered to the modern civilization.

The distribution of current potential differences and the transfer of energy along a transmission line can be analyzed by various methods, and it is expected that all lead to the same result. In engineering problems, in general, it can not indiscriminately apply a single formula for solving a specific problem, without the full knowledge of the limitations and simplifications accepted in its derivation. It is worth mentioning that this circumstance would lead to its misuse. The so-called mathematical solutions of physical phenomena typically require simplifications and idealizations.

Therefore, there are several models that represent the transmission lines and can be classified as to the nature of their parameters in the model parameters constant and variable parameters the models with frequency.

The parameters in the models in terms of frequency are easy to use, but can not adequately represent the line in the entire range of frequencies at which these phenomena are transient in nature. In most cases, these designs increase the amplitude of the high order harmonics, distorting the waveform peaks and producing exaggerated errors.

For adequate representation of the transmission line should be considered that the longitudinal line parameters are strongly frequency dependent, including the variable

parameters in the models with frequency, the sum of the effect of soil, developed by Carson and Pollaczek, with skin effect, whose behavior as a function of frequency can be calculated using formulas derived from the Bessel equations.

Models with variable parameters in terms of frequency are considered more accurate when compared to models that consider the constant parameters. The variation depends upon the frequency. This variation may be represented by series and parallel combination of resistive and inductive elements pure.

As transmission lines are inserted in an electrical system that has many nonlinear elements and, thus, it is difficult to represent them in the frequency domain, there is a preference for line models that are developed directly in the time domain.

Another factor, that makes the model lines developed directly in the time domain are most commonly used, is that the majority of programs that perform simulations electromagnetic transients in electrical systems require that the system components are represented in time domain.

Probably, the first model to represent the transmission line directly in the time domain is designed for H. W. Dommel, which was based on the method of the characteristics or Bergeron's method. The model has combined the method of characteristics with the trapezoidal method numerical integration, resulting in an algorithm able to simulate transient electromagnetic networks whose parameters are distributed or discrete. This algorithm has undergone successive developments and is now known as the Electromagnetic Transients Program, or EMTP simply.

In situations where one wants to simulate the propagation of electromagnetic waves resulting from operations carried out maneuvers and switching the transmission lines, it can be represented the same as a cascade of π circuits.

In this model, each segment consists of a combination of series and parallel circuits composed by resistors and inductors. This results in an equivalent resistance and an inductance that depend on the frequency. It is considered in hese equivalent elements the ground and skin effects.

Due to the fact that EMTP-type programs are not easy to use, several authors suggest describing the currents and voltages in the cascade of π circuits by means of state variables. The state equations are then transformed into differential equations and can be solved using any computer language or mathematical software.

The representation of the line by means of state variables can be used in teaching basic concepts of wave propagation in transmission lines, the analysis of the distribution of currents and voltages along the line, and the simulation of electromagnetic transients in transmission lines with non-linear elements.

Although the technique of state variables is widely used in the representation of transmission lines, it can be seen in recent publications, that it was only used to represent

the parameters of which longitudinal rows can be considered constant and independent of frequency.

However, it is recognized today that the use of constant parameters to represent the line in the whole frequency range, present in the signals during the occurrence of disturbances in it, may result in responses in which the high frequency harmonic components have larger amplitudes than the real ones.

Thus, in this chapter, it is intended, using the MatLab™, to enter the effect of frequency on a line represented by π circuits connected in cascade and obtain the currents and voltages on the line from the use of the technique of state variables. The method is applied in a single-phase line, which considers the presence of soil and peel effect.

This line is approximated by a cascade of π circuits which will then be represented by state equations. The state equations, which are the voltages and currents along the line, will then be simulated in MatLab™. The cascade will also be implemented in software such as EMTP, used for simulations of electromagnetic transients in power systems. Then the results obtained with Matlab™ and EMTP are compared, considering the single-phase line.

Besides this mentioned modeling, for MatLab™ software users, the main contribution of this chapter is to demonstrate the application of this program for analyzing and simulating transient phenomena in power systems. It is important for undergraduate students for understanding the basic concepts of wave propagation and transmission lines. On the other hand, specific programs, like EMTP type programs, use a specific numeric method for solving the linear systems through numeric integration. With MatLab™ basic tools, the graduate students can apply different numeric methods for solving the same problem, comparing the results and analyzing the accuracy, precision and computational time related to each method.

2. Trapezoidal integration

It is a brief equation that relates the numerical method for lumped parameters used by the ATP. Is an arbitrary differential equation given by:

$$y(t) = k\frac{dx(t)}{dt} \tag{1}$$

The solution between the instants t_0 and t is obtained by the following:

$$dx(t) = \frac{1}{k}y(t)dt \Rightarrow \int_{x_0}^{x} x(t)dx = \frac{1}{k}\int_{t_0}^{t} y(t)dt \Rightarrow x(t) - x_0 = \frac{1}{k}\int_{t_0}^{t} y(t)dt \tag{2}$$

or finally:

$$x(t) = x_0 + \frac{1}{k}\int_{t_0}^{t} y(t)dt \tag{3}$$

The numerical solution of the equation is obtained at discrete instants of time, given:

$$\int_{x(t-\Delta t)}^{x(t)} x dx = \frac{1}{k} dx(t) = \frac{1}{k}\int_{t-\Delta t}^{t} y(t)dt \Rightarrow x(t) = x(t-\Delta t) + \frac{1}{k}\int_{t-\Delta t}^{t} y(t)dt \tag{4}$$

The solution of equation (2.4) consists of numerical integration of a discrete function y(t) which must be performed in discrete steps size Δt, which solution is the area under the curve, limited by the instants t and t-Δt as shown in Figure 1.

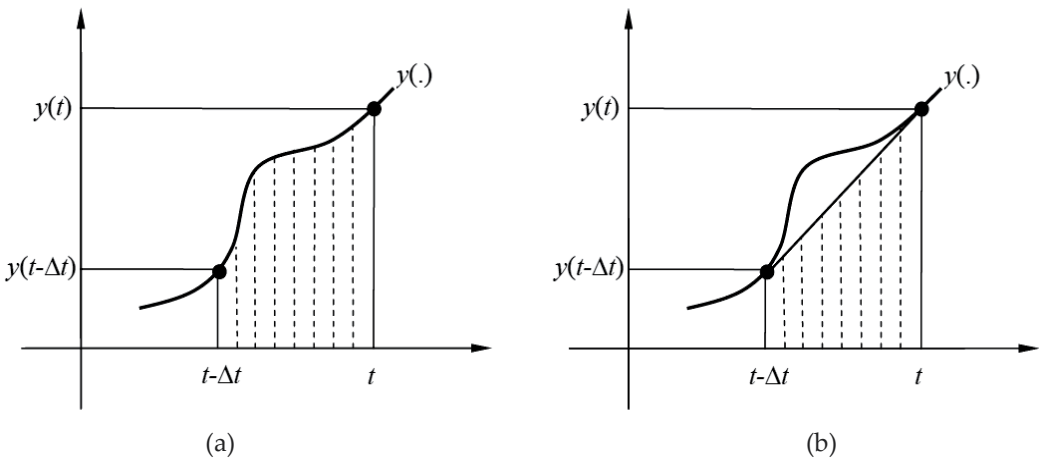


Figure 1. (a) function y(t) slight to be integrated; (b) trapezoidal rule integration.

Solving equation (2.5) by the trapezoidal rule integration, which allows a linear variation between y (t-Δt) and y (t) of the integrated, as shown in figure 1 (b), as:

$$x(t) = x(t-\Delta t) + \frac{1}{k}\int_{t+\Delta t}^{t} y(t)dt = x(t-\Delta t) + \frac{1}{k}\frac{y(t)+y(t-\Delta t)}{2}\Delta t \tag{5}$$

or:

$$x(t) = \frac{\Delta t}{2k} y(t) + \left[x(t-\Delta t) + \frac{\Delta t}{2k} y(t-\Delta t) \right] \tag{6}$$

in other words:

$$x(t) = C_{TR} y(t) + h_{TR}\left(t-\Delta t\right) \tag{7}$$

where:

$$\begin{cases} C_{TR} = \dfrac{\Delta t}{2k} \\ h_{TR}(t-\Delta t) = x(t-\Delta t) + \dfrac{\Delta t}{2k} y(t-\Delta t) \end{cases} \tag{8}$$

It may be noted that the term historical $h_{TR}(t-\Delta t)$ depends on x (t) and y (t) after t seconds, Δt, is already known during the solution.

3. Modeling cascada of π circuits

Discussion will now represent a mathematical model for a transmission line using an electrical circuit. With this model, you can make a study of the behavior of a transmission line for energizing and switching transient simulations. A transmission line, whose parameters can be considered independent of frequency, can be represented in an approximate manner and following a series of restrictions, as a cascade of π circuits.

Each segment of π circuit consists of one resistor and one series inductance in series. Completing the π circuit, there are components in parallel: capacitance and conductance. It is shown in Figure 2.

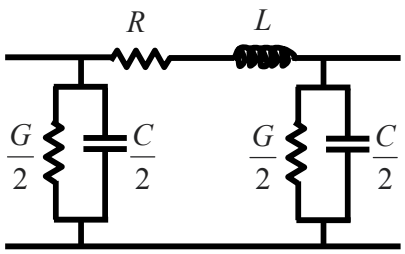


Figure 2. A π circuit.

It's represented a transmission line through this model, connects to n π circuits in cascade. So Figure 3 shows a single-phase transmission line length d represented by n π circuits connected in cascade.

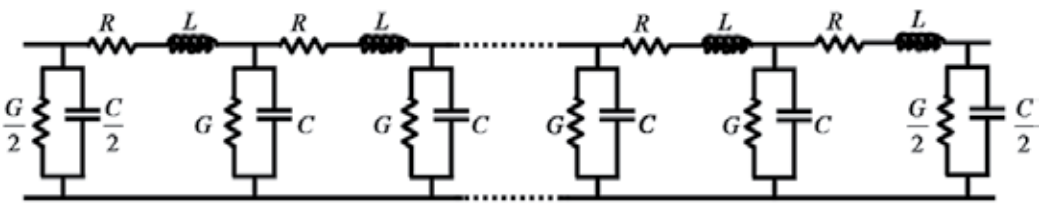


Figure 3. Line represented by a cascade of π circuits.

In Figure 3, the parameters R and L are the resistance and inductance of the longitudinal line, respectively. The parameters G and C are the capacitance and conductance of transverse dispersion, respectively. These parameters are written as:

$$R = R'\frac{d}{n} \tag{9}$$

$$L = L'\frac{d}{n} \tag{10}$$

$$G = G'\frac{d}{n} \tag{11}$$

$$C = C'\frac{d}{n} \tag{12}$$

In the equations (3.1) to (3.4), R 'and L' are the resistance and inductance of the longitudinal row per unit length, respectively while the terms G 'and C' are the capacitance and conductance per unit cross-sectional line in length.

Using this representation of the line, a state model is formulated for the energy system that uses the voltages on the capacitors and the currents through the inductors as state variables. The system that describes the state equations is transformed into a set of differential equations whose solution is given by the use of trapezoidal integration. The state variables are found by solving the set of equations.

Although the technique of state variables is widely used in the representation of transmission lines, it is applied only to depictions of longitudinal lines whose parameters can be considered constant and independent of frequency.

However, it is recognized today that the use of constant parameters to represent the line in the whole frequency range, present in the signals during the occurrence of disturbances in it, may result in responses in which the high frequency harmonic components have larger amplitudes than are actually.

4. Line representation with parameters of frequency dependent

The representation of transmission lines through cascades of π circuits, taking into account the effect of frequency, is usually implemented in EMTP-type programs.

Another drawback of the type EMTP programs is that they limit the amount of π circuit that can be used to represent the line. Thus, depending on the length of the line to be depicted, the quality of results obtained from simulations may be compromised.

To circumvent the difficulties mentioned, it is suggested to describe the cascade of π circuits by means of state equations. However, some authors disregarded the effect of frequency on the parameters of the longitudinal line.

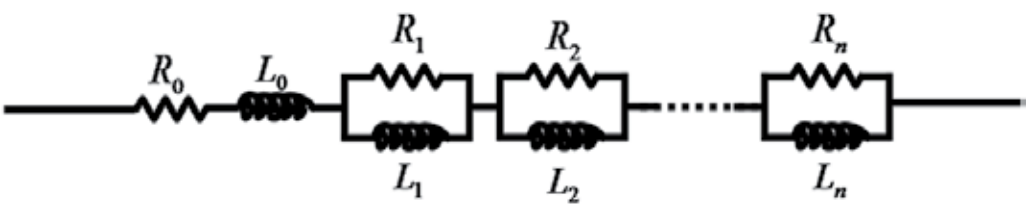


Figure 4. Circuit that represents the longitudinal parameters of the line.

The models proposed by would become more complete if the effect of frequency on the longitudinal parameters of the line was inserted in them.

So, the parameters of a transmission line can be synthesized by means of a circuit of the type shown in Figure 4.

It's used a cascade of π circuits to represent a transmission line taking into account the effect of frequency on the longitudinal parameters. In this case, every π circuit will be shown in Figure 5.

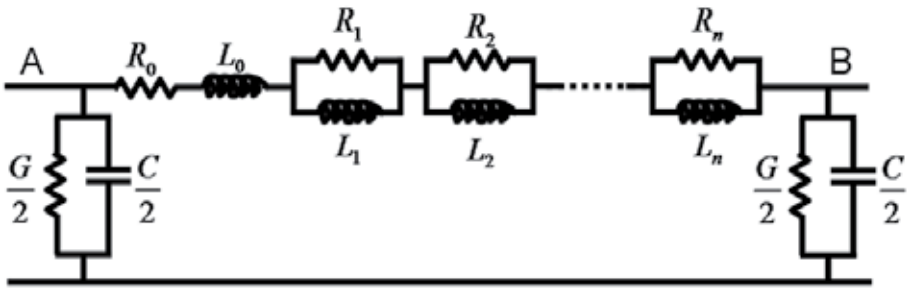


Figure 5. Cascade of π circuits considering the effect of frequency.

In Figure 5, the RL parallel associations are as many as necessary to represent the variation of parameters in each decade of frequency that will be considered. First, arrays are displayed in a state for a line represented by a single π circuit, whereas the effect of frequency is synthesized by n RL associations. Then, the results will be extended to a line represented by a cascade of n π circuit, considering n RL associations to synthesize the effect of frequency.

Before being certain state equations for a line represented by a cascade of n π circuits, it will be shown in detail the development of equations of state considering only one π circuit.

Then, the development done for a single π circuit element can be extended to a generic cascade with any quantity of these circuits.

Considering, as shown in the Figure 5, a transmission line represented by a single π circuit, the effect of frequency on the longitudinal parameters is represented by n RL associations.

In line shown in Figure 5, the voltages at terminals A and B are u(t) and $v_k(t)$, respectively. It is also considering that the currents i_{k0}, i_{k1}, ..., i_{km} are circulating through the inductors L_0, L_1, L_2, ..., L_m, respectively. These currents are dependent time functions and the notations do not include the time dependence for more simplicity. This simplification is also applied to the $v_k(t)$ state variable. From the currents and voltages in the circuit of Figure 5 it can be determined:

$$\frac{di_{k0}}{dt} = \frac{i_{k0}}{L_0}\left(-\sum_{j=1}^{m} R_j\right) + \frac{1}{L_{L0}}\left(\sum_{j=1}^{m} R_j i_{kj}\right) + \frac{1}{L_0}u(t) - \frac{1}{L_0}v_k \tag{13}$$

$$\frac{di_{k1}}{dt} = \frac{R_1}{L_1} i_{k0} - \frac{R_1}{L_1} i_{k1} \tag{14}$$

$$\frac{di_{k2}}{dt} = \frac{R_2}{L_2} i_{k0} - \frac{R_2}{L_2} i_{k2} \tag{15}$$

$$\frac{di_{km}}{dt} = \frac{R_m}{L_m} i_{k0} - \frac{R_m}{L_m} i_{km} \tag{16}$$

$$\frac{dv_k(t)}{dt} = \frac{2}{C} i_{k0} - \frac{G}{C} v_k(t) \tag{17}$$

The equations (4.1) to (4.5), describing the circuit shown in figure 4.2, can be written as:

$$\dot{x} = Ax + Bu \tag{18}$$

For one π circuit, the A matrix is substituted by:

$$A_\pi = \begin{bmatrix} -\frac{\sum_{j=0}^{j=m} R_j}{L_0} & \frac{R_1}{L_0} & \frac{R_2}{L_0} & \cdots & \frac{R_m}{L_0} & -\frac{1}{L_0} \\ \frac{R_1}{L_0} & -\frac{R_1}{L_0} & 0 & \cdots & 0 & 0 \\ \frac{R_2}{L_2} & 0 & -\frac{R_2}{L_2} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ \frac{R_m}{L_m} & 0 & 0 & \cdots & -\frac{R_m}{L_m} & 0 \\ \frac{2}{C} & 0 & 0 & \cdots & 0 & -\frac{G}{C} \end{bmatrix} \tag{19}$$

$$B^T = \begin{bmatrix} \frac{1}{L_0} & 0 & 0 & \ldots & 0 & 0 \end{bmatrix} \tag{20}$$

$$x_k^T = \begin{bmatrix} i_{k0} & i_{k1} & i_{k2} & \cdots & i_{km} & v_k \end{bmatrix} \tag{21}$$

$$\dot{x}_k = \frac{dx_k}{dt} = \begin{bmatrix} \frac{di_{k0}}{dt} & \frac{di_{k1}}{dt} & \frac{di_{k2}}{dt} & \cdots & \frac{di_{km}}{dt} & \frac{dv_k}{dt} \end{bmatrix}^T \tag{22}$$

In the equations (4.8) and (4.9), B^T e x_k^T correspond to transposed B and x_k, respectively.

The obtained results show that the vesctor x_k have (m + 2) elements and the matrix A is a (m + 2) order square matrix.

Based on the equations and the results for one π circuit, it can be extended the analysis to a cascade of π circuits. Thus, the matrix A will have an order of n(m +2) and the vector x has dimension n(m +2). The A matrix can be written as:

$$A = \begin{bmatrix} A_{11} & A_{12} & 0 & \cdots & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & \ddots & \ddots & \ddots & 0 \\ 0 & A_{32} & A_{33} & \ddots & 0 & \ddots & 0 \\ 0 & 0 & A_{43} & \ddots & A_{(n-3)(n-2)} & 0 & 0 \\ 0 & \ddots & 0 & \ddots & A_{(n-2)(n-2)} & A_{(n-2)(n-1)} & 0 \\ 0 & \ddots & \ddots & \ddots & A_{(n-1)(n-2)} & A_{(n-1)(n-1)} & A_{(n-1)n} \\ 0 & 0 & 0 & \cdots & 0 & A_{n(n-1)} & A_{nn} \end{bmatrix} \tag{23}$$

The elements x_1, x_2, …, x_n are describe by equation (4.9). In equation (4.11), A is a tridiagonal matrix which elements are square matrices of order (m +2). In this case, a generic element A_{KK} at main diagonal of the matrix A is written as:

$$A_{kk} = A_\pi \tag{24}$$

The A_π matrix is defined in equation (4.7).

An element of any upper subdiagonal in equation (4.11) is a square matrix of order (m +2) which the only one nonzero element is located in the first column of last row and has the value $\left(-1/C\right)$.

The structure is:

$$A_{ik} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \cdots & \vdots \\ 0 & \cdots & \ddots & \vdots \\ -1/C & 0 & \cdots & 0 \end{bmatrix} \tag{25}$$

$$i = k-1, \quad 2 \le k \le n$$

The subdiagonal elements in the equation (4.11) are square matrices of order (m +2). These arrays have a single nonzero element which is in the last column of first row. It is a value of $\left(1/L_0\right)$. The structure is:

$$A_{ik} = \begin{bmatrix} 0 & \cdots & 0 & 1/L_0 \\ \vdots & \ddots & \cdots & 0 \\ \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \qquad (26)$$
$$i = k+1, \quad 1 \le k \le n-1$$

Considering a cascade of π circuits, the vector B has the dimension of n(m +2) and if it is connected a u(t) source at the beginning of the line, the vector B has a single nonzero element, which is the first array element and it has the value $\left(1/L_0\right)$.

The state equation, that describes a line representation by a cascade of π circuits cant be solved by numerical methods, like Euler and Heun methods. Other numeric methods are described in the next item

5. Other numeric methods

Besides the trapezoidal integration, there are also other numeric methods that can be used for solving state equations related to the modeling of transient phenomena in transmission lines. Among them, it may be mentioned, for example, the Simpson's and the Runge-Kutta's method. The following items will describe these methods. In this case, the MatLab™ is important because it facilitates the comparisons among the mentioned methods. With this software, undergraduate students can analyze the application of different numeric methods for solving an engineering problem, besides the analysis about wave propagation and the transmission line modeling. Graduated students can analyze what it is the best option for specific transmission line characteristics and develop the numeric method related to the simulation of transient phenomena in power systems.

5.1. Simpson's rule

Simpson's rule is based on the assumption that, given short intervals of time the derivative of the function to be integrated, the function (y') can be approximated by a second degree function as shown in Figure 6.

From Figure 1, it is obtained:

$$y\left(t_{k+1}\right) = \frac{1}{3}\left[y'\left(t_{k+1}\right) + 4y'\left(t_M\right) + y'\left(t_k\right)\right]\Delta t \qquad (27)$$

From (5.1) and (4.6), it is obtained:

$$x\left(t_{k+1}\right)=a_1\left(a_2+a_3+a_4\right) \tag{28}$$

Equation (5.2) is applied Simpson's rule for solving the state equation of state of the type shown in (5.1). In this case, the matrices a_1, a_2, a_3 and a_4 are written as:

$$a_1=\left(I-\frac{\Delta t}{6}A\right) \tag{29}$$

$$a_2=\left(I+\frac{\Delta t}{6}A\right)x\left(t_k\right) \tag{30}$$

$$a_3=\frac{2\Delta t}{3}A\,y\left(t_M\right) \tag{31}$$

$$a_4=\frac{\Delta t}{6}B\left[u\left(t_k\right)+4u\left(t_M\right)+u\left(t_{k+1}\right)\right] \tag{32}$$

It is defined the following terms:

$$\Delta t=t_{k+1}-t_k \tag{33}$$

$$t_M=\frac{t_{k+1}+t_k}{2} \tag{34}$$

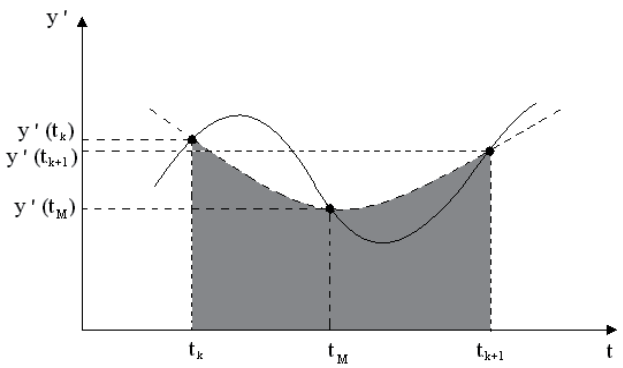


Figure 6. Approximation of the derivative of the function to be integrated (solid curve) by a function of the second degree (dashed curve).

5.2. Runge-Kutta's rule

Runge-Kutta's rule is defined by the following:

$$x\left(t_{k+1}\right)=x\left(t_k\right)+\frac{1}{6}\left(k_1+2k_2+2k_3+k_4\right) \tag{35}$$

The terms in the last equation are described by:

$$k_1 = \Delta t\left[A\,x\left(t_k\right) + B\,u\left(t_k\right)\right] \tag{36}$$

$$k_2 = \Delta t\left\{A\left[x\left(t_k\right) + \frac{k_1}{2}\right] + B\,u\left(t_k\right)\right\} \tag{37}$$

$$k_3 = \Delta t\left\{A\left[x\left(t_k\right) + \frac{k_2}{2}\right] + B\,u\left(t_k\right)\right\} \tag{38}$$

$$k_4 = \Delta t\left\{A\left[x\left(t_k\right) + k_3\right] + B\,u\left(t_k\right)\right\} \tag{39}$$

5.3. Comparisons among the numeric methods

The Matlab™ software is easily applied for the development of numeric routines. Using this characteristic, it is possible to compare different numerical methods. Figure 7 shows the results for the three methods investigated for the simulation of electromagnetic transients in a single-phase transmission line.

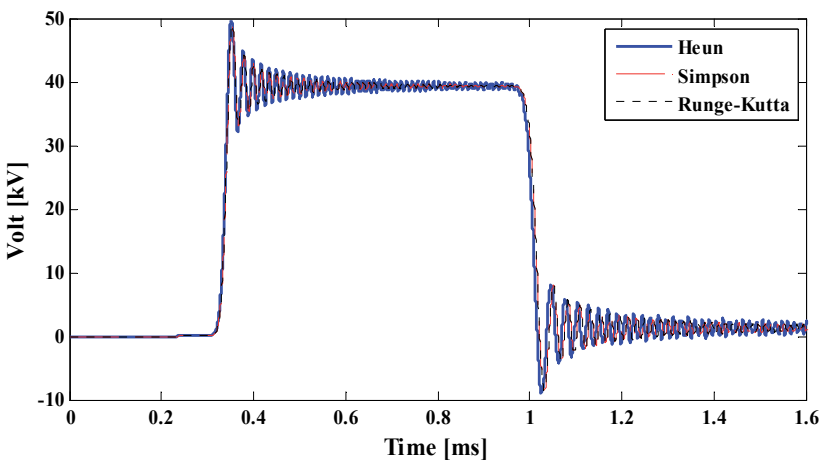


Figure 7. Comparisons among the three numerical methods.

In the last figure, it is shown the result obtained with the application of a step voltage source on the initial transmission line. This is a 20 kV step voltage source. The transmission line is modeled as a mono-phase circuit and the linear system is solved using three numeric methods: trapezoidal rule, Simpson's rule and Range-Kutta's one.

Based on the results shown in Figure 7, the three numeric methods lead to similar results considering the method accuracy. Because of these results, it is necessary to analyze the other characteristics. An important characteristic for the application of the numeric methods for transient analysis in transmission lines is the simulation time. For example, this is

important for analysis in very short simulation times. The next table shows the simulation time comparisons obtained with the MatLab™ software.

Step of calculating (μs)	Computational effort (simulation time)		
	Heun	Simpson	Runge-Kutta
0,1	t1	22,52 t1	0,63 t1
0,5	t2	8,94 t2	0,23 t2
1,0	t3	8,40 t3	0,26 t3

Table 1. Simulation time required by numerical methods

Table 1 shows the time relationship between the applied numerical methods. It is possible to carry out these comparisons because the Matlab™ provides a simple way to replace the codes related to the numeric routine without changing the structure of the main code, maintaining the same computational flowchart. Using this characteristic, it is possible to determine the simulation time of each numeric method, because the computational time for introducing the data and other characteristics of the simulated transmission line is equal for all compared numeric methods. Because of this, the MatLab™ is an adequate tool for developing new models and numeric routines used for analysis and simulations of transient phenomena in transmission lines.

6. Model implementation of single phase line

In this section, it is presented the implementation of a single-phase transmission line through a cascade of π circuits, considering the effect of frequency on their longitudinal parameters and using the concept of state variables. Then, the results obtained are compared with results obtained with EMTP.

6.1. Block diagram of the program for single-phase line

The model that represents the single-phase line was implemented in a microcomputer using the software MatLab™.

Data from single-phase transmission line are read in the first part of the program, then the parameters are calculated from the transmission line considering the influence of frequency.

After observing the behavior of single-phase line parameters as a function of frequency, it is necessary to represent this influence on the transmission line model proposed in item 4. This was done using the method called vector fitting and the longitudinal single phase line parameters are fitted by means of rational functions.

With these parameters synthesized and distributed in the proposed model of a single-phase line, it is possible to calculate the voltages and currents at the terminals of this line or at any point of the line.

The model represented by a cascade π circuits can be represented by a linear system that is represented by state variables. For the solution of the system represented by $\dot{x} = [A]x + [B]u$, it's used Heun formula (trapezoidal integration method). This method is widely used in simulations of electromagnetic transients in power systems

Figure 8 is shown a block diagram of the algorithm of the program developed for single-phase line.

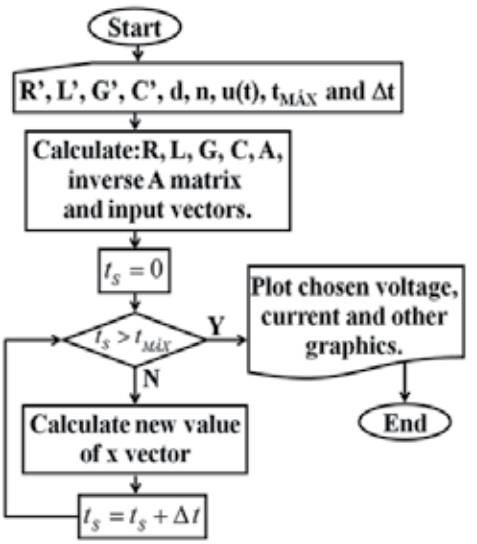


Figure 8. Block diagram of the routine developed for a single-phase.

6.2. Calculation of the parameters of the transmission line single phase

It's possible to bring the longitudinal parameters of single-phase line by means of rational functions. Using the vector fitting method and allowing the fitting the frequency dependence of these parameters, this effect is entered in the discrete parameter model.

The equation that summarizes the parameters of longitudinal single-phase line is given by:

$$Z_{VFIT}(\omega) = R_0 + j\omega L_0 + \frac{j\omega R_1}{j\omega + \frac{R_1}{L_1}} + \frac{j\omega R_2}{j\omega + \frac{R_2}{L_2}} + \frac{j\omega R_3}{j\omega + \frac{R_3}{L_3}} + \frac{j\omega R_4}{j\omega + \frac{R_4}{L_4}} \tag{40}$$

The values of R and L in the equation (6.1) found by the method vector fitting are shown in Table 2.

From the values of Table 1, it can view a summary of the parameters of longitudinal line as follows: replacing the values of Table 1 in the expression (6.1) and attribute values are included in the frequency range 10^1 to 10^6 for it is possible to calculate the longitudinal impedance synthesized.

In the equivalent circuit, the values were considered in the frequency range from 10^1 to 10^6, because transients that occur in the transmission line are within this frequency range. So the four blocks RL in parallel, shown in Figure 9, are representing the influence of frequency on longitudinal parameters of single-phase transmission line.

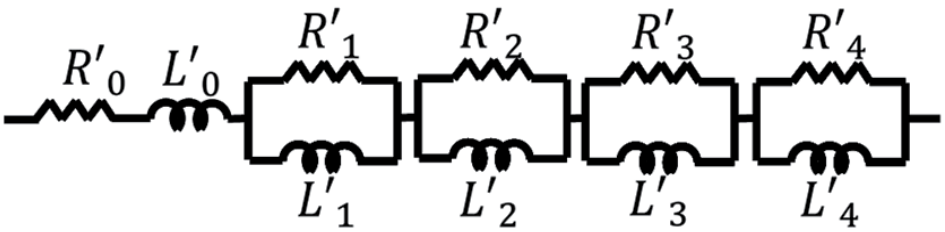


Figure 9. Circuit that represents the longitudinal parameters of the line.

Resistors (Ω/km)		Inductors (mH/km)	
R'_0	0,026	L'_0	2,209
R'_1	1,470	L'_1	0,74
R'_2	2,354	L'_2	0,12
R'_3	20,149	L'_3	0,10
R'_4	111,111	L'_4	0,05

Table 2. Values of elements R e L for a single-phase.

7. Transmission line with corona effect

7.1. General aspects about the corona

The corona discharge mechanism is an electrostatic phenomenon due to ionization in an insulating material, usually a gas, subject to electric field intensities above a critical level.

Electrical discharges in gases are usually triggered by an electric field that accelerates free electrons therein. When these electrons acquire enough energy from the electric field, they can produce new electrons from the collision with other atoms. It is the process of impact ionization. During its acceleration in the electric field, each free electron collides with atoms of oxygen, nitrogen and other present gases, missing, that collision, part of its kinetic energy. Occasionally, it can achieve an electron atom with sufficient force so as to excite it. Under these conditions, the atom is achieved to a higher energy state. The orbital state of one or more electrons and the electron moves colliding with the atom loses some of its energy to create this state. Subsequently, the atom can hit revert to its initial state, releasing the excess energy as heat, light, electromagnetic radiation and acoustic energy. An electron can also collide with a positive ion, converting it into neutral atom. This process, called recombination, also releases excess energy.

7.2. Corona effect representation

After the pioneering work of Peek (1915), several measurements have made on lines and experimental laboratories have examined the nature of the corona and its influence on wave propagation in transmission lines. These works have had fundamental importance, contributing to the understanding of the basic mechanism of the corona.

In 1954, Wagner et al. (1954) and Wagner and Lloyd et al. (1955) published two articles that would be a reference for future work on corona. Voltage experimental measurements were made of in the laboratory of a conductor under corona (project called Tidd 500 kV).

Through these experimental results, it has developed empirical formulas and procedures for considering the effects of attenuation and distortion in the spread of outbreaks. These formulas and procedures are based on the voltage gradient, and voltage curves are obtained from attenuation measurements and the power dissipation due to the corona effect. The usefulness of these methods is limited however, because it requires different approaches and uses abacuses.

The corona model can be divided into three classes: analog models, mathematical models and physical models. The electrical circuit analog models are designed to reproduce the geometric increase in the capacitance of the conductor to the voltage reaches critical ionization.

Like the analog models, mathematical models roughly reproduce the characteristics of drivers under the corona, but by means of mathematical equations. Several authors have empirical formulations for the variation of capacitance of the line based on functions and constants derived from measurements. Most models show a linear relationship between capacitance and dynamic tension.

Due to the complexity in describing mathematically the physical phenomena involved and the insufficient amount of data propagation in corona, it is not available generic models of this nature yet.

7.3. Gary's model

The equations describing the corona effect is not easily implemented in the differential equations of the transmission line in order to obtain a solution formulation easyly. Thus, to obtain responses directly in the time domain, numerical models are used such as the finite difference method and the method of the characteristics. This last category of models has been developed to be implemented in EMTP-type programs. Some of these models use non-linear resistors and capacitors that are dependent on the voltage applied on them. However, most existing models of corona present satisfactory results only for a specific situation.

The mechanism of the corona effect can also be represented by the model of Gary, using a capacitance and a non-linear conductance to represent the accumulation and the pressure drops in the line. The capacitance and conductance mentioned above are variables with the

applied voltage on them and are called corona capacitance (Cc) and corona conductance (Gc). Cc and Gc elements are obtained by known analytical function, and, therefore, this representation for the corona effect is called analytical model of the corona effect. This model for the corona effect can be inserted in lines represented by a cascade of π circuits, where currents and voltages along line is described by state variables.

If the capacitance is represented by Gary's corona model, it is defined as:

$$C_C = \begin{cases} C\eta\left(\dfrac{v}{V_C}\right)^{\eta-1} & se \quad v \geq V_C \\ 0 & se \quad v < V_C \end{cases} \tag{41}$$

In equation (7.1), C_C is the corona capacitance, C is the geometric capacitance of the line segment represented by a π circuit, v is the voltage being applied to the line capacitance, C_V is the minimum voltage required to the corona effect accurrance and η is a coefficient defined as:

$$\eta = 0,22r + 1,2 \tag{42}$$

where: r the radius of the conductor in centimeters.

Thus, given the presence of the corona effect and the effect of frequency, a differential element row can be represented as shown in Figure 10.

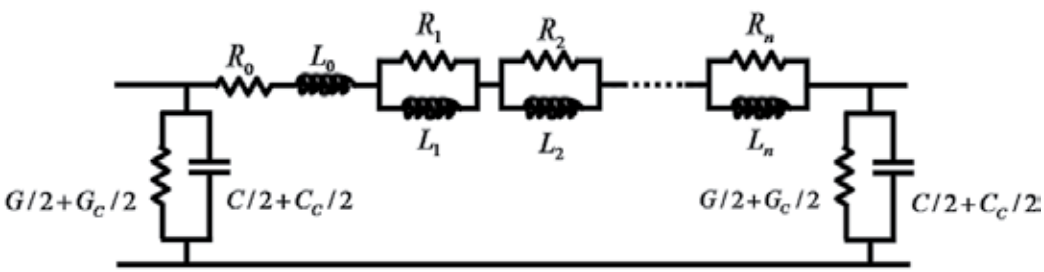


Figure 10. A differential element of line considering the corona effects

The corona conductance to Gary's model is defined as:

$$G_C = k_C\left(\frac{1 - V_C}{v}\right)^2 \tag{43}$$

Gary's model considers that the corona effect is manifested only if the voltage V_C is greater than v and the rate of variation over time v is positive. Thus, if the corona effect is manifested at a given point P of the line, the voltage V_P at this point must satisfy the following conditions:

$$V_p > V_C \quad and \quad \frac{dV_p}{dt} > 0 \tag{44}$$

Thus, for the corona effect is present at a generic point of the line represented by a cascade of π circuits, as shown in figure 10, it is necessary that the voltage at this point satisfying the two conditions shown in equation (7.4). If a condition is not met, this point will not have increased the capacitance and conductance representing corona. Therefore, the A matrix shown in equation (4.6) should be changed for each iteration as a function of cross-line voltage.

8. Testing the model

Checking the effectiveness of the developed model, it is simulated the energizing of a transmission line shown in figure 11, considering frequency independent line parameters and frequency dependent ones.

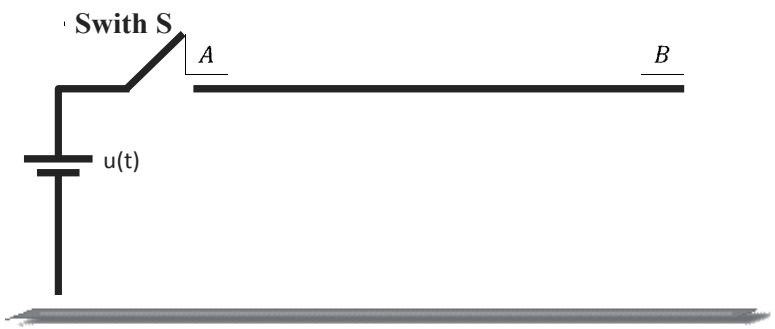


Figure 11. Single phase line representation with opened terminal.

In Figure 11, S is a switch that be closed at time t = 0, energizing the line through a voltage source u(t). In the current procedure, the terminal B is open and the other terminal is powered by a constant voltage source. For frequency dependent line parameters, it is considered that the longitudinal parameters of the line per unit length can be perfectly summed up by a circuit consisting of four RL parallel blocks connected in series. The structure is completed using a RL series block as shown in Figure 9.

The values of R and L used to synthesize the effect of frequency on the longitudinal parameters of the line were obtained using the method proposed by [10] and are shown in Table 1. The parameters of the unit transverse line shown in figure 3 are G′=0,556 μS/km e C′ =11,11nF/km.

9. Simulation analysis of transmission lines with single phase representation

In all simulations, it is used the following values: the transmission line has 10 kilometers. It is represented through 200 π circuits. The time step used in the simulations is 50 ns and the

simulation period is 600 μs. The voltage in the initial of the line is 1 kV. It can also be considered as 1 pu.

Since the values of R and L elements of the cascade of π circuits that describe the line are known, it can be obtained the state equations that describe the behavior of currents and voltages along the line. The simulations using the model proposed in this chapter were performed in MatLab™ program, using the trapezoidal integration method. Considering the frequency independent line parameters, the circuit of Figure 4 is reduced to the R_0 and L_0 elements. In this case, the values are: R_0 = 0.05 Ω/km and L_0 = 1 mH/km.

The values of Table 9.1 are written per kilometer for the generic π circuit. For the simulation, it used the values represented in the Table 9.2 that are resistance and inductance [Ω] and [H], respectively. In the Figs 12-21, it is used the values of Table 4.

Figures 12-17 show the relationship between the number of RL parallel blocks and the inclusion of the frequency influence. In Figure 12, the resistance values are obtained using only one RL parallel block related to the high frequencies. Because of this, the resistance values for all frequency values are equal. In this case, the value is equal to the R_4 value per length unit.

Parameter	100 π circuits	200 π circuits
R	5 mΩ	2.5 mΩ
L	100 mH	50 mH
G	556 μS	278 μS
C	1.111 nF	555.5 nF

Table 3. Values of line parameter used in all simulations without frequency.

Resistors(Ω)		Inductors(μH)	
R_0'	0.0013	L_0'	110.45
R_1'	0.0735	L_1'	37
R_2'	0.1177	L_2'	6
R_3'	1.0075	L_3'	5
R_4'	5.5556	L_4'	2.5

Table 4. Values of line parameter used in simulations with 200 π circuits.

From the results of Figures 13 and 14, it is observed that the synthesis of the effect of frequency on the resistance can only be considered when inserted in the cascade of π circuits, at least, two RL parallel blocks, because each block is related to a frequency set point.

So, if more blocks are used, more frequency set points are obtained. It is confirmed through the results shown in Figures 15, 16 and 17 where the inductance values are presented. So, it is concluded that the synthesis of the longitudinal line parameters is improved with the increase of the number of RL parallel blocks.

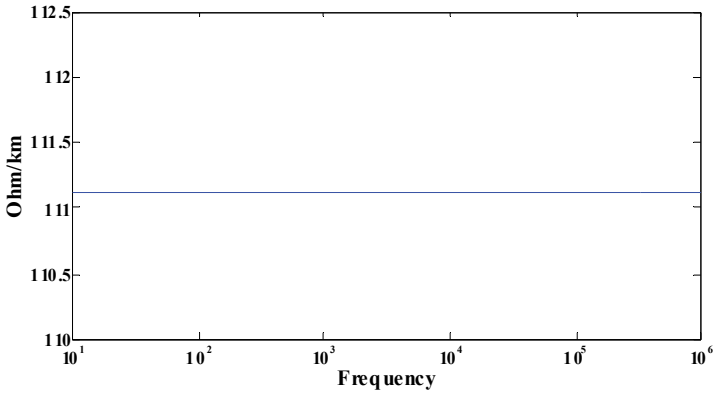


Figure 12. Synthesis of resistance per unit length using $R'_4L'_4$ parallel block.

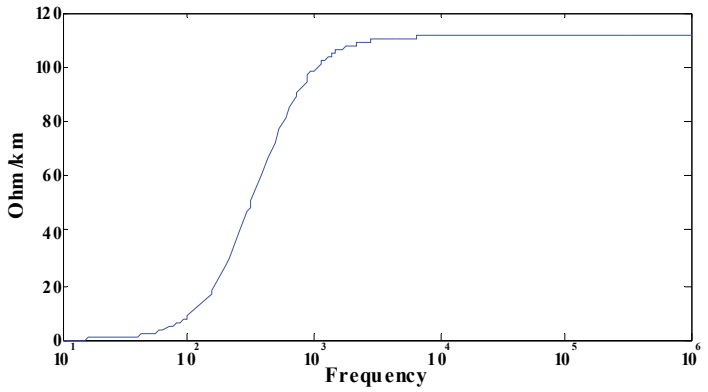


Figure 13. Synthesis of resistance per unit length using $R'_1L'_1$ and $R'_4L'_4$ parallel blocks.

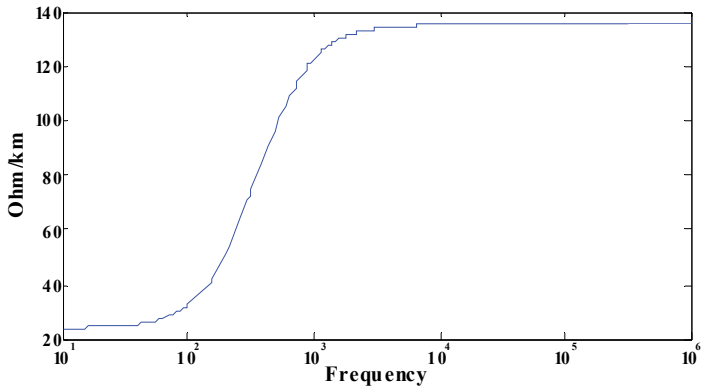


Figure 14. Synthesis of resistance per unit length using four RL parallel blocks.

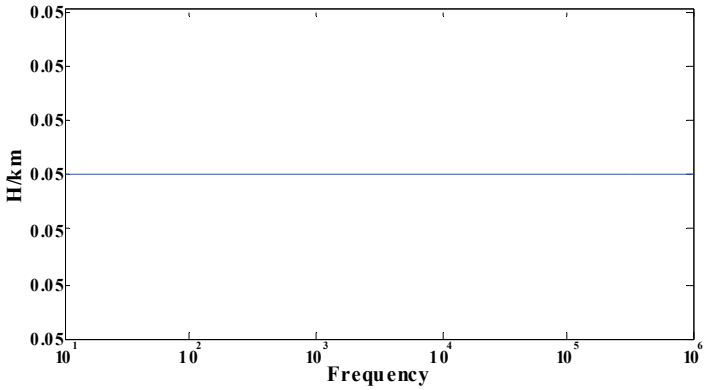


Figure 15. Synthesis of inductance per unit length using only $R'_4L'_4$ parallel block.

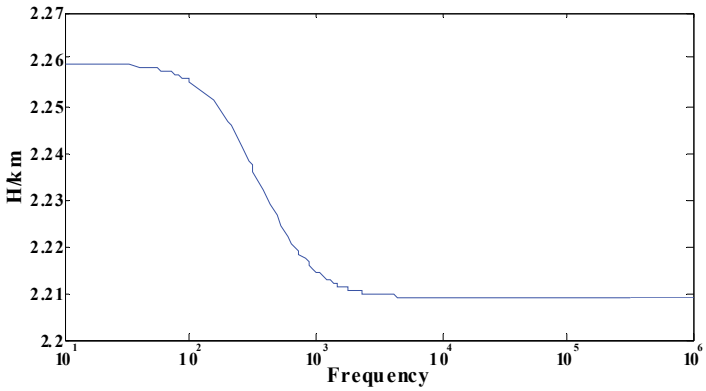


Figure 16. Synthesis of inductance per unit length using $R'_1L'_1$ and $R'_4L'_4$ parallel blocks.

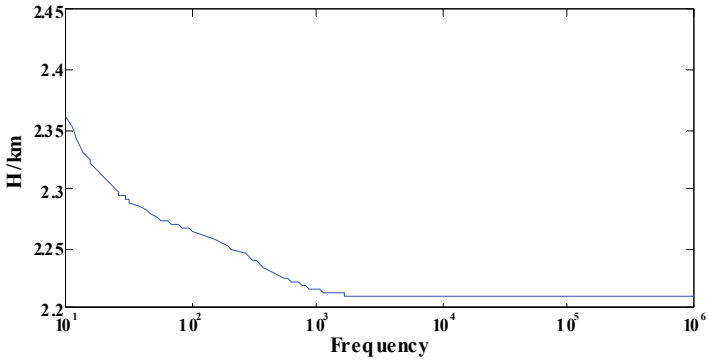


Figure 17. Synthesis of inductance per unit length using four RL parallel blocks.

The result of the simulation made for the voltage input signal u(t) can be seen in Figure 18. It is shown the output voltage at the receiving end terminal of the line without the frequency

influence. Using the routine without the influence of frequency from the results of Figure. 18, it is observed that there is a period of time related to the time of signal propagation through the line.

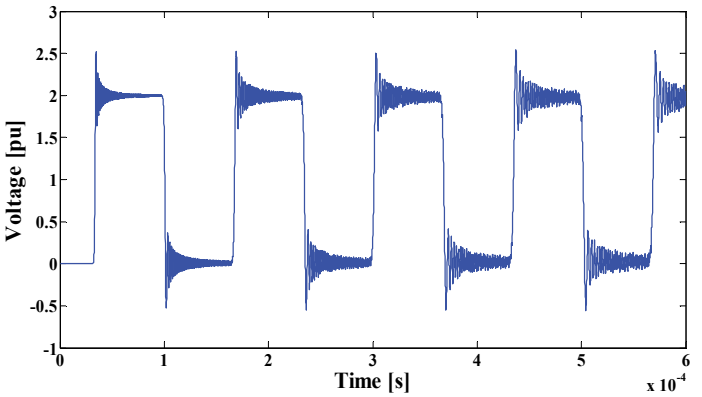


Figure 18. Energization of the transmission line without the effect of frequency considering 200 π circuits.

Thus, it represents a time delay between input signal and output signal. After the delay, there are oscillations associated with wave reflections on the transmission line terminals that make up the output voltage shown.

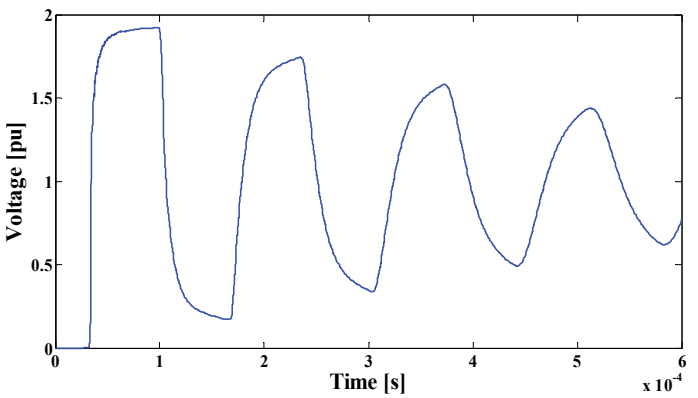


Figure 19. Voltage at the end of the transmission line with the effect of frequency considering 200 π circuits.

Using the routine with frequency influence in longitudinal parameters, it is obtained the results of Figure 20. In this figure, comparing to Figure 19, the voltage signal is attenuated because the inclusion of the frequency influence.

Figures 21 and 22 show the influence of frequency on the current results. Without the frequency influence, the obtained signal current is not attenuated and is highly modified by numeric oscillations (Figure 14). On the other hand, when the routine considers the

influence of frequency (Figure 22), it is clear that the current signal could not contain those oscillations shown in Figure 21. So, those oscillations are numeric oscillations and they can be associated to the representation of the longitudinal line parameters which does not consider the frequency influence.

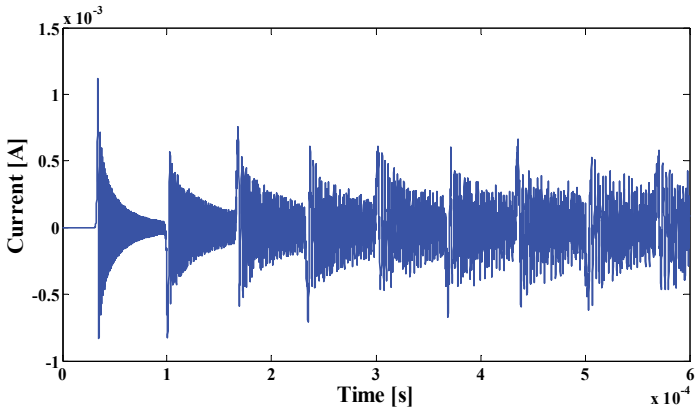


Figure 20. Current at the end of the transmission line without the effect of frequency.

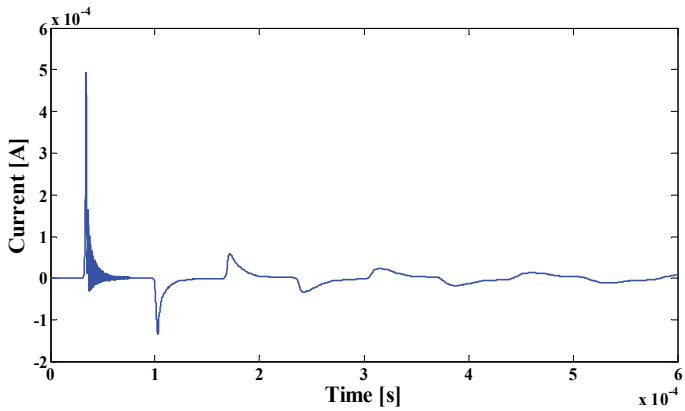


Figure 21. Current at the end of the transmission considering the effect of frequency.

So, for the sequence of this work, it should be investigated what is the saturation point for the number of the π circuits and the number of the RL parallel blocks. It is carried out using the simulation results from several voltage signals that will be used as voltage sources at the initial line terminal.

For Figures 22 and 23, using the routine without the effect of frequency, it is analyzed a particular stretch of the simulation at the end of the line. It is shown the voltage values without frequency influence. In Figures. 22 and 23 it is used the values of Table 2, using 100 and 200 π circuits, respectively.

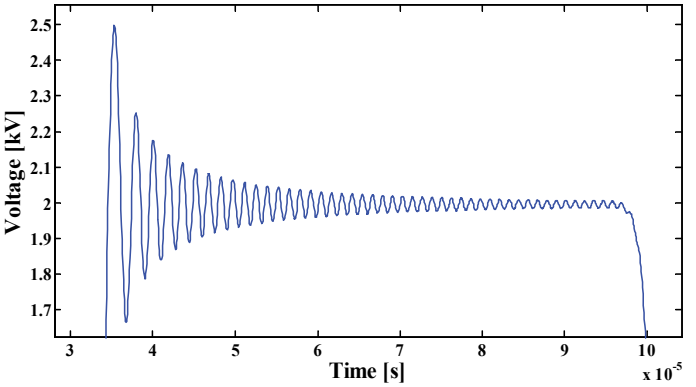


Figure 22. Specific portion of the simulation at the end of the line with 100 π circuits and without the effect of frequency.

Based on the mentioned results, it concludes that increasing the number of π circuits leads to a decrease in the numerical oscillations.

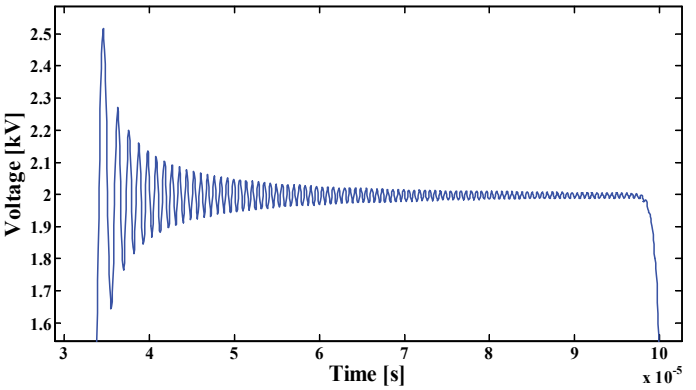


Figure 23. Specific portion of the simulation at the end of the line with 200 π circuits and without the effect of frequency.

In Figure 23, it is clearly a greater condensation of numerical oscillations in both x and y axes in contrast to Figure 23, where the period of the oscillations and the peak value are higher than those obtained in Figure 23. In Figures 24, 25 and 26, it is used the routine with the effect of frequency, analyzing the number of the RL parallel blocks inserted in the cascade of π circuits, searching for reducing of numerical oscillations. In Figures 24, 25 and 26, it was used the values of Tables 3 and 1, as well as, 3 and 4 RL parallel blocks, respectively.

From the results of Figures 24, 25 and 19, it is observed that the used routine including the effect of frequency in the cascade of π circuits obtains considerably fewer numerical oscillations when compared to Figures 22 and 23. From these results, it is evident that if it is increased the number of the RL parallel blocks, it is reduced the numerical oscillations and the voltage in the line end has a smoothly curve in the time domain.

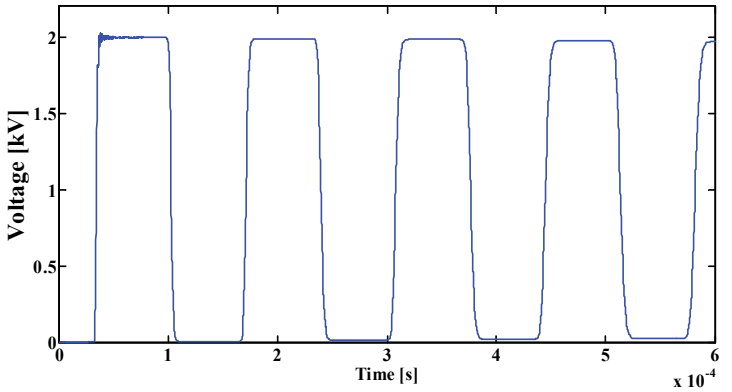


Figure 24. Voltage at the end of the line with the effect of frequency considering 200 π circuits and using R_4L_4 parallel block.

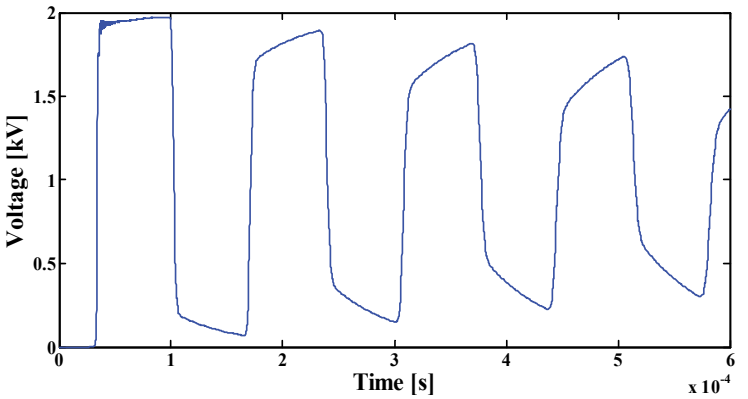


Figure 25. Voltage at the end of the line with the effect of frequency considering 200 π circuit and using R_0L_0, R_2L_2 e R_4L_4 parallel blocks.

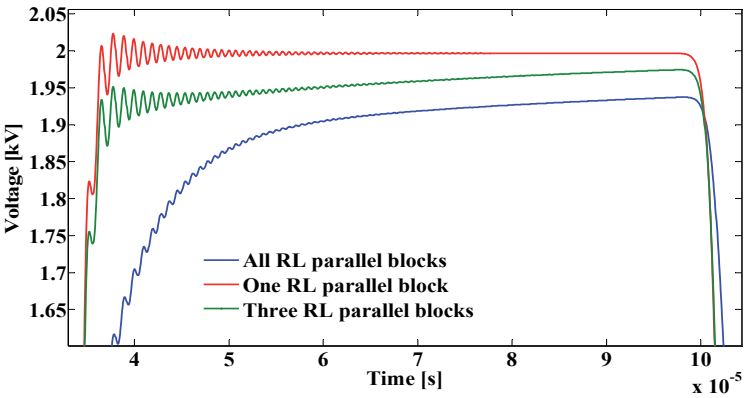


Figure 26. Comparison of figures 24, 25 and 19 of the tension at the end of the line with the effect of frequency.

Concluding the analysis of the results, Figure 26 shows a comparison among the results of Figures 24, 25 and 19, leaving a clear decrease in the numerical oscillations. In this figure, it is shown a time range of the time simulation used at the last three figures. In this case, it is clearly shown that, increasing the number of RL parallel blocks, the numerical oscillations are decreased and the frequency influence is better reproduced through the state equations in mathematical software.

Considering the corona effect, Gary's model is applied with the routine described in the previous items. It is used the line representation without frequency influence, because the representation with frequency influence has not hardly analyzed. The corona effect is introduced by

$$C_C = C \cdot \eta \cdot \left(\frac{V}{V_C} \right)^{\eta - 1}$$
$$\eta = 0.22 \cdot R_{COND} + 1.2 \tag{45}$$

In this case, R_{COND} is the conductor phase radius in [cm], C is the transversal line capacitance and the C_C is the new value of the transversal line capacitance because the corona effect. In this chapter, the R_{COND} value is 2.54 cm. The simulations are carried out for some relations between V and V_C. The V value is 1 pu for all following shown simulations.

Considering Figures 27-31, the corona effect is related to the 10 π circuits in the middle line. It corresponds to the 500 m of the represented line. Figures 27-30 show results for different relative values of the corona voltage when compared to the line nominal voltage. In Figure 31, it is shown the comparisons among the results for the voltage values in the receiving end terminal. So, using a simple routine based on π circuits for transmission line representation, undergraduate students can analyze and simulate traveling wave phenomena in transmission lines.

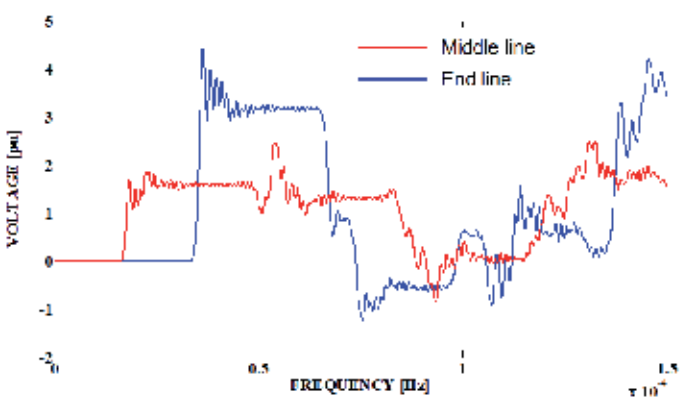


Figure 27. The time domain simulations with the corona effect for V_{CORONA} = 0,35 V.

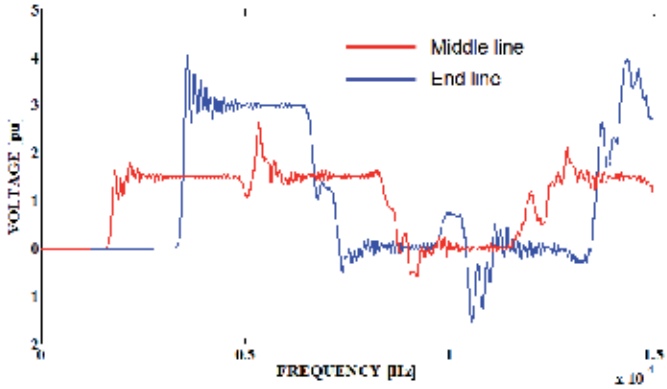


Figure 28. The time domain simulations with the corona effect for V_{CORONA} = 0,5 V.

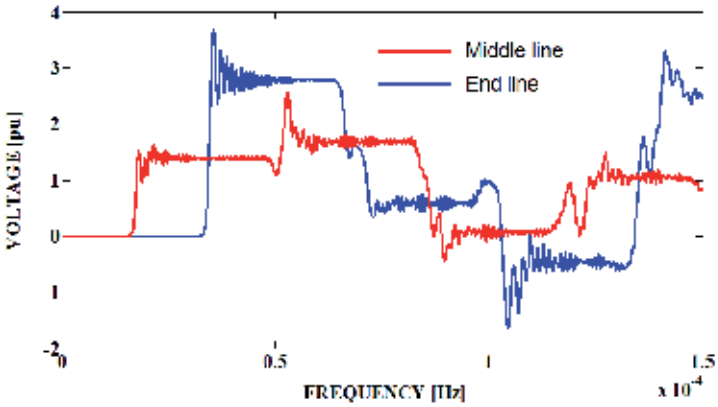


Figure 29. The time domain simulations with the corona effect for V_{CORONA} = 0,7 V.

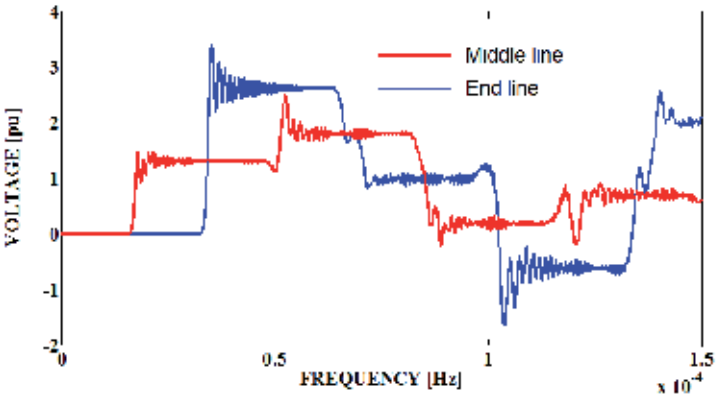


Figure 30. The time domain simulations with the corona effect for V_{CORONA} = 0,9 V.

Based on Figure 10, it's simulated the corona effect with effect of frequency for some relations between V and V_C. The V value is 1 pu to the following shown simulation.

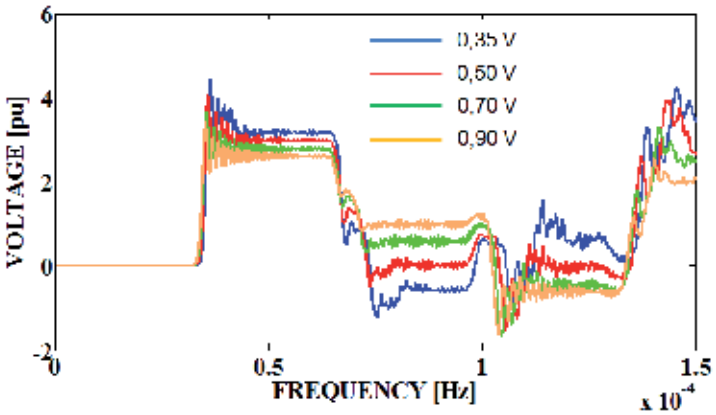


Figure 31. Comparisons of the time domain simulations with the corona effect for some V_{CORONA} values.

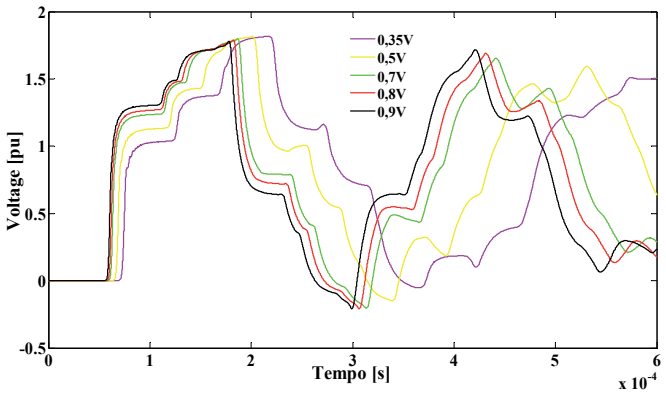


Figure 32. Comparisons of the time domain simulations with the corona effect with effect of frequency for some V_{CORONA} values.

10. Conclusions

It is described the application the MatLab™ software in analysis and simulations of transient phenomena in transmission lines. Using the characteristics of this software, transmission lines are easily modeled as a mono-phase circuit. Transient simulations are also easily carried out. For these applications, it is used basic and simple tools of the MatLab™ software. So, this software improves the analysis of the proposed problem, because it is possible to obtain several types of the graphic results that are not available in the specific programs for transient analysis like the EMTP programs. So, it is possible to analyze the resistance and inductance values that depend on the frequency when it is considered detailed transmission line models. It is possible to analyze the application of different numeric methods for solving the differential state equations by numeric integration routines. On the other hand, with a simple model of the transmission lines and the MatLab™ software, it is possible to develop a routine that is used by undergraduate students, making easy the learning about important concepts as wave propagation, transient phenomena and transmission lines. This routine can be modified, introducing elements that are able to consider the frequency influence in the transmission line

parameters. These parameters have their characteristics distributed along the line and this is considered in the mentioned routine.

The shown analysis and results can be used by undergraduate students for learning about the important concepts of power systems, transmission lines and wave propagation, for example. Related to the graduated students, it can be used for analyzing transient phenomena, developing transmission line models, improving numeric routines and comparing different numeric integration methods. So, the MatLab™ software is an excellent tool for basic and profound studies of transient phenomena in transmission lines.

Author details

Leonardo da S. Lessa, Afonso J. Prado, Rodrigo Cleber Silva,
Sérgio Kurokawa and Luiz F. Bovolato
Transient Electromagnetic Studies Laboratory, Department of Electrical Engineering,
UNESP – The University State of São Paulo, Brazil

José Pissolato Filho
Department of Electrical Engineering, UNICAMP – The State University of Campinas,
Campinas, Brazil

11. References

Chipman, R. A. Teoria e problemas de linhas de transmissão. São Paulo: Mc Graw Hill do Brasil, 1976. 276p.

Dommel H.W., Electromagnetic Transients Program. Reference Manual (EMTP Theory Book), Bonneville Power Administration, Portland, 1986.

Dommel, H.W. Digital computer solution of electromagnetic transients in single and multiphase networks. IEEE Trans. On Power App. And Systems, v. PAS-88, n.4, p. 388- 399, 1969.

Faria, A.B.; Washington, L.A.; Antônio, C.S. Modelos de linhas de transmissão no domínio das fases: estado da arte. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 14, 2002, Natal. Anais... Natal: [s.n.], 2002. p. 801-806.

Fuchs, R. D. Transmissão de energia elétrica: linha aéreas; teoria das linhas em regime permanente, 2.ed. Rio de Janeiro: Livros Técnicos e Científicos, 1979. 582 p.

Greenwood, A. Electrical transients in power systems. New York: John Wiley&Sons, 1971. 558 p.

Hedman, D. E. Teorias das linhas de transmissão-II. 2.ed. Santa Maria: Edições UFSM, 1983. v. 2 e 3.

J. R. Marti, "Accurate modelem of frequency-dependent transmission lines in 105 electromagnetic transient simulations", IEEE Trans. on Power Apparatus and Systems, vol. PAS-101, nº 1, pp. 147-155, January, 1982.

Kurokawa, S.; Yamanaka, F. N. R.; Prado, A. J.; Pissolato Filho, J.. Using state-space techniques to represent frequency dependent single-phase lines directly in time domain. In: THE 2008 IEEE/PES Transmission and Distribution Conference and Exposition: Latin America, 2008, Bogotá. Proceedings, Bogotá:[s.n.], 2008. p. 312-316,.

Kurokawa, S.; Yamanaka, F. N. R.; Prado, A. J. Representação de linhas de transmissão por meio de variáveis de estado levando em consideração o efeito da freqüência sobre os parâmetros longitudinais. Sba Controle& Automação, Campinas, v.18, n.3, p.337- 346, 2007.

Kurokawa, S.; Yamanaka, F. N. R.; Prado, A. J.; Pissolato, J. Representação de linhas de transmissão por meio de variáveis de estado considerando o efeito da frequência sobre os parâmetros longitudinais. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA- CBA, 16, 2006, Salvador. Anais... Salvador: [s.n.], 2006. v. 1 p. 268-273,

Mácias, J. A. R.; Expósito A. G.; Soler, A. B. A Comparison of techniques for state- space transient analysis of transmission lines. IEEE Transactions on Power Delivery, [S.l.], v.20, n.2, p. 894-903, 2005.

Mamis, M. S. Computation of electromagnetic transients on transmission lines with nonlinear components. IEE. Proc. General Transmission and Distribution, [S.l.], v.150, n.2, p. 200-203, 2003.

Mamis, M. S. State-space transient analysis of single-phase transmission lines with corona. In: INTERNATIONAL CONFERENCE ON POWER SYSTEMS TRANSIENTS- IPST, [s.n.], 2003, New Orleans. Anais New Orleans: [S.l.], 2003. 5p.

Mamis, M. S.; Nacaroglu, A. Transient voltage and current distributions on transmission lines. IEE. Proc. General Transmission and Distribution., [S.l.], v.149, n. 6, p. 705-712, 2003.

Martí, L. Simulation of transients in underground cables with frequency-dependent modal transformation matrices. IEEE Transactions on Power Delivery, [S.l.] , v. 3, n.3, p.1099-1110, 1988.

Nelms, R. M.; Sheble, G. B.; NEWTON, S. R.; GRIGSBY, L. L.; Using a personal computer to teach power system transients. IEEE Transactions on Power Systems, [S.l.] v. 4, n. 3, p. 1293-1297, 1989.

Swokowski, E.W. Cálculo com geometria analítica. São Paulo: Ed. Makron do Brasil, 1994. v. 2, 792p.

Tavares, M. C.; Pissolato, J.; Portela, M. C. Quasi-modes multiphase transmission line model, Electric Power Systems Research, [S.l.],v. 49, n. 3, p. 159-167, 1999.

Yamanaka, F. N. R.; Kurokawa, S.; Prado, A. J.; Pissolato, J.; Bovolato, L. F. Analysis of longitudinal and temporal distribution of electromagnetic waves in transmission lines by using state-variable techniques. In: SIXTH LATIN-AMERICAN CONGRESS ON ELECTRICITY GENERATION AND TRANSMISSION, 16, 2005, Mar del Plata. Proceeding... Mar del Plata: [s.n.], 2005. p. 1-7.

Chapter 19

Using a Low Complexity Numeric Routine for Solving Electromagnetic Transient Simulations

Rafael Cuerda Monzani, Afonso José do Prado, Sérgio Kurokawa, Luiz Fernando Bovolato and José Pissolato Filho

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/48507

1. Introduction

Different methods can be applied to accomplish the analysis of transmission lines. Many mathematical tools can be used, the main tools used are: circuits analysis with the use of Laplace or Fourier Transform, State Variables and Differential Equations. These tools can be included in a numeric routine in order to obtain voltage and current values in simulation of electromagnetic transients, at any point of the circuit.

The EMTP (ElectroMagnetic Transient Program) [1] is the main kind of this software. The prototype was developed in 60's by professionals of power system area led by Dr. Hermann Dommel (University of British Columbia, in Vancouver, B.C., Canada), and Dr. Scott Meyer (Bonneville Power Administration in Portland, Oregon, U.S.A.). Currently, EMTP is the basis of electromagnetic transients simulations in power systems.

With EMTP type programs, the following analysis can be done: simulation of switching and lightning surges, transient and temporary overvoltage, electrical machines, resonance phenomena, harmonics, power quality and power electronics applications. The most known programs of EMTP type are:

- MicroTran Power Systems Analysis of the University of British Columbia, Vancouver, Canada, founded in 1987 by: Hermann W. Dommel, Jose R. Marti (University of British Columbia), and Luis Marti (University of Western Ontario, Hydro One Networks Inc.).
- PSCAD®, also known as PSCAD®/EMTDC™ of Manitoba HVDC Research Centre. Commercially available since 1993, PSCAD® is the result of continuous research and development since 1988.
- ATP has been continuously developed through international contributions by Drs. W. Scott Meyer and Tsu-huei Liu, the co-Chairmen of the Canadian/American EMTP User

Group. The birth of ATP dates to early in 1984. For the free software, however there are some rules for using it.

This kind of program, in general, doesn't present an easy interface where data can be included [5]. Thus, many times, undergraduate students may not be interested in initiateing works in this area because the transmission lines studies involve complex models and numeric routines which are truly complex if earth effect and line parameters are considered to be frequently dependent ones.

The objective of this chapter is to introduce concepts about power systems, more especifically, in transmission lines, considering a simplified model of monophasic line in order to analyze electromagnetic transients. [2]-[6].

Considering this purpose, a transmission line can be represented as a monophasic circuit and modeled by circuits (Fig. 1). State variables are used to represent this model. The obtained linear system can be solved by using trapezoidal integration techniques.

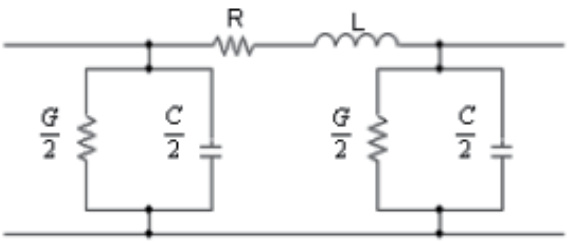


Figure 1. π circuit

Based on these conditions, a simplified numeric routine for a first contact of undergraduate students with the study of travelling waves was obtained. This numeric routine can led to a satisfactory precision and accuracy for the simulation of electromagnetic transients for a monophasic line transmission representation. The numeric routine was developed with the use of MatLab™.

2. Mathematical model

In order to equate the linear system Kirchhoff's Laws should be used. Nodal analysis is used to calculate the algebraic sum of the currents. Given that the current in the capacitor is defined as the time derivative of the voltage multiplied by the capacitance factor, the first state equation is obtained. Mesh analysis is used to calculate the algebraic sum of the voltages, the voltage across the inductor is defined as the time derivative of the current multiplied by the inductance factor, the second state equation is obtained.

A linear system can be described algebraically by using state equations variables as it follows:

$$\dot{x} = Ax + Bu \tag{1}$$

where: x – vector of state variables;
u – vector of linear system entries;
A and B – matrices which feed the system.

The solution of this system can be obtained numerically by trapezoidal rule.

$$x[k+1] - x[k] = \frac{T}{2}(Ax[k+1] + Bu[k+1] + Ax[k] + Bu[k]) \tag{2}$$

On the other hand, to solve the linear system of equations of state variables above, an interactive method can be used, where T is the integration step applied for the system solution. For time domain simulations, the integration step is a time step.

Rearranging equation (2), one can obtain:

$$x[k+1] = x[k] + \frac{T}{2}(Ax[k+1] + Bu[k+1] + Ax[k] + Bu[k]) \tag{3}$$

Solving this equation by using numeric methods, the equation can be rewritten as:

$$\left[I - \frac{T}{2}A\right]x[k+1] = \left[I + \frac{T}{2}A\right]x[k] + \frac{T}{2}B\left[u[k] + u[k+1]\right] \tag{4}$$

Verifying the equation (4), there are some constant terms, thus the equation can be simplified to:

$$A'x[k+1] = A''x[k] + B'\left[u[k] + u[k+1]\right] \tag{5}$$

A′, A′′ e B′ are constant matrices and can be described by the following equations:

$$\begin{aligned} A' &= \left[I - \frac{T}{2}A\right] \\ A'' &= A' \cdot \left[I + \frac{T}{2}A\right] \\ B' &= A' \cdot \frac{T}{2}B \end{aligned} \tag{6}$$

In these equations, I is eye matrix of order$(2nx2n)$, n is the number of π circuits. A a matrix is based on a cascade of π circuits. B is a matrix that inserts the entry values of the system. If the entrance signal is a voltage source, the inductor will be the component introduced in B

matrix for the correspondent π circuit. If the entrance signal is a current source, the element will be a capacitor.

The line transmission model for analysis without frequency influence over longitudinal parameters is based on a cascade of π circuits, which is constituted of a branch of resistor and inductor in series with a branch of capacitor and conductor in parallel. The precision of system is higher as the amount of π circuits increase.

For the transmission line represented in Fig. 2, the A matrix is described in Eq. 7. In this case, the A matrix has a special format; it's a sparse matrix, with elements non null in the three diagonals as seen below.

$$A = \begin{bmatrix} -\frac{G}{C} & -\frac{2}{C} & 0 & \dots & \dots & 0 \\ \frac{1}{L} & -\frac{R}{L} & -\frac{1}{C} & 0 & \ddots & \vdots \\ 0 & \frac{1}{C} & -\frac{G}{C} & -\frac{1}{L} & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & \frac{1}{L} & -\frac{R}{L} & -\frac{1}{C} \\ 0 & \dots & \dots & 0 & \frac{2}{C} & -\frac{G}{C} \end{bmatrix} \tag{7}$$

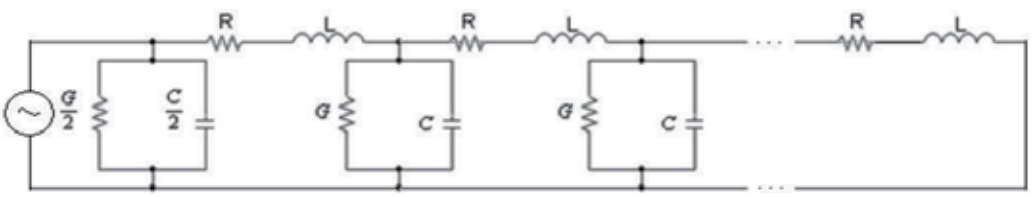


Figure 2. Transmission Line Model

In the main diagonal of the A matrix, there are the negative values of the parameters alternating between $-G/C$ and $-R/L$. In the upper diagonal, the element values alternate between $-1/C$ and $-1/L$. Considering the lower diagonal the element values are positive and similar to the upper diagonal.

The A matrix is a matrix of order ($2n\ x\ 2n$), if the transmission line is considered opened for sending and receiving in the terminal of the line.

On the other hand, the modeling of frequency influence over longitudinal parameters can be done when introduced branches in parallel of resistor and inductor associated in series with the main branch in series of an inductor and a resistor, in each π circuit [5]. The Fig. 3 shows this description.

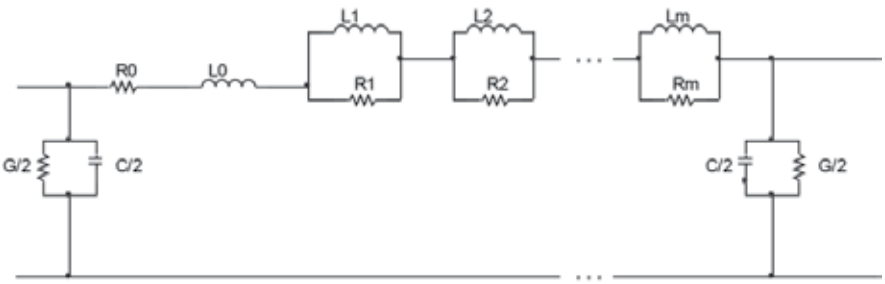


Figure 3. π circuit considering frequency dependency

The parameters R_0, L_0 refers to resistance and inductance values in series, respectively. The parametersR_m, L_m, with m being the number of branches in series, refers to value of resistance inductance of each one of the m branches in series of each π circuit, respectively. The parameters C and G, refer to the value of capacitance and conductance, respectively.

From analysis of Kirchhoff's Laws of the current in the inductor and the voltage in the capacitor, the linear system of state variables is obtained.

$$\begin{aligned}
\frac{d\,i_{10}}{dt} &= \frac{i_{10}}{L_0}\left(-\sum_{j=1}^{m} R_j\right) + \frac{1}{L_0}\left(\sum_{j=1}^{m} R_j i_{1j}\right) + \frac{1}{L_0}u(t) - \frac{1}{L_0}v_1(t) \\
\frac{di_{11}}{dt} &= \frac{R_1}{L_1}i_{10} - \frac{R_1}{L_1}i_{11} \\
\frac{di_{12}}{dt} &= \frac{R_2}{L_2}i_{10} - \frac{R_2}{L_2}i_{12} \\
\frac{di_{1m}}{dt} &= \frac{R_m}{L_m}i_{10} - \frac{R_m}{L_m}i_{1m} \\
\frac{dv_1(t)}{dt} &= \frac{2}{C}i_{10} - \frac{G}{C}v_1(t)
\end{aligned} \tag{8}$$

n is the number of π circuit and m is the number of series branch. In order to represent these equations, the A matrix is square of order $(m+2)$ and it's described by:

$$A = \begin{bmatrix} [A_{11}] & [A_{12}] & \cdots & [A_{1n}] \\ [A_{21}] & [A_{22}] & \cdots & [A_{2n}] \\ \vdots & \vdots & \ddots & \vdots \\ [A_{n1}] & [A_{n2}] & \cdots & [A_{nn}] \end{bmatrix} \tag{9}$$

where, each A_{kk} matrix is represented by:

$$A' = \begin{bmatrix} -\frac{\sum_{j=0}^{j=m} R_j}{L_0} & \frac{R_1}{L_0} & \frac{R_2}{L_0} & \cdots & \frac{R_m}{L_0} & -\frac{1}{L_0} \\ \frac{R_1}{L_1} & -\frac{R_1}{L_1} & 0 & \cdots & 0 & 0 \\ \frac{R_2}{L_2} & 0 & -\frac{R_2}{L_2} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ \frac{R_m}{L_m} & 0 & 0 & \cdots & -\frac{R_m}{L_m} & 0 \\ \frac{2}{C} & 0 & 0 & \cdots & 0 & -\frac{G}{C} \end{bmatrix} \tag{10}$$

For n π circuits, the A matrix is formed in the diagonal line by A' matrices, in the upper diagonal these matrices have just one element not null which is in the first column and in the last row and it's described by $-1/C$.

The lower matrices contain just one non null element which is in the last column in first line and it's described by$1/L_0$. Both matrices, upper and lower are $(m+2)$ order one. The B vector is an $n(m+2)$ order with the first element non null described by$1/L_0$. A generic vector x is shown below:

$$x^T = [x_1 \cdots x_n] \tag{11}$$

Each element of x has the following structure:

$$x_k^T = [i_{k0}\ i_{k1}\ \cdots\ i_{km}\ v_{kn}] \tag{12}$$

This state equation describes the transmission line represented by n π circuits, by using numeric methods.

3. Routine development

The routine used for introducing the proposed model shown in previous items is implemented in MatLab™ software. For this, only basic notions of programming are necessary to make the development of this routine easy for undergraduate students. Initially, the source values, the number of π circuits, the line length and the line parameters per unit length in the numeric routine are introduced. By using the line parameters per unit length, the parameter values for each π circuit it is determined by using the following definitions:

$$R = \frac{R' \cdot d}{n} \tag{13}$$

$$L = \frac{L' \cdot d}{n}$$

$$C = \frac{C' \cdot d}{n}$$

$$G = \frac{G' \cdot d}{n}$$

Where R′, L′, C′, G′ are the line parameters per unit length, d is the length of the line and n is the number of π circuits. Fig. 4 shows a window program that applies the proposed numeric routine.

In the case of frequency influence the user may choose the number of branches and also specify the value for each resistor and inductor. These values can be calculated by using any routine that considers the frequency influence in transmission line parameters. After this step, the simulation time (t), the time step (T) and the sources that are connected at the line are defined. Then, the equation that describes each source should be introduced in the numeric routine by using a specific MatLab™ tool. Simulations of electromagnetic transient of many different input signals can be obtained just by inserting their functions in MatLab™ program routine as seen in Fig. 4. After specifying all the input values, the routine generates the A and B matrices and also the input vector U in discrete time. Then, the equation (6) terms are calculated numerically by using the chosen time step and the simulation time. The chosen current and voltage output results are uploaded in a vector file. Finally, a graph is plotted with the results of the simulation.

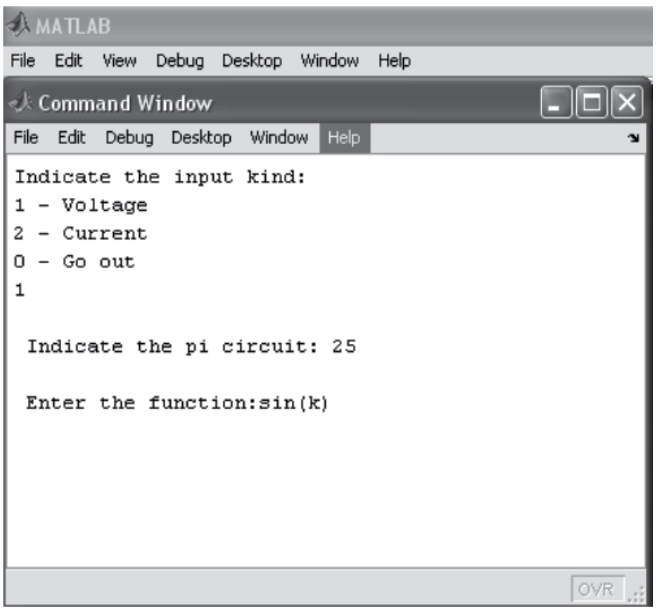


Figure 4. MatLab window.

In order of results, both routines, with and without frequency influence, will be shown and discussed step by step. The objective of these routines is to show how to use MatLab™ by using simple functions or commands to present the wave propagation of current or voltage to undergraduate students. The functions and commands used into routine will be explained; also the loops and conditions used into it will also be explained below the routine.

The first routine doesn't consider the frequency influence.

MatLab code

```
%Clear variables and command window
clear;
clc;
%Close windows
Close all;
%Definitions of Entrance Elements
disp('Transmission Line Analyze')
P = input ('Enter the Number of Pi Circuits = ');
D = input ('Enter the line length (km) = ');
R = input ('Enter the value of distributed resistance (ohm/km) = ');
L = input ('Enter the value of distributed inductance (H/km) = ');
C = input ('Enter the value of distributed capacitance (F/km) = ');
G = input ('Enter the value of distributed conductance (S/km) = ');
T = input ('Set time step [s]: ');
t = input ('Enter the simulation total time [s]: ');
%Distributed Elements
R=R*D/P;
L=L*D/P;
G=G*D/P;
C=C*D/P;
%e = Pi circuits which will receive entries
cont=1;
while (i ~= 0)
   clc;
   disp('Indicate the input kind:');
   disp('1 - Voltage');
   disp('2 - Current');
   disp('0 - Exit');
   i = input('');
   if (i==1)
      e(cont)= input ('\n Indicate the pi circuit = ');
      fun = input('\n Enter the function:');
      fun = inline(fun);
```

```
        u(2*e(cont)-1,1) = fun;
    end
    if (i==2)
        e(cont)= input ('\n Indicate the pi circuit= ');
        fun = input('\n Enter the function:');
        fun = inline(fun);
        u(2*e(cont),1) = fun;
    end
    cont=cont+1;
end
%A matrix
for j=1:(2*P-1)
    h = rem(j,2);
    if h==1
        A(j,j)= -(R/L);
        A(j,j+1)= -(1/L);
        A(j+1,j)= 1/C;
    else
        A(j,j)= -(G/C);
        A(j,j+1)= -(1/C);
        A(j+1,j)= 1/L;
    end
end
%A (2*P,2*P)
A(2*P,2*P)=-(G/C);
%B matrix
B(2*P,1)=0;
for j=1:numel(u)
    h = rem(j,2);
    if h==1
        B(j,j)=1/L;
    else
        B(j,j)=1/C;
    end
end
%Constant terms
A1=inv(eye(2*P)-(T/2)*A);
A2=(eye(2*P)+(T/2)*A);
A2=A1*A2;
B1=(T/2)*B;
B2=A1*B1;
x(2*P,1)=0;
%Iterations to solve the linear system
```

```
j=1;
for q= 0: T: t
  u1=feval(u,q);
  u2=feval(u,q+T);
  x=A2*x+B2*(u1+u2);
  y(j,1)=q;
  y(j,2)=x(2*P,1);
  y(j,3)=x(2*P-1,1);
  j=j+1;
end
plot(y(:,1),y(:,2),'b')
title('Voltage in the end of transmission line');
ylabel('Voltage [kV]');
xlabel('Time [s]');
pause
plot(y(:,1),y(:,3),'r')
title('Current in the end of transmission line');
ylabel('Current [A]');
xlabel('Time [s]');
```

In the beginning of the routine some commands are used like:

- **clear all:** cleans all the variables existents in MatLab™.
- **clc:** cleans the command window.
- **close all:** closes all the graphics opened.

The command **disp** shows a message in the command window for users to know what happens in the routine, that way no one thinks the routine is not working, because sometimes, it does take a lot of time to finish all the procedures, thus, a message is important to enlighten that.

The command **input** asks the user to insert a value for a variable, instead of defining a constant value inside the routine, this command makes it more interactive, so the user can put any value wanted and analyze the answer for different values inserted.

The loop **while** is used with the purpose to insert voltage or current sources as many as the user wants. The user will specify if the source is of voltage (1) or current (2), after specifying the kind of source, it shall specify the π circuit that will receive the source, and finally specify the function that represents the source. A loop while is used, to mount a vector of entries. The user will get out of the loop inserting the value (0). The function of entry (voltage or current) must be inserted as a string, as explained above. At least one source for the routine works correctly is necessary, as an example, the user can insert a step function in the first circuit. Entering the following:

Indicate the input kind:

1 – Voltage
2 - Current
0 - Exit
1
Indicate the pi circuit = **1**
Enter the function: **'1'**

For mounting the A matrix a loop **for** is used because the number of interactions is known, inside the loop the command **if** is used to construct the mainly, upper and lower diagonals as shown in Eq. 7. The same is done in the B matrix.

Finally, a loop for is used to solve the trapezoidal rule considering the time step and the total time elapsed. The variable y retains in the first column the value of steps of time, the second column retains the value of voltage, and the third column retains the value of current in the terminal of the line transmission.

Fig. 4 shows the previous version of the routine, in that routine the command **syms** was used. This command creates a symbol as a variable to solve this. The command **subs** must be used, in order to substitute the variable in the function with the value requested. In this case, the function entry of current or voltage source didn't need to be inserted as a string, but the user always had to insert the function by using the variable specified, what was not always done. The second purpose used the command **inline** which gets a string and converts it into a function; with this command the user can use any variable, to solve this in the trapezoidal rule. It's used the command **feval**, which gets the function and substitutes the values with the time step.

The second routine considers the frequency influence. Almost all the steps used in this routine were the same as shown in the first one. Only the different steps done here will be described.

MatLab code

```
%Clear variables and command window
clear all;
clc;
%Close all graphic windows
close all;
disp('Transmission Line Analysis');
sprintf('\n');
P = input ('Enter the Number of Pi Circuits = ');
D = input ('Enter the line length (km) = ');
%It's defined R(1) and L(1) which represents R0 and L0 respectively,
%values of R1, L1, ... Rm, Lm, will be with a plus number.
R(1) = input ('Enter the value of distributed resistance (ohm/km) = ');
```

```
L(1) = input ('Enter the value of distributed inductance (H/km) = ');
C = input ('Enter the value of distributed capacitance (F/km) = ');
G = input ('Enter the value of distributed conductance (S/km) = ');
m = input ('Enter the number of coupling circuits = ');
for i=2:m+1
   R(i) = input (['Enter the value of resistance of the branch ' int2str(i-1) ' [ohm/km] = ']);
   %Distributed Resistance
   R(i) = R(i)*D/P;
   L(i) = input (['Enter the value of inductance of the branch ' int2str(i-1) ' [H/km] = ']);
   %Distributed Inductance
   L(i) = L(i)*D/P;
end
%Time step and total time
T = input ('Set time step [s]: ');
t = input ('Enter the simulation total time [s]: ');
%Distributed Elements
R(1)=R(1)*D/P;
L(1)=L(1)*D/P;
G=G*D/P;
C=C*D/P;
%A matrix
A=zeros(size(P*(m+2)));
c=0;
for i=1:m+1
   A(i,i)=-R(i)/L(i);
   A(1,i)=R(i)/L(1);
   A(i,1)=-A(i,i);
end
%First term as positive
A(1,1) = -A(1,1);
for i=1:m+1
   %c variable will get the values to put in A(1,1), uses recall
   c=A(1,i)+c;
end
A(1,1)=-c;
%Terminal elements of matrix
A(1,m+2)=-1/L(1);
A(m+2,1)=1/C;
A(m+2,m+2)=-G/C;
```

```
%Mainly matrix
AP=A;
%Putting the matrix in all the positions
for j=1:(P-1)
   AP(j*(m+2)+1:(j+1)*(m+2),j*(m+2)+1:(j+1)*(m+2))=A;
end
%Lower matrix
AI(1,m+2)=1/L(1);
AI(m+2,m+2)=0;
%Upper matrix
AS(m+2,1)=-1/C;
AS(m+2,m+2)=0;
%Putting the lower and upper matrices in their places
for j=1:(P-1)
   AP(j*(m+2),j*(m+2)+1)=-1/C;
   AP(j*(m+2)+1,j*(m+2))=1/L(1);
end
%Element of the first matrix
A(m+2,1)=2/C;
%Element of last matrix
A(P*(m+2),1)=2/C;
%e = Pi circuits which will receive entries
%cont sets which pi circuit will receive the entry
cont=1;
while (i ~= 0)
   clc;
   disp('Indicate the input kind:');
   disp('1 - Voltage');
   disp('2 - Current');
   disp('0 - Exit');
   i = input('');
   if (i==1)
      e(cont)= input ('\n Indicate the pi circuit = ');
      fun = input('\n Enter the function:');
      fun = inline(fun);
      u(e(cont)*(m+2)-(m+1),1) = fun;
   end
   if (i==2)
      e(cont)= input ('\n Indicate the pi circuit = ');
```

```
        fun = input('\n Enter the function:');
        fun = inline(fun);
        u(e(cont)*(m+2),1) = fun;
    end
    cont=cont+1;
end
%Clear the command window
clc;
disp('PROCESSING...');
%B matrix
B(P*(m+2),1)=0;
for j=1:m+2:numel(u)
    h = rem(j,2);
    if h==1
        B(j,j)=1/L(1);
    else
        B(j,j)=1/C;
    end
end
A1=inv(eye(P*(m+2))-(T/2)*AP);
A2=(eye(P*(m+2))+(T/2)*AP);
A2=A1*A2;
B1=(T/2)*B;
B2=A1*B1;
%x matrix
x(P*(m+2),1)=0;
j=1;
for q = 0: T: t
   u1=feval(u,q);
   u2=feval(u,q+T);
   x=A2*x+B2*(u1+u2);
   y(j,1)=q;
   y(j,2)=x(P*(m+2),1);
   y(j,4)=x(P*(m+2)/2,1); %middle of line
   y(j,3)=x(P*(m+2)-(m+1),1);
   y(j,5)=x((P*(m+2)/2)-(m+1),1); %middle of line
   if (q==30*T)
      for kk = 1:P
         z(kk,1)=kk*D/P;
```

```
        z(kk,2)=x(kk*(m+2),1);
    end
end
if (q==60*T)
    for kk = 1:P
        z(kk,3)=x(kk*(m+2),1);
    end
end
if (q==90*T)
    for kk = 1:P
        z(kk,4)=x(kk*(m+2),1);
    end
end
if (q==120*T)
    for kk = 1:P
        z(kk,5)=x(kk*(m+2),1);
    end
end
if (q==150*T)
    for kk = 1:P
        z(kk,6)=x(kk*(m+2),1);
    end
end
if (q==180*T)
    for kk = 1:P
        z(kk,7)=x(kk*(m+2),1);
    end
end
if (q==210*T)
    for kk = 1:P
        z(kk,8)=x(kk*(m+2),1);
    end
end
if (q==240*T)
    for kk = 1:P
        z(kk,9)=x(kk*(m+2),1);
    end
end
if (q==260*T)
```

```
    for kk = 1:P
        z(kk,10)=x(kk*(m+2),1);
    end
  end
  if (q==290*T)
    for kk = 1:P
        z(kk,11)=x(kk*(m+2),1);
    end
  end
  if (q==310*T)
    for kk = 1:P
        z(kk,12)=x(kk*(m+2),1);
    end
  end
  if (q==330*T)
    for kk = 1:P
        z(kk,13)=x(kk*(m+2),1);
    end
  end
  if (q==360*T)
    for kk = 1:P
        z(kk,14)=x(kk*(m+2),1);
    end
  end
  if (q==390*T)
    for kk = 1:P
        z(kk,15)=x(kk*(m+2),1);
    end
  end
  j=j+1;
end
plot(y(:,1),y(:,2),'b')
title('Voltage in the end of transmission line');
ylabel('Voltage [kV]');
xlabel('Time [s]');
pause
plot(y(:,1),y(:,3),'r')
title('Current in the end of transmission line');
ylabel('Current [A]');
```

```
xlabel('Time [s]');
pause
plot(z(:,1),z(:,2),'b',z(:,1),z(:,5),'r',z(:,1),z(:,11),'g',z(:,1),z(:,15),'k')
title('Voltage in line path');
ylabel('Voltage [kV]');
xlabel('Length (km)');
```

This routine uses the loop **for** in the beginning to obtain all the values series branches. The user specifies the amount of series branches and enters the resistance and inductance values for each branch. In the last loop **for**, a sequence of **if** is used to obtain the wave propagation in different time instants. This is a very important part of the routine, because it shows how the voltage or current waves propagates into the line. This representation shows to undergraduate students that a signal put in the beginning of line will not appear instantaneously in the end of line. It takes a little time, like milliseconds to arrive at the end, because the length of the line is considered, differently from a bipole, the signal put in a terminal is at the other terminal at the same time. Another observation is that, with the EMTP type programs, the user can only analyze a specific point of the circuit and in this case in a time range. The routine shows how the wave propagates into the line for different line points in the time instant.

4. Obtained results

For all simulations the following values were used: the number of π circuits was 100 and the transmission line has 10 kilometers. The resistance value was $0.05\ \Omega/Km$. The inductance value was $1\ mH/Km$. The capacitance values was $11.11\ nF/Km$. The conductance value was $0.556\ \mu S/Km$. The time step was $50\ ns$ and the period of simulation was $600\ \mu s$.

Resistors (Ω)		Inductors (mH)	
R_0	0,026	L_0	2,209
R_1	1,470	L_1	0,74
R_2	2,354	L_2	0,12
R_3	20,149	L_3	0,10
R_4	111,111	L_4	0,05

Table 1. Longitudinal parameters values using the routine considering the frequency influence

A simulation was made for voltage input unitary step signal. It was a step function with a unitary step after $t = 0s$. The result of this simulation can be seen in Fig. 5. the voltage output at the receipting line end is shown. By using the routine without frequency influence, from the results of Fig. 5, it is observed that there is a period time related to the propagation time of the signal through the line. So, it represents a time delay between the input signal and the output signal. After the time delay, there are oscillations associated to wave reflections on the sending and receipting line ends that compose the shown voltage output.

By using the routine with frequency influence at longitudinal parameters, it is obtained Fig. 6. In this figure, it is noticed that are not so many transients as in Fig. 5, because in this case the series branches dampen the signal.

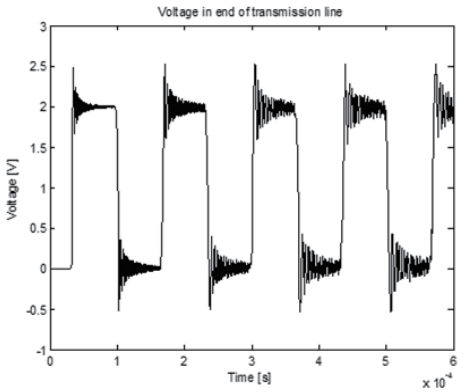


Figure 5. Voltage in the end of transmission line by using routine without frequency influence.

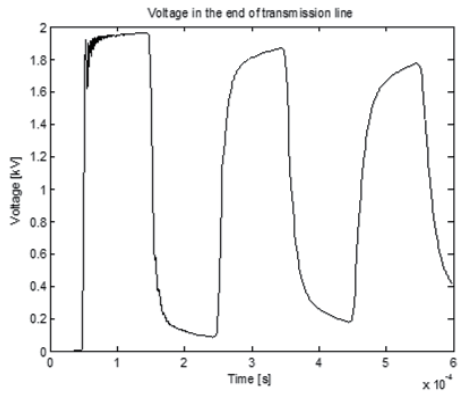


Figure 6. Voltage in the end of transmission line by using routine with frequency influence

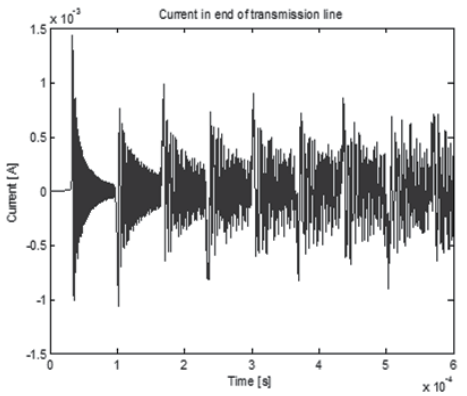


Figure 7. Current in the end of transmission line by using routine without frequency influence.

In Figs. 7 and 8, it's possible to observe the influence of frequency in the current. Without frequency influence the signal is very disturbed. On the other hand, when it is used the routine that considers frequency influence, it is noticed that in the end of the line it should not have any current, supposing an end line opened, but, there are some transients because of the last branch. As in voltage signal, the current signal is damped because of the series branches in circuit. By using the routine of frequency influence, it can also be obtained how the voltage signal goes to the path of line for the time in Fig. 9.

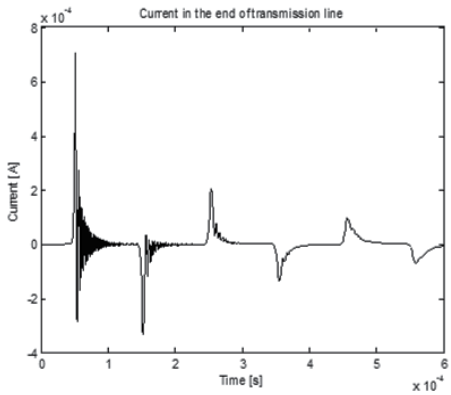


Figure 8. Current in the end of transmission line by using routine with frequency influence.

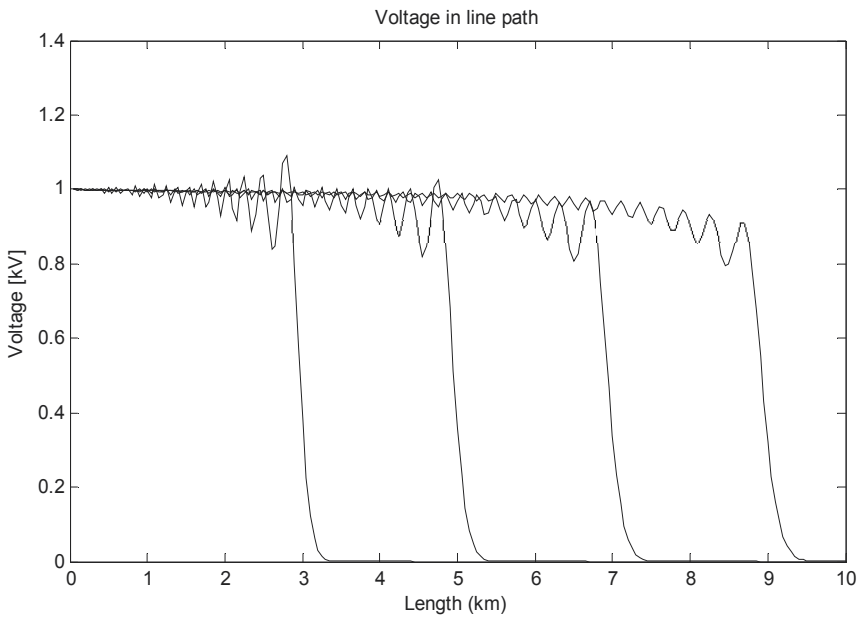


Figure 9. Voltage in line path.

5. Conclusions

By using a mono-phase circuit representation of transmission lines associated to the state variables and the trapezoidal rule, π circuits are applied for electromagnetic transient simulations. The π circuits represent the mono-phase circuit of transmission lines and considering no frequency influence, they are included in a numeric routine through a sparse matrix where only three diagonal lines have non-null elements. Considering the frequency dependent longitudinal line parameters, the parameters are included in a matrix, by using the first line, the first column and the main diagonal of this matrix. This matrix is introduced in another large matrix, rounded by other matrices with one non-null element each. In the upper matrix, the parameter is introduced$-1/C$. In the lower matrix, the parameter is introduced$1/L_0$. The obtained linear system, which is based on state variables, it is solved by using trapezoidal rule integration method. The numeric solution uses the trapezoidal rule method, which solves the numeric integration by the addition of infinitesimal trapeziums areas. This method increases the accuracy of the calculation, comparing with the Euler method for the same time step. The numeric routine is applied to a mathematical matricial program and, because of this, it is possible to simulate the propagation of electromagnetic transients on transmission lines. Using MatLab™ software, it is possible to describe the input signal through mathematical functions that are included as an initial datum of the numeric routine. It is not included in the structure of the routine, but during the routine running. By using the mentioned routine, examples of a unitary step were applied as voltage input signal and some electromagnetic transients generated from these signals are shown in this chapter. The numeric routine permits the inclusion of any function describing the voltage or current source that represents the simulated transient. The included function can be located on any point of the represented transmission line.

The shown numeric routine is simple and because of this, it can be used by undergraduate students for their first contact with electromagnetic transient phenomena, by traveling wave propagation, transmission line analyses. For the first contact with wave propagation simulations for undergraduate students, the proposed routine is an excellent tool. It is simple and its use is easy. So, by using basic concepts of traveling wave, linear systems and computing, it is possible to manipulate the numeric routine, observing the wave propagation characteristics, such as time delays, wave reflection and refraction, numeric oscillations (Gibbs' oscillations), transient oscillations. For undergraduate students, it is possible to compare the results and the modeling of the electromagnetic transients in transmission lines considering or not frequency dependent line parameters. This is possible by accessing the proposed routine numeric code. In courses related to the transmission line area, the manipulation of this code can make the course more interesting for the undergraduate students. These advantages are added to the other ones that are shown in this chapter and related to the application of the MatLab™ software. The use of this software makes the implementation of the mentioned routine extremely easy.

Author details

Rafael Cuerda Monzani, Afonso José do Prado,
Sérgio Kurokawa and Luiz Fernando Bovolato
Department of Electrical Engineering at FEIS/UNESP – The University of São Paulo State, Brazil

José Pissolato Filho
Department of Electrical Engineering, DSCE/UNICAMP – The State University of Campinas, Brazil

6. References

[1] Microtran Power System Analysis Corporation, Transients Analysis Program Reference Manual, Vancouver, Canada, 1992.

[2] R. M. Nelms, Steven, R. Newton, G. B. Sheble, L. L. Grigsby. "Simulation of Transmission Line Transients", IEEE.

[3] R. M. Nelms, Steven, R. Newton, G. B. Sheble, L. L. Grigsby. "Using a personal Computer to teach power system Transients", IEEE Transaction on Power System, Vol. 4, No. 3, August 1989.

[4] S. R. Newton, B. B. Reid, G. B. Sheble, R. M. Nelms, L. L. GRIGSBY. "Electromagnetic transient simulator for large-scale spacecraft power systems". IEEE.

[5] H. W. Dommel, EMTP Theory Book, Department of Electrical Engineering, University of British Columbia, Vancouver, 1996.

[6] J. R. Marti, "Accurate modeling of frequency-dependent transmission lines in electromagnetic transients simulations", IEEE Trans. on PAS, vol. 101, pp. 147–155, January 1982.

[7] E. Clarke, Circuit Analysis of AC Power Systems, vol. I, Wiley, New York, 1950.

[8] S. Kurokawa, F. N. R. Yamanaka, A. J. Prado, L. F. Bovolato, J. Pissolato, "Representação de Linhas de Transmissão por meio de variáveis de Estado Levando em Consideração o Efeito da Freqüência Sobre os Parâmetros Longitudinais", SBA. Sociedade Brasileira de Automática, Controle & Automação, v. 18, p. 337-346, 2007.

[9] S. Kurokawa, F. N. R. Yamanaka, A. J. Prado, J. Pissolato, L. F. Bovolato, "Utilização de variáveis de estado no desenvolvimento de modelos de linhas de transmissão: Inclusão do efeito da freqüência nas matrizes de estado", XIX Seminário nacional de produção e transmissão de energia elétrica (XIX SNPTEE), p. 1-8, Rio de Janeiro. 2007.

[10] S. Kurokawa, F. N. R. Yamanaka, A. J. Prado, J. Pissolato, "Representação de linhas de transmissão por meio de variáveis de estado considerando o efeito da freqüência sobre os parâmetros longitudinais", XVI Congresso Brasileiro de Automática, v. 1, p. 268-273, Salvador, 2006.

[11] F. N. R. Yamanaka, S. Kurokawa, A. J. Prado, J. Pissolato, L. F. Bovolato, "Analysis of longitudinal and temporal distribution of electromagnetic waves in transmission lines by using state-variable techniques", Sixty Latin-American Congress: Electricity Generation and Transmission – VI CLAGTEE, Mar Del Plata, 2005.

Edited by Vasilios N. Katsikis

This excellent book represents the final part of three-volumes regarding MATLAB-based applications in almost every branch of science. The book consists of 19 excellent, insightful articles and the readers will find the results very useful to their work. In particular, the book consists of three parts, the first one is devoted to mathematical methods in the applied sciences by using MATLAB, the second is devoted to MATLAB applications of general interest and the third one discusses MATLAB for educational purposes. This collection of high quality articles, refers to a large range of professional fields and can be used for science as well as for various educational purposes.

Photo by koya79 / iStock

IntechOpen

ISBN 978-953-51-5708-3