AN ADAPTIVE TIME-STEP CONTROL ALGORITHM FOR NONLINEAR TIME-DOMAIN ENVELOPE TRANSIENT

by

Jude Alexander

A thesis submitted to the Engineering Graduate
Faculty of Lakehead University in partial
fulfillment of the requirements for the
Degree of Master of Science in Control Engineering

Department of Electrical Engineering

Thunder Bay

2006

APPROVED BY:

Chair of Advisory Committee

Your file Votre référence
ISBN: 978-0-494-24049-6
Our file Notre référence
ISBN: 978-0-494-24049-6

Abstract

AN ADAPTIVE TIME-STEP CONTROL ALGORITHM FOR NON LINEAR TIME-DOMAIN ENVELOPE TRANSIENT (Under the supervision of Dr. Carlos Christoffersen)

This thesis outlines a general method to analyze circuits with several time variables using a technique known as Multi Partial Differential Equation, MPDE. The key idea of MPDE is to convert the system of ordinary differential equations that describes a circuit into a system of partial differential equations using multiple time dimensions. Multivariate Finite Difference Time Domain (MFDTD) and Time Domain Envelope following (TD-ENV) methods are based on the MPDE and give faster simulation time and reduce the memory requirements for a system with widely separated time scales.

In this research, a novel time-step control method in one of the time dimensions is proposed. The algorithm uses two models: the first is the set of differential algebraic equations that represent the circuit. The second is a 'coarse' model that is cheap to evaluate. The main difference between the traditional and the proposed method is the dynamic tolerance changes and coarse model representation. The optimum time step is estimated from an error term obtained from the coarse model. An estimation of the Local Truncation Error (LTE) is used to optimize the time step size. The simulations show that fewer time steps are rejected, *i.e.* faster computation, compared with a traditional time step control algorithm.

A rectifier circuit is simulated to show the difference between the conventional method and the MFDTD method for steady state analysis. The MFDTD method is used in steady state analysis and the TD-ENV method is used in transient simulations. A DC-DC converter circuit simulation using adaptive TD-ENV and its advantages are presented. Simulations of a switched rectifier circuit and DC-DC converter circuit with different controllers (P and PI) are presented. The PI controller circuit experiences a duty cycle oscillation and higher LTE, which increases the simulation time with the proposed model. The FDTD method is used to solve the problem in one of the dimensions (fast time axis) and the Backward Euler (BE) method is used on the other dimension (slow time axis).

Biographical Summary

Jude Sasiharan Alexander was born in North of Srianka, Jaffna. He worked with Celestica Inc and Canadian Instrumentation and Research Inc as an Engineer. Currently, he is working as Bioinformatics-IT Engineer at Genesis Genomics Inc. He received the Computer Engineering degree at McMaster University, Canada in 2000. In 2002, he was awarded an entrance scholarship to pursue M.Sc Engineering degree at Lakehead University. Currently he is pursuing his M.Sc Engineering in Control Engineering degree at Lakehead University, where he is working as research assistant to Dr. Carlos Christoffersen. He is a student member of the IEEE and Vice Chair at Lakehead University student Chapter during 2003-2004. His research interests include time domain simulation, machine learning, bioinformatics, computer aided analysis of circuits, RF and microwave circuit design.

Acknowledgments

I have had the distinct privilege of associating with first rate professors throughout my journey. I would like to express my gratitude to my supervisor Dr. Carlos E. Christoffersen for his support and guidance during my graduate studies. With an enthusiasm enjoyed by only a select few, Mr. Raj Singh introduced me to the grammar of problem solving in my high school, Emery CI. In completely different ways, graduate students in lakehead University constantly challenged my intuition and taught me the valuable lessons in control engineering. Many professors from Lakehead has helped me in various ways, it is hard to estimate the impact that these professors have made in my life. Last, definitely not least I want to thank to my co-supervisor, Dr. Abdelhamid Tayebi for his support throughout my graduate studies.

I dedicate this work to my parents Mr. & Mrs. Alexander, and to my fiance Quincy Gnananathan for their love, guidance and support.

Contents

List of Figures

List of Tables

List of Symbols

$MPDE$	–	Multi Partial Differential Equations.
$TD-ENV$	–	Time domain ENVelope transient method.
$MFDTD$	–	Multi Finite Difference Time Domain method.
HB	–	Harmonic Balance analysis.
h	–	Time step-size.
h_1	–	Time step-size in T_1 time axis.
h_2	–	Time step-size in T_2 time axis.
$\hat{h}$	–	Adaptive time step.
$\tilde{h}$	–	Variable time step.
$\Re()$	–	Real part.
ω	–	Angular frequency.
t	–	Time.
J	–	Jacobian matrix in MFDTD method.
n_t	–	Number of tones.
C	–	Nonlinearity order.
n_s	–	Number of state variables.
ε	–	Local Truncation Error (LTE).
ε_N	–	Normalized Truncation Error.
KCL	–	Kirchhoff's Current Law.
KVL	–	Kirchhoff's Voltage Law.

Chapter 1

Introduction

Circuit simulations are widely used. The overwhelming use of circuit abstraction in electronic engineering has enabled the design of complex systems [5]. Circuit simulations are important to understand the dynamics of complex systems of interacting elements, to test new concepts, and to optimize the designs [5]. It is a very economical way to test complex designs. In integrated circuits (ICs), it is important to verify the design's accuracy before fabrication to avoid unnecessary prototypes. IC development and simulation are interconnected to each other, if better simulation methods are used, they will indirectly improve IC development. Also the circuit designs can be optimized using circuit simulations. Millimeter-wave circuits are becoming more popular and coupled with large scale production, require more sophisticated design techniques than before.

There are three main circuit simulation techniques: frequency domain techniques, time domain techniques, and mixed (time and frequency) domain techniques. The most widespread method of nonlinear circuit analysis is time-domain analysis (transient analysis) using programs like SPICE [33]. This is a well known time domain circuit simulator. It was originally developed to assist the design of integrated circuits, where timing and waveform shape are important. SPICE use numerical integration to determine the circuit response at one instance of time, given the circuit's response at a previous instance of time.

Harmonic Balance (HB) analysis is mainly used in RF and microwave circuits. The difference between HB and other traditional time domain methods is that in time domain methods, waveforms are represented as a collection of samples, whereas in harmonic balance they are represented using the coefficients of sinusoids [3]. Thus, it approximates naturally the periodic and quasi-periodic signals found in a steady-state responses.

When simulating a circuit with a transient analysis, a set of differential equations are formed and solved. The differential equations have infinite number of solutions and it is necessary to specify a complete set of boundary conditions in order to identify the desired solution [5]. Most simulators require user-defined initial conditions. If the initial conditions are not specified, then the simulators use the DC solutions as initial conditions.

1.1 Circuit Simulators

A circuit simulator numerically computes the response of the particular circuit to a particular stimulus. To calculate the response, the simulator has to formulate the circuit equations and then solve them numerically. In a linear circuit with energy storage elements, voltages and currents are the solutions to linear, constant coefficient differential equations. Simulation techniques vary in the way of solving the differential equations.

Circuit simulators first began to appear in late 1960's [34]. Two groups contributed significantly to the development of the modern circuit simulators. First is ASTAP group at IBM, which developed many numerical methods [9]. Second is SPICE group at the University of California, Berkeley. SPICE started as a class project of Dr. Ron Rohrer. It was first released in 1972 and then in 1975 [34]. SPICE was written by Dr. Larry Nagel, under the guidance of Dr. Don Pederson. SPICE became very popular at that time because it had all the device models built in it to simulate ICs, the source code was very affordable and the graduates encouraged SPICE in their companies.

In late 80's Berkeley also released a new type of circuit simulator called SPECTRE [33]. SPECTRE used harmonic balance to directly compute the steady-state solution of nonlinear circuits in the frequency domain. SPECTRE was picked up by Hewlett-Packard, where it was known as Microwave Nonlinear Simulator (MNS).

Recently another simulator called Freeda was developed by Dr. Carlos E Christoffersen and Dr. Michael B. Steer at North Carolina State University. Freeda has various simulation techniques such as HB, transient algorithms, AC and DC. It provides flexibility to add new device models and circuit analysis algorithms. It also supports the local reference concept [7], which is fundamental to the analysis of spatially distributed circuits and also to simultaneous thermal-electrical simulations [5].

Commercial advanced simulators, implementing most of the simulation methods are also available in the current market. For example, Agilent ADS(former HP) implements circuit envelope method [10], SPECTRE RF implements envelope method, and Aplac implements MPDE [16], where MPDE has advantage on computation time and memory saving.

1.2 Importance Of Time-Step Control

In traditional time domain methods, numerical integration is used to determine the circuit response at each time instance using previous responses. As a result, these simulation techniques heavily depend on the time step size.

Stiff circuits have extreme range of operating frequencies or time scales and are difficult to simulate. Time step control is very important on stiff circuits. RF circuits have extreme range of operating frequencies or time scales. The main challenge of this research is to develop an *adaptive time step control* algorithm to simulate stiff circuits. First the system of ordinary differential equations(ODEs) of a circuit is converted into a system of partial differential equations(PDEs) using multiple time variables. The PDEs are solved using method called multivariate finite difference time domain method (MFDTD) and time-domain envelope following method (TD-ENV).

Time-step control for the numerical solution of ordinary differential equations(ODEs) is used in various applications [32]. MATLAB has various methods for solving ODEs, such as Runge-Kutta method [11],[12]. Time-step schemes are also used in to differential algebraic equations(DAEs). DAEs are systems consisting of ODEs with algebraic equations. Due to the long simulation time in stiff circuits, it is necessary to develop a time-step control method. In this research the focus is to develop a time-step control algorithm using MPDE formulation in one the dimensions of PDE.

1.2.1 Time Domain Methods

In this thesis two time domain methods are studied:

1. MFDTD : Multivariate Finite Difference Time Domain method

2. TD-ENV: Time domain ENVelope transient method

Some time domain solutions are often computationally slow since the results are determined by small time step increments. Therefore, an adaptive time step control for MPDE is proposed. TD-ENV with adaptive time step gives significant speed advantage over TD-ENV with fixed time step. For example, the MPDE method uses a bi-dimensional representation to represent fast and slow axis separately. When time step control is applied in the slow axis the simulation speed is increased.

1.3 Thesis Outline

This chapter introduced circuit simulators and simulation techniques. The MPDE and the importance of time stepping were introduced. The following chapters are organized as follows:

Chapter 2 gives the literature review and gives details of basic integration, interpolation, state variable representation, MPDE, MFDTD and envelope following methods. Chapter 3 explains stiff circuits and an adaptive time step control algorithm. Chapter 4 presents case studies for MFDTD and TD-ENV methods and simulation results. Chapter 5 presents the conclusions of this research and recommendations for future work.

Chapter 2

Literature Review

The main focus of this chapter is to review the concepts and techniques that will be applied to the work described in later chapters. This chapter is organized as follows: Section 2.1 describes the interpolation methods. Section 2.2 briefly presents a review of time marching integration methods MPDE, MFDTD and TD-ENV. Section 2.3 explains the Newton-Raphson method. Section 2.4 describes the stiff differential equations. Section 2.5 describes the local truncation error. Section 2.6 gives the stability analysis of the system. Section 2.7 introduces basic circuit analysis techniques and their literature reviews. Section 2.8 describes the widely separated time scales to form MPDE. Section 2.9 explains the Multi-time simulation of nonlinear circuits. Section 2.10 describes the MPDE representations. Section 2.11 explains the Finite difference and the multi-finite difference time domain methods. Section 2.12 explains the MPDE envelope following method. Section 2.13 explains the time-step control in SPICE and SPECTRE. Section 2.14 presents the state variable approach to analyze nonlinear circuits. Section 2.15 is a brief summary of the chapter.

2.1 Interpolation Methods

Often a function is represented by discrete points such as (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) as shown in Figure 2.1. Finding the value of y for x within the interval of (x_0, y_0) to (x_n, y_n) is called

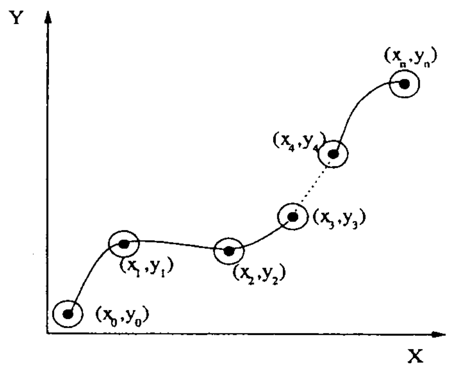


Figure 2.1: Discretized function representation

interpolation. Interpolation uses the data to approximate a function, which will fit all the data points. The data is used to approximate the values of the function inside the bounds of the data. The value of y can be found for any x if a continuous function $y = f(x)$ is given by $y_i = f(x_i)$, where $i = 1, ...n$. If x falls outside the range of x_n, it is no longer interpolation but instead it is called extrapolation.

2.1.1 Polynomial Interpolation

Polynomial interpolation is the most common choice of interpolants because they are easy to evaluate, differentiate, and integrate. From Figure 2.1, for given data (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) a polynomial of order n can be obtained as

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{2.1}$$

where $a_0, a_1, \cdots, a_n$ are real constants. Then Gaussian elimination can be used to set up $n + 1$ equations to find $n + 1$ constants. Depending on the order of x, this polynomial interpolation can be categorized as linear (1st order), quadratic (2nd order), cubic (3rd order) and so on.

2.2 Time Marching Numerical Integration

Numerical integration is performed if an analytical integration is infeasible and if a tabulated data is to be integrated rather than a known function. Time marching integration means, the integration is performed sequentially for each time point. The Euler method is used to perform this integration.

2.2.1 Euler method

The Euler method was invented by 18th century Swiss mathematician Leonhard Euler [34]. This method can be explained using a 1^{st} order ODE

$$x' = f(t,x), \qquad x(t_0) = x_0 \tag{2.2}$$

where x is unknown function and t is time. The numerical solution to a differential equation is an approximation to the actual solution. The solution x_n is a continuous function of a continuous variable t_n, where $n = 0, 1, 2, \cdots$ and $t_n < t_{n+1}$.

If equally spaced grid points with a time step h is considered as in Figure 2.2, the curve $x(t)$ is approximated as a straight line between the neighboring grid points t_n and t_{n+1}. Where

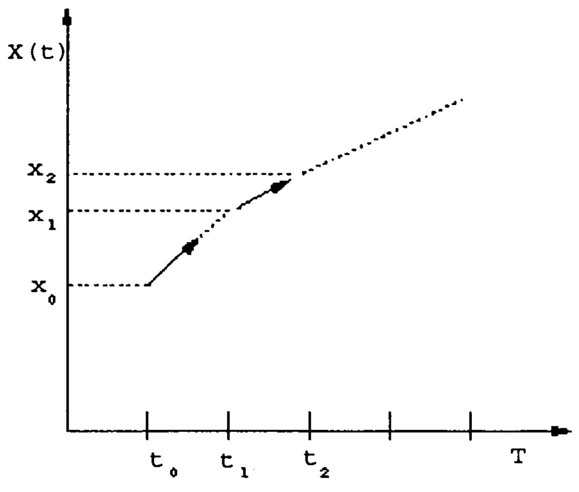


Figure 2.2: Illustration of Euler's method.

$$\begin{aligned} t_{n+1} &= t_n + n\,h \\ x_{n+1} &= x_n + x'_n\,h. \end{aligned} \tag{2.3}$$

Equation (2.3) is the essence of Euler's method and x_n can be found, given the initial values x_0 and t_0.

The integration of differential equations can be performed using explicit integration or implicit integration. The main difference between explicit predictors and implicit corrector is the use of current time step t_n in interpolation. Explicit integration is not very stable compared to implicit integration. Implicit methods also have an advantage of being faster to process due to larger time steps, especially in stiff circuits.

Explicit Forward Euler:

The solution x_n is approximated by assuming that a tangent straight line with slope x'_{n-1} connects the point x_n from the point x_{n-1}. Forward Euler (FE) formulation can be written as, $x_n \approx x_{n-1} + x'_{n-1}h_{n-1}$. There h_{n-1} represents the step time during $n-1$ time point. The step size h_{n-1} can be either constant or variable. However Figure (2.2) shows a constant step size.

Implicit Backward Euler:

This is an implicit representation and x_n, x'_n are all unknown. We may assume some initial value for x_n and iterate to approximate the solution x_n and x_{n-1}. The Backward Euler (BE) formulation can be written as, $x_n \approx x_{n-1} + x'_n h_{n-1}$.

2.2.2 Trapezoidal Rule

The trapezoidal rule is a simple average of the Forward-Euler and Backward-Euler schemes. The current value can be evaluated using the previous point, previous and current differentials and the time step as shown in Figure (2.2):

$$x_{n+1} = x_n + h\frac{(x'_n + x'_{n+1})}{2} \tag{2.4}$$

2.3 Newton-Raphson method

Newton-Raphson method is used to find the roots of complicated functions. Consider the Taylor Series expansion of $f(x)$ from x_0 to x:

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2}(x - x_0)^2 f''(x_0) + \cdots . \tag{2.5}$$

Setting the quadratic and higher terms to zero and solving the linear approximation gives:

$$f(x + h) \simeq f(x) + f'(x)h$$

where $h = x_0 - x$. This linear function of h that approximates f near a given x. Therefore the nonlinear function f can be replaced by a linear function, whose zero is easily determined to be $h = -f(x)/f'(x)$, assuming that $f'(x) \neq 0$.

The iteration scheme for Newton's method from [45] is as follows:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \tag{2.6}$$

Geometrically, x_{i+1} can be interpreted as the value of x at which a line, passing through the point $(x_i, f(x_i))$ and tangent to the curve $f(x)$ at that point, crosses the y axis. Figure 2.3 provides a graphical interpretation for Newton-Raphson method.

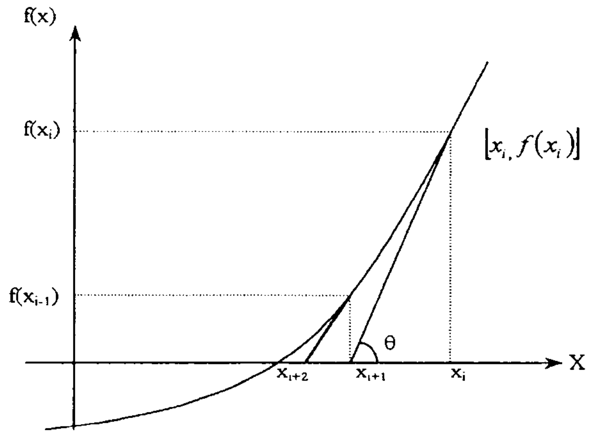


Figure 2.3: Graphical interpretation of Newton Raphson method.

From the above one-dimensional Newton's method, the iteration Equation (2.6) can be generalized for a vector input of n dimensions. If a vector input of n dimensions is used, a vector function $f(x_k)$, and $f'(x_k)$ will produce Jacobian matrix elements:

$$J_{f_{ij}(x)} = \frac{\partial f_i(x)}{\partial x_j}, \qquad J_{f_{ij}(x)} \in \Re \tag{2.7}$$

where i and j are row and column indexes respectively.

The derived Newton iteration equation for a vector function [23] is as follows:

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \mathbf{J^{-1}_{f(x_k)} f(x_k)} \tag{2.8}$$

Equation (2.8) is a vector form of Equation (2.6), where real quantities are replaced by vector using bold letters. The non-linear equation solver f solve in Octave uses the minpack hybrd routine, where a Quasi-Newton method is used. If a close initial guess is chosen for the solution, then the Newton method will converge fast. However depending on the initial guess x_0 of the coefficients, the scheme may not converge at all [23].

2.4 Stiff Differential Equations

Standard numerical techniques can give significant difficulties when applied to approximate the solution of a differential equation. When the exact solution contains terms of the form $e^{\lambda t}$ with λ is a complex number with negative real part, it will decay to zero with increasing time. On the other hand, generally the approximation for $e^{\lambda t}$ will not show the decaying property, unless a restriction is placed on the step size (h) of the method. This problem is particularly acute when the exact solution contains widely separated time scales on it, such as a steady-state term that does not grow significantly with t, together with a transient term that decays rapidly to zero.

A wide range of applications have rapidly decaying transient solutions that occur naturally, such as in DC-DC converter circuits, study of spring and damping systems, the analysis of control system and problems in chemical kinetics. These are examples of stiff system of differential equations. The following initial-value example problem [42] shows the stiff properties.

$$\begin{aligned} u_1' &= 9\,u_1 + 24\,u_2 + 5\ \cos\ t - \frac{1}{3}\ \sin\ t \\ u_2' &= -24\,u_1 + 51\,u_2 - 9\ \cos\ t + \frac{1}{3}\ \sin\ t \end{aligned}$$

with initial condition of $u_1(0) = \frac{4}{3}$ and $u_2(0) = \frac{2}{3}$ has the unique solution as follows:

$$\begin{aligned} u_1(t) &= 2\,e^{-3t} - e^{-39t} + \frac{1}{3}\ \cos\ t \\ u_2(t) &= -1\,e^{-3t} + 2\,e^{-39t} - \frac{1}{3}\ \cos\ t. \end{aligned}$$

In this solution, the transient term e^{-39t} causes this system to be stiff. Figure 2.4 shows the simulation results for U_1 and for U_2.

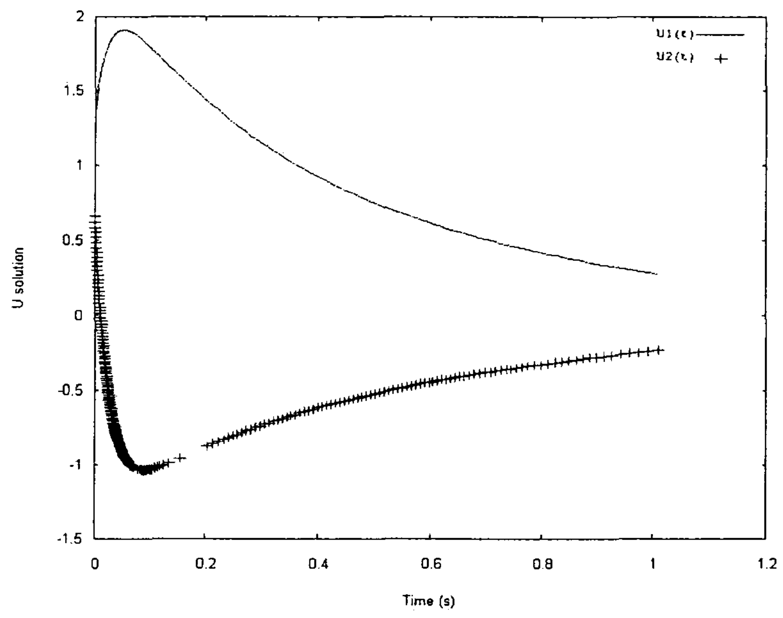


Figure 2.4: Stiff system simulation using adaptive time step

A 4th order Runge-Kutta method for time stepping gives disastrous results for $h = 0.1$, but gives accurate approximation for when $h = 0.05$ as evidenced in Table A.1 as referenced in [42].

2.4.1 Stiff systems

Time constants for stiff circuits differ by many orders of magnitude. A converter circuit is a good example of a stiff circuit. The time-step limitation is a computational problem when the circuit equations are stiff, that is when the ratio $\frac{|\lambda_{max}|}{|\lambda_{min}|}$ is several orders of magnitude, where λ_{max} is the eigenvalue of largest magnitude, and λ_{min} is the eigenvalue of smallest magnitude. The linear time invariant (LTI) system shown in Figure 2.5 is a stiff system and it can be considered to analyze the stiff properties. It contains small time constants that are due to parasitic

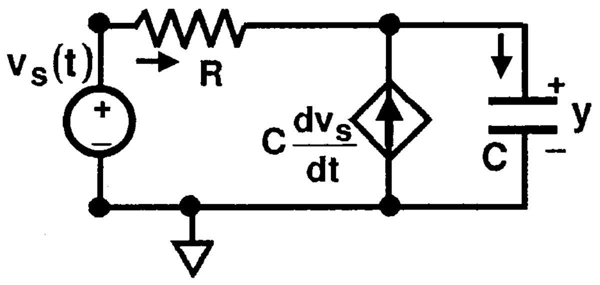


Figure 2.5: Circuit with widely separated $\lambda_{1,2}$.

components such as capacitor (C) and voltage controlled current source ($C\frac{dV_s}{dt}$), and large time constants that are due to coupling or bypass elements [43]. This circuit is a combination of voltage source-$V_s(t)$, resistor-R, current source-$C\frac{dV_s}{dt}$, and capacitor-C. A nodal analysis for the above circuit gives:

$$y'(t) = -\frac{1}{RC}[y(t) - V_s(t)] + \frac{dV_s}{dt} \tag{2.9}$$

where $V_s(t) = 1 - e^{-\lambda_2 t}$ and initial condition $y(0) = 2$. The solution $y(t)$ for the circuit becomes

$$y(t) = y_0 e^{-\lambda_1 t} + (1 - e^{-\lambda_2 t}). \tag{2.10}$$

In Equation (2.10), if $\lambda_1 = 10^6$ and $\lambda_2 = 1$, then the first part dies out in about 5 μs but the second part has a time constant of 1 sec.

$2e^{-\lambda_1 t}$ and $1 - e^{-\lambda_2 t}$ terms have a major contribution to the transient behavior and on the other hand $1 - e^{-\lambda_2 t}$ term has a major contribution to the steady state behavior, as time increases the transient behavior dies out and end-up with only the steady state behavior. The simulation step size for this circuit in Figure (2.5) depends on both $\lambda_{1,2}$, but we can not just take $|\ 1/\lambda_{maximum}\ |$ as the time step, due to longer computation time for smaller time step. The step size has to be adjusted to tradeoff between accuracy and computation speed. On the above

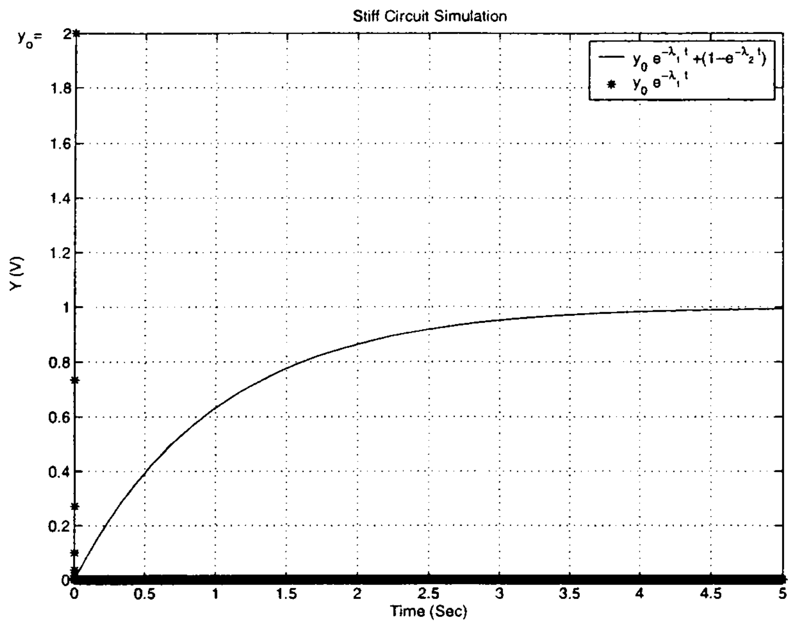


Figure 2.6: Solution for widely separated $\lambda_{1,2}$.

example circuit in Figure (2.5), if we use FE algorithm with variable time step method, such as 4 initial steps of $h = 10^{-6}$ sec and the rest with $h = 1$ sec, then the results are as shown in the Table 2.1. Variable time-step control on this example give efficient and faster computation.

Table 2.1: Stiff circuit simulated time and Output voltages Y(V)

Time (sec)	2 $e^{-\lambda_1 t}$	1-$e^{-\lambda_2 t}$	Y(V)
0	2	0	2
1×10^{-6}	0.73576	1×10^{-6}	0.735761
2×10^{-6}	0.27067	2×10^{-5}	0.270672
3×10^{-6}	0.099574	3×10^{-4}	0.202574
$3\times10^{-6}+1$	0	0.63212	0.63212
$3\times10^{-6}+2$	0	0.86466	0.86466
$3\times10^{-6}+3$	0	0.95021	0.95021
$3\times10^{-6}+4$	0	0.98168	0.98168
$3\times10^{-6}+5$	0	0.99326	0.99326

2.5 Local Truncation Error

The local truncation error (LTE) measures the error introduced in taking one time-step of any method assuming that all the values computed at previous time points are exact as shown in Figure 2.7. For a fixed-step method, the local truncation error only depends on the last step-size, but for a variable-step method, the error depends on previous step-sizes in a nonlinear way as well.

The algorithm uses a straight line interpolation to calculate the value of the state variable at the next time step. For example, a time series of t_{n-1}, t_n, t_{n+1} with state values of x_{n-1}, x_n, x_{n+1} and time derivatives of x'_{n-1}, x'_n, x'_{n+1} is considered. The LTE of any interpolation method is the difference between x_{n+1} and the exact solution at t_{n+1}, given that the past solutions $(x_{n-1}, x_n, ...)$ are exact. The LTE, can be measured either by the error in x or by the

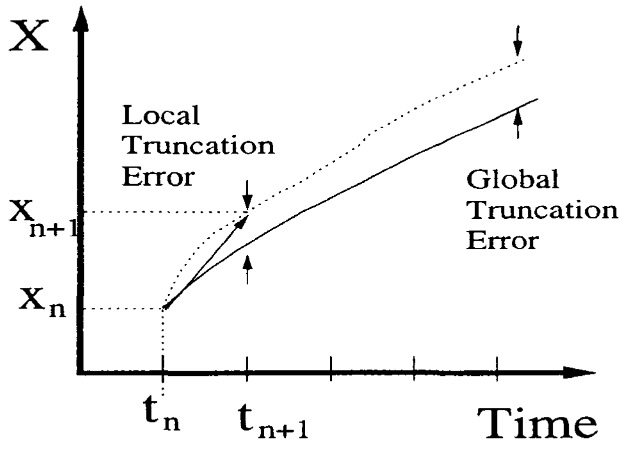


Figure 2.7: Truncation Error

error in x'. From BE algorithm:

$$\begin{aligned} x_{n+1} &= x_n + h_n x'_n + \frac{h_n^2}{2}\frac{d^2x}{dt^2}(\xi) \\ x'_{n+1} &= x'_n + h_n \frac{d^2x}{dt^2}(\xi) \end{aligned} \tag{2.11}$$

where $x_{n+1} = x(t_{n+1})$, $x'_{n+1} = x'(t_{n+1})$, $x_n = x(t_n)$, $x'_n = x'(t_n)$, and $t_n \leq \xi \leq t_{n+1}$.

From Equation (2.11), the BE formula is obtained to be

$$x_{n+1} = x_n + h_n x'_{n+1} - \frac{h_n^2}{2}\frac{d^2x}{dt^2}(\xi) \tag{2.12}$$

where the LTE estimation is expressed as $\frac{h_n^2}{2}\frac{d^2x}{dt^2}(\xi)$ and $(x_{n+1} - x_n)/h_n - x'_{n+1} \cong 0$. The LTE ($\varepsilon_x$) of the BE algorithm is the 2nd derivative term in Equation (2.12). In a circuit, ε_x is due to elements such as capacitors and inductors with units of charge and units of flux respectively.

A stable integration algorithm will obey the following inequality

$$\mid E_x(t_{n+1}) \mid \leq \Sigma_{i=1}^{n+1} \mid \varepsilon_x(t_i) \mid \tag{2.13}$$

where $E_x(t_{n+1})$ is the total error at time point t_{n+1}. If E_T is the total absolute error within T time, the error at each time point can be written as

$$\mid \varepsilon_x(t_{n+1}) \mid \leq \frac{h_n}{T} E_T \tag{2.14}$$

Using Equations (2.12) and (2.14) yield the time step constant h_n as

$$\frac{h_n^2}{2}\frac{d^2x}{dt^2}(\xi) \leq \frac{h_n}{T}E_T \tag{2.15}$$

$$h_n \leq \frac{2E_T}{T \mid \frac{d^2x}{dt^2}(\xi) \mid} \tag{2.16}$$

If $\varepsilon_{x'}$ is the LTE estimation in terms of x'_{n+1}, and the absolute value of error (E_D) is allowed per time point, then the LTE constraint will satisfy such that

$$\mid \varepsilon_{x'} \mid \leq E_D. \tag{2.17}$$

Using BE integration Equation () for $\mid \varepsilon_{x'} \mid$ yields the time step constraint to be

$$h_n \leq \frac{2E_D}{\mid \frac{d^2x}{dt^2}(\xi) \mid} \tag{2.18}$$

The estimated ε_x and h_n from Equation (2.16), and the estimated $\varepsilon_{x'}$ and h_n from Equation (2.18) are equivalent if $E_D = \frac{E_T}{T}$. For implicit polynomial methods, ε_x and $\varepsilon_{x'}$ are related by $\varepsilon_{x'} = \beta_0 \varepsilon_x$. For most polynomial methods, $\beta_0 < h_n$. Therefore, Equation (2.18) requires a smaller time step than Equation (2.16).

The relative tolerance to Equation (2.16) is more relevant in terms of $\varepsilon_{x'}$. Adding relative tolerance (ε_r) and absolute tolerance (ε_a) to Equation (2.16) gives

$$h_n \leq \frac{2[\varepsilon_r \mid x'_{n+1} \mid + \varepsilon_a]}{\mid \frac{d^2x}{dt^2}(\xi) \mid} \tag{2.19}$$

The exact value of ξ is required to find the differential term in Equation 2.12. Because the exact value of ξ is unknown and only the range of ξ which is $t_n \leq \xi \leq t_{n+1}$ is known, it is difficult to find LTE. To eliminate this problem the proposed method in [47] is used. From the Equation (2.11), it can be concluded that the difference between nonlinear solution (x_{n+1}) and the extrapolated solution ($x_n + h_n\, x'_n$) is proportional to the LTE of the system. Which can be written as, LTE = Extrapolated value - Nonlinear solution. Stability is guaranteed for adaptive step control algorithm, when the BE method is used. Stability will be shown in Section 2.6.

2.6 Stability

The LTE of any integration method is the difference between x_{n+1} and the exact solution at t_{n+1}, given that the past solutions $(x_n, x_{n-1}, \ldots)$ are exact. If x_n is exact, then the error of x_{n+1} is due to the LTE in computing the solution at t_{n+1}. However, if x_{n+1} is used to determine x_{n+2}, the error of x_{n+2} will depend on the LTE that occurred both at t_{n+2} and t_{n+1}. Therefore it can be concluded that x_n error depends on all of the time points t_1, t_2, ..., t_n.

An integration method is said to be stable, if the contribution of the LTE at time point t_k to the total error at t_n, where $n < k$, decreases as n increases. On the other hand a method is unstable, if the contribution of the LTE at a time point t_k increases without bound. The choice of an integration method usually involves a tradeoff between LTE and stability. Dahlquist [43] has defined an integration algorithm to be *A-Stable* if it results in a stable difference equation approximation to a stable differential equation. A numerical integration method is A-Stable for the following test Equation (2.20) if all numerical approximations tend to zero, as $n \rightarrow \infty$ in the time axis.

$$x'(t) = \lambda\, x \tag{2.20}$$

for a fixed positive time step (h) and an eigenvalue (λ) in the left-half plane.

2.6.1 Stability Analysis for numerical integrations

Stability is a global property related to the growth or decay of errors introduced at each time point and propagated to successive time points. Nonlinear circuit simulation is an iterative process, where the stability and convergence properties of numerical integration have to be considered during time domain simulation. Generally large time steps result in instability but small time steps can result in excessive computation and large errors due to numerical rounding error. Some integration methods are stable regardless of the time step that is used, whereas other methods are stable only for a certain range of time step values. Since the general stability analysis problem is difficult, in this thesis the stability of different methods will be compared for the test differential Equation (2.20). The exact solution for this equation is

$$x(t) = x_0 \, e^{\lambda x} \tag{2.21}$$

where x_0 is the initial condition. The FE and BE methods contain equal magnitude of LTE term. However, these two algorithms have different stability properties when applied to Equation (2.20).

- **FE Formula:**

  Applying FE method to Equation (2.20) without LTE gives

$$\frac{x_{n+1} - x_n}{h_n} = \lambda \, x_n. \tag{2.22}$$

  The iterations steps are as follows:

$$\begin{aligned}
\text{step 1:} \quad x_1 &= (1+\lambda h)x_0 \\
\text{step 2:} \quad x_2 &= (1+\lambda h)x_1 = (1+\lambda h)^2 x_0 \\
\vdots \quad & \quad \vdots \\
x_n &= (1+\lambda h)^n x_0
\end{aligned}$$

  For x_n to be stable after an infinite number of time steps, the following condition must be satisfied:

$$\mid (1+\lambda h) \mid \leq 1 \tag{2.23}$$

  If the eigenvalue (λ) is real and positive, Equation (2.21) is unstable in the sense that x increases without limit as t approaches infinity. Figure 2.8 shows the stable condition on complex plane. If λh is within this circle the integration scheme will be stable. For the case of a negative real λ, the FE solution is stable only if the time step is in the range of $-2 \leq Re(h\lambda) \leq 0$.

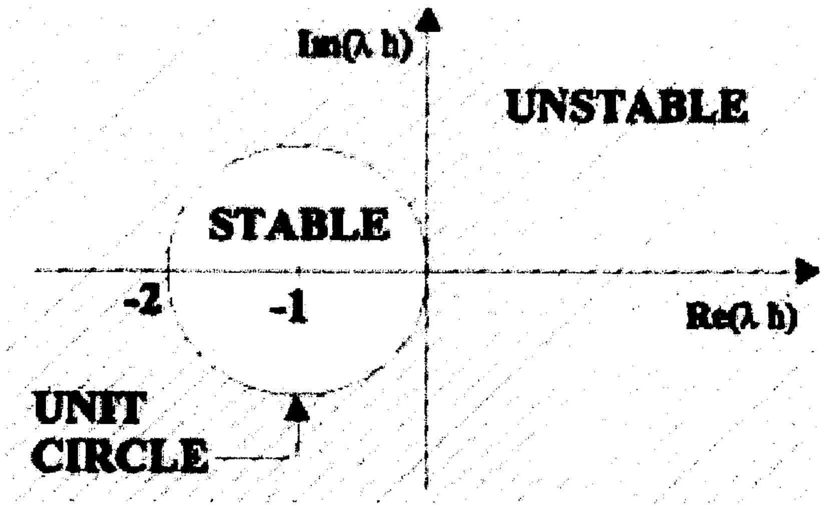


Figure 2.8: Region of stability for FE algorithm

- **BE Formula:**

  Applying BE method to Equation (2.20) gives the following iteration steps:

$$\begin{aligned}
\text{step 1:}\quad x_1 &= \tfrac{x_0}{(1-\lambda h)} \\
\text{step 2:}\quad x_2 &= \tfrac{x_1}{(1-\lambda h)} = \tfrac{x_0}{(1-\lambda h)^2} \\
\vdots \quad & \quad \vdots \\
x_n &= \tfrac{x_0}{(1-\lambda h)^n}
\end{aligned}$$

For x_n to be stable after an infinite number of time steps, the following condition

$$\left| \frac{1}{(1-\lambda h)} \right| \geq 1 \tag{2.24}$$

must be satisfied, BE solution will have better stability properties since Equation (2.24) is stable for any time step for $\lambda \leq 0$. Figure 2.9 shows the stable condition for BE method on complex plane. For FE method, the stability region in complex plane is shown inside the left unit circle, but in BE method the stability region is outside the right unit circle, which is a much larger region than that of FE.

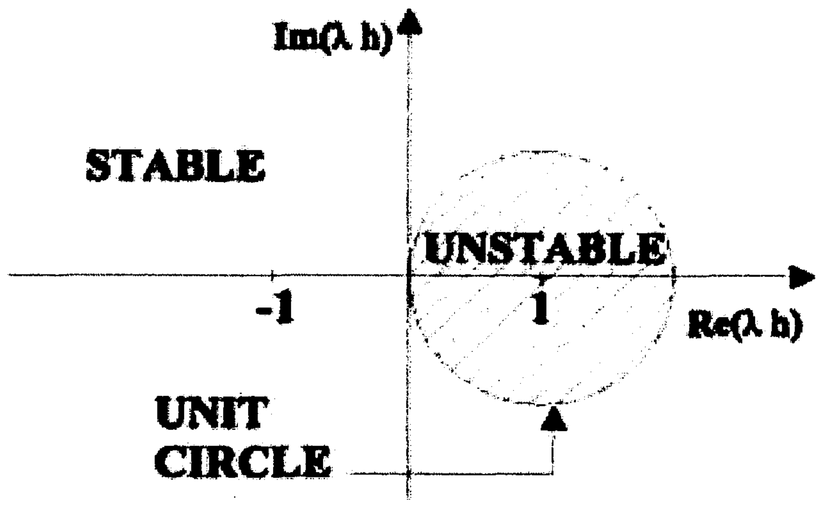


Figure 2.9: Region of stability for BE algorithm $\lambda_{1,2}$.

- **Trapezoidal Rule:**

Consider $y = f(x)$ over $[x_0, x_1]$, where $x_1 = X_0 + h$. The trapezoidal rule is To determine stability and accuracy for this method, it is applied to the linear ODE

$$f(x)' = \lambda f(x) \tag{2.25}$$

and iterating n times leads to

$$\begin{aligned} x_1 &= x_0 + \tfrac{\lambda h}{2}(x_0 + x_1) \\ &\vdots \\ x_n &= x_0 \left(\tfrac{2+\lambda h}{2-\lambda h}\right)^n \end{aligned} \tag{2.26}$$

To be numerically stable, h has to satisfy the following condition:

$$\left|\frac{2+\lambda h}{2-\lambda h}\right| < 1 \tag{2.27}$$

In Equation (2.26), $\left(\frac{2+\lambda h}{2-\lambda h}\right)^n$ approaches zero as n approaches infinity if the real part of (λh) is negative. Therefore the trapezoidal method is A-stable. This method is stable if the exact solution is stable and is unstable if the exact solution is unstable. The stability region for the trapezoidal rule is shown in Figure 2.10. Stable solution of a differential equation model will only be obtained if $Re(\lambda h) < 0$ condition is true.

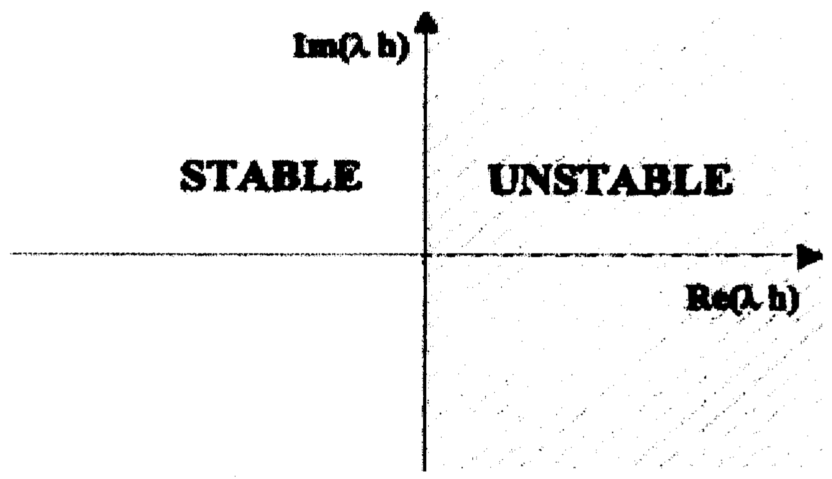


Figure 2.10: Region of stability for trapezoidal rule integration.

2.7 Circuit Analysis Techniques

Due to technical development, more and more complicated circuits have been introduced and the number of nonlinear components has increased, which emphasizes the importance of mathematics in circuit analysis techniques. Most simulators implement time-domain transient analysis, DC analysis and small signal AC analysis. The steady-state behavior of an analog circuit is typically of primary interest to a designer. Examples of quantities that are best measured when a circuit is in steady state include frequency, distortion, power, noise, and transfer characteristics such as gain and impedance. Many of these can only be measured accurately when the circuit is in the steady state.

2.7.1 Transient Analysis (TA)

TA computes the response of a circuit as function of time and solves the operation of a circuit in the time domain. The solution gives the node voltages and branch currents in the circuit with all transients [9]. The steady-state solution of a differential equation will asymptotically be reached when the transient phenomena die out. In fact, the inability to directly capture the steady-state response of systems is the most notable shortcoming of conventional TA.

There is no known method to directly solve the circuit equations during TA. The best one can hope for is to solve for a finite-difference approximation to the actual solution, such as finite sequence of points. Time is discretized and the solution is computed piecewise. The time-derivative operator is replaced with a finite difference approximation to compute the simulation. A set of Differential Algebraic Equations (DAE) can be formed from the ODE using the BE rule given by

$$\frac{dq(t_i)}{dt} \approx \frac{q(t_i) - q(t_{i-1})}{t_i - t_{i-1}} \tag{2.28}$$

Where $q(t)$ is a variable depending of time and the simulation interval is broken into small individual time steps. The approximation in Equation (2.28) is made in order to evaluate the time domain derivative and the differential equation is solved over the span of one time step at a time. Another important aspect of transient analysis is that there is history in the calculations, *i.e.* the solution at every time point is built from the solution at the previous time point. As a result, an error made at one time point can degrade the accuracy of future time points. Error accumulates or dissipates depending on the type of circuit being simulated. Stiff circuits are very sensitive to error build-up in transient analysis. Stiff circuits will be discussed in detail in Chapter 3.

2.7.2 DC Analysis

DC Analysis is used to find equilibrium points, *i.e.*, operating points that do not change with time. To find the solution of a nonlinear system of equations, simulators formulate and solve a sequence of linear systems of equations using Newton's method. This is discussed Chapter 3.

2.7.3 AC Analysis

In AC analysis the circuit is driven with 'small' sinusoidal signals and the steady-state solution is calculated. AC analysis are a family of frequency domain analysis which includes transfer function analysis, scattering parameter (SP), time domain reflectrometry (TDR) analysis, and noise analysis. All of the above analysis are based on the phasor mathematical technique [46].

2.7.4 Harmonic Balance (HB)

The HB technique is used to find the steady-state solution in the frequency domain [42]. HB is good to analyze RF amplifiers, mixers, and oscillators. Harmonic balance was given its name because it was viewed as a method for balancing currents between the linear and nonlinear sub circuits. Furthermore, harmonic balance is usually considered a mixed-domain method, because the nonlinear devices are evaluated in the time domain while the linear devices are evaluated in the frequency domain. However, evaluating the nonlinear devices in the time domain is not a fundamental part of the algorithm.

Currently, the standard method for the periodic steady-state analysis of non-linear RF and microwave circuits is the HB technique. It solves the Fourier coefficients and yields the steady-state response of a circuit as

$$v_r(t) \approx \sum_{k=0}^{n} V_k e^{j\omega_k t} \tag{2.29}$$

where $v_r(t)$ is the steady-state response at r^{th} node, k is the harmonic number, V_k is the voltage at k^{th} harmonic and ω_k is the k^{th} natural frequency.

HB is an accurate and efficient method when the solution can be represented with relatively few periodic steady-state sinusoids. This technique cannot be applied accurately and efficiently to analyze mixers, nonlinear amplifiers, samplers, etc., because they contain signals that are far from sinusoidal.

Krylov-subspace methods are techniques available to reduce the memory requirements and increase the speed of HB solution [11]. This option is useful in designing large RF integrated circuits or RF/IF subsystems, where a large number of devices or large numbers of harmonics and inter modulation products are involved.

Waveforms of a nonlinear circuit can be represented by Fourier coefficients. When these waveforms have sharp transitions, more Fourier coefficients must be introduced to get accurate representations.

2.8 Methods for circuits with widely separated time scales

Widely separated time scales cause difficulties to existing time-domain methods, because the time step has to be chosen according to a high frequency signal which cause a long computation time. When a circuit is driven by a source with widely separated time scales or with more than one frequency variation, the multi dimensional time representation can be used to simulate the circuit. To simulate the steady-state solution, it takes long simulation time and integration over an excessive number of periods. In stiff circuits, the transients take thousands of periods to die out. This makes the simulation consume a very long CPU-time. Very large number of integration steps means also a loss of accuracy. Therefore TA is expensive when it is necessary to resolve low modulation frequencies in the presence of a high carrier frequency. There are two major conventional classifications to simulate these circuits with widely separated time scales:

- Envelope methods
- Steady-state methods: There are two main techniques used in steady state analysis, one is Multi tone Harmonic Balance (Frequency domain)[33] and the other is Multi-Finite Difference Time Domain methods (MFDTD). MFDTD are discussed in detail in Section 2.11.

2.8.1 Envelope methods

Envelope method efficiently handles transient and steady-state analysis of microwave circuits for arbitrary modulated carrier excitation, without excessive computation overhead [12]. This method can be divided into two parts:

(1) Sample-envelope methods

These methods operate in time-domain [5], and they are based on shooting methods. The envelope is represented by a slowly varying sampled version of the waveform. Sample-envelope methods can be classified as Envelope following method and Quasi-periodic shooting method[15].

Envelope following method: This method uses a time-domain integration method in order to predict the sample envelope evolution without sweeping the fast carrier cycles. The envelope is approximated with a piecewise polynomial [47].

Basic Idea For Envelope Following Method For traditional envelope following method in one dimensional grid with one time variable, assume the following nonlinear circuit equation described by

$$\dot{y}(t) - f(y(t), t) = 0 \tag{2.30}$$

where $y(t)$, $\dot{y}(t) \in R^N$ with N state variables.If the initial state variables are known, $y(t)$ can be found in subsequent time instants by integrating Equation (2.30).

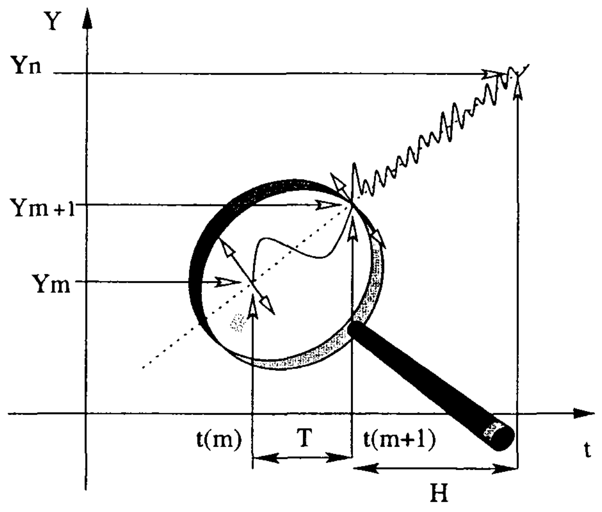


Figure 2.11: Envelope Following Method

In switched circuits given in Section 4.1, at least one of the input functions is periodic and it is assumed the period is T. If an accurate solution has been computed with conventional integration in the m^{th} clock cycle $[mT,\ (m+1)T]$, the initial state at the beginning of the subsequent n^{th} cycle $[nT,\ (n+1)T]$ can be explicitly predicted through linear extrapolation:

$$y_n = y_{m+1} + \frac{H}{T}(y_{m+1} - y_m) \tag{2.31}$$

where $H = (n - m - 1)T$ is jump length , T is period of a following envelope excitation as shown in Figure 2.11. For simplicity the following terminologies are used: $y_n = y(t_n)$, $y_{n+1} = y(t_{n+1})$, $y_m = y(t_m)$, $y_{m+1} = y(t_{m+1})$, and H is jump length of envelope integration.

Generally the envelope following method is based on a time domain shooting scheme, in which the nonlinearities are resolved by the time domain integration rather than explicitly being expressed as harmonics of fundamental frequencies.

- **Shooting Method:**

  Shooting method initially guesses the past differential and shoots to get the present value. Until it gets the proper present value, it will iteratively have various past differential guesses and shoot for the present value. To solve a second order equation of the form $\ddot{y} = f(t, y, \dot{y})$ subject to $y(0) = c_0$ and $y(1) = c_1$ using shooting method, the boundary condition $y(0) = c_0$ is applied and an initial guess is made to be $\dot{y}(0) = a_0$. Through calculations the initial conditions are achieved back to prove the guess is correct. The calculation proceeds until a value for $y(1)$ is achieved. Within some acceptable tolerance, the guess is revised for $\dot{y}(0)$ to some value a_n, and the time integration is repeated to obtain a new value for $y(1)$. This iteration is continued until $y(1) = c_1$ is within acceptable tolerance.

  Achieving an acceptable tolerance solution depends on the choice of a. A good refinement algorithm for a will give a small number of iterations. For such cases a root finding method such as Newton-Raphson method can be chosen. For higher order-n systems with m-boundary conditions at $t = t_0$ and $(n-m)$ boundary conditions at $t = t_1$, it will require guesses for $(n - m)$ initial conditions. The computational cost of refining these $(n-m)$ guesses will rapidly become large as the dimensions of the matrix increase, such as Jacobian matrix in Newton method. Sparse matrix techniques can be used to improve the computation time. The following references from Brambilla [27], and Furse [32] propose improving the shooting method through Envelope Following Method.

(2) Fourier-envelope methods

These methods operate in frequency domain [5] and they are transient envelope methods based on HB. Fourier-envelope methods can be classified as

- **Circuit envelope method:** This technique was developed to simulate modern wireless circuits with complex digitally modulated RF signals [?], [35]. Circuit Envelope (CE) method can be considered as a type of time-varying Harmonic Balance (HB) simulation. This technique not used for steady state analysis.
- **Transient envelope method:** This method uses time domain method, such as FDTD, instead of HB during simulation. This technique handles the transient and the steady state analysis of a communication circuit [12],[4].

Modifications of HB are not usually called envelope methods, but it can be included in envelope methods as they do follow the shape of the signal instead of the signal itself. HB based envelope methods are inefficient for strong nonlinear circuits, as they are based on Fourier coefficients.

2.9 Multivariate simulation

In conventional time domain methods, if there are widely separated frequencies in a circuit, the time step h is determined by the highest frequency of the circuit. A large number of periods of the high frequency components is required to reach the steady state response.

Quasi periodic signals are difficult to simulate due to the widely separated compound frequencies. Therefore they are non-periodic signals. Consider the signal from [2]:

$$B(t) = sin\left(\frac{2\pi t}{T_1}\right) sin\left(\frac{2\pi t}{T_2}\right) \tag{2.32}$$

If $\frac{T_1}{T_2}$ is a rational number, the signal is a periodic signal. If the ratio $\frac{T_1}{T_2}$ is irrational, the signal is a true quasi periodic signal. However, if $\frac{T_1}{T_2}$ is either very large or very small, the signal can be treated as the 2-tone quasi periodic signal.

Figure 2.12 shows a single time variable plot for a 2-tone quasi periodic signal. On this figure, the ratio $\frac{T_1}{T_2} = 100$. If $n = 20$ samples are chosen in a time period of T_2, there will be 2000 samples in the completed period of T_1 signal.

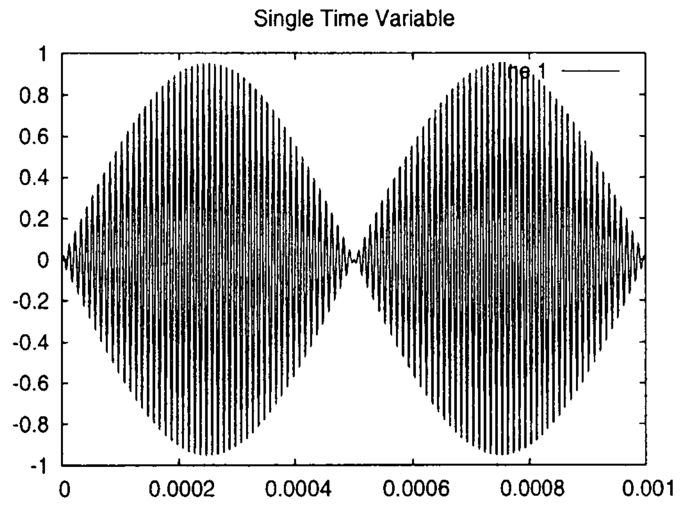


Figure 2.12: Single time variable 2-tone quasi periodic signal

For a multi-variable 2-tone quasi periodic signal shown in Figure 2.13, two time variables t_1 and t_2 are introduced to Equation (2.32) and the bi-variate form is given such that

$$\hat{B}(t_1, t_2) = sin\left(\frac{2\pi t_1}{T_1}\right) sin\left(\frac{2\pi t_2}{T_2}\right). \tag{2.33}$$

On this figure, each period is sampled at n points, and results in a $n \times n$ grid. If a 20×20 grid is used to represent Equation (2.33), a total of 400 points will be required, which is less than 2000 points needed in single variable time step method or traditional method. The advantage of multi-variate method is that it takes less storage and computation time to represent the signal. The multi-variate method contains all the information needed to recover the original signal completely.

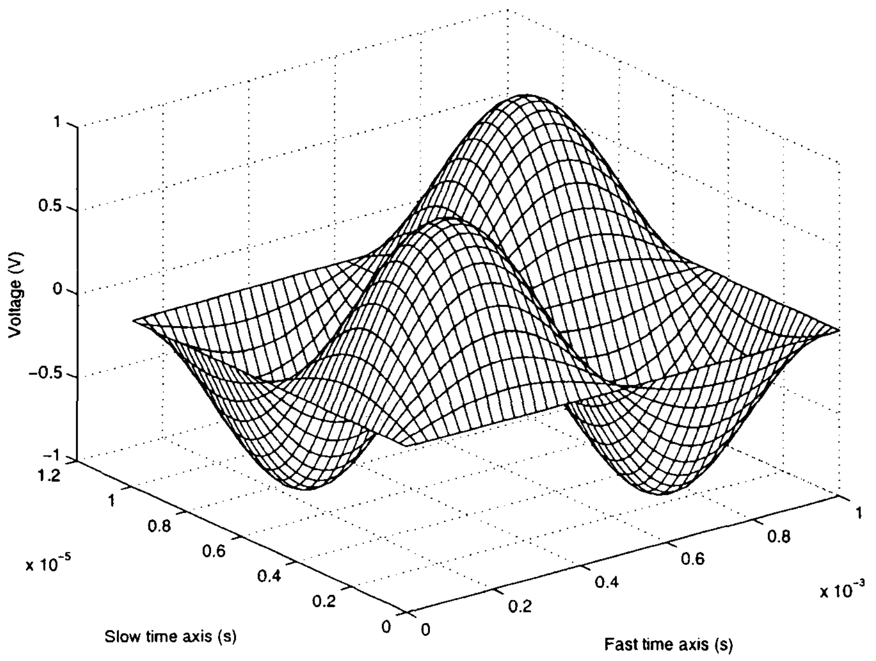


Figure 2.13: Multi-time variable 2-tone quasi periodic signal

2.10 ODE to MPDE representation

PDE with multi time variables are called MPDE. The traditional ODE form of a circuit is given by the Differential Algebraic Equation (DAE)

$$\dot{q}(x) = f(x) + b(t). \tag{2.34}$$

Where q is the charge, f is the resistive terms [2, 13], b is the excitation produced by the independent voltages and current sources, and $x(t)$ is the vector of the unknown voltages and currents of the circuit.

Circuits with widely separated time scales are more difficult to simulate because they have fast varying signal, and long simulation time due to the low frequency signal. Nevertheless, when the circuit is characterized by multi-rate behavior, its variables can often be represented efficiently using multiple time variables. By denoting the new multivariate forms to $x(t)$ and $b(t)$ in Equation (2.34) by $\hat{x}(t_1, ..., t_m)$ and $\hat{b}(t_1, ..., t_m)$ respectively, the DAE (2.34) can be transformed into

$$\frac{\partial q(\hat{x}(t_1, ..., t_m))}{\partial t_1} + .. + \frac{\partial q(\hat{x}(t_1, ..., t_m))}{\partial t_m} = f(\hat{x}(t_1, ..., t_m)) + \hat{b}(t_1 .... t_m) \tag{2.35}$$

An extended discussion on differential equation solution can be found in the Appendix A.3. The existence and uniqueness of an MPDE solution can not be guaranteed, because the DAE does not necessarily have a solution. However, it has been proved [2] that a periodic solution of the MPDE generates a quasi periodic one for the DAE, and if the original problem has a quasi periodic solution, then the MPDE also has a corresponding solution. Note, that if the value of $\hat{b}(t_1, ..., t_m)$ is known, the value of $b(t)$ is easy to calculate at any time moment be substituting $t_1 = t_2 = ... = t_m = t$. In this thesis, a widely separated time scales system is simulated using bi-dimensional MPDE, and transformed into traditional one time step solution. This is presented in Chapter 4 in detail.

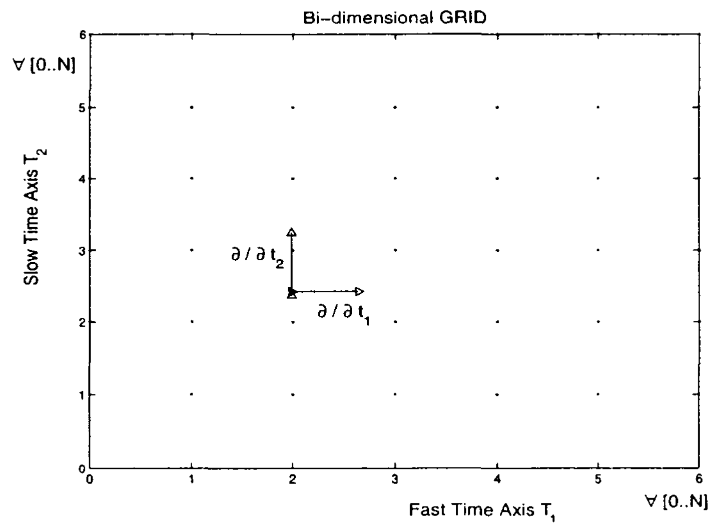


Figure 2.14: Grid representation of MPDE

Figure 2.14 shows the grid used for the numerical solution of the MPDE. For simplicity Figure 2.14 shows a bi-dimensional representation, where $\frac{\partial}{\partial t_1}$ is the fast axis differential and $\frac{\partial}{\partial t_2}$

is the slow axis differential form.

Theorem 1 (MPDE–DAE Relation): If $\hat{x}(t_1,...,t_m)$ and $\hat{b}(t_1,...,t_m)$ satisfy the MPDE in Equation (2.35), then $x(t) = \hat{x}(t+c_1, t+c_2, ..., t+c_m)$ and $b(t) = \hat{b}(t+c_1, t+c_2, ..., t+c_m)$ satisfy the circuit's DAE in Equation (2.34), for any fixed $c_1,...,c_m$.

The following proof of Theorem 1 [2] gives the relationship between MPDE and DAE. *Proof*: From Equation (2.34) and Equation (2.35), $q(x(t))$ can be written in the form of widely separated time scales in DAE such that

$$q(x(t)) = q(\hat{x}(t+c_1, t+c_2, ..., t+c_n)) \tag{2.36}$$

where $c_1 = t_1 - t$, $c_c = t_2 - t$ ...... $c_n = t_n - t$. A bivariate function is assumed to be

$$x(t) = \hat{x}(t_1(t), t_2(t)) \tag{2.37}$$

and the ODE $\frac{\partial x}{\partial t}$ becomes

$$\frac{\partial x}{\partial t} = \frac{\partial \hat{x}}{\partial t_1}\frac{\partial t_1}{\partial t} + \frac{\partial \hat{x}}{\partial t_2}\frac{\partial t_2}{\partial t}. \tag{2.38}$$

Where $\frac{\partial t_1}{\partial t}$ and $\frac{\partial t_2}{\partial t}$ are equal to 1, as $t_n = t + c_n$. Which implies, $\frac{\partial t_n}{\partial t_n} = \frac{\partial t}{\partial t_n} + \frac{\partial c_n}{\partial t_n} \rightarrow \frac{\partial t}{\partial t_n}$. Substituting $\frac{\partial t_n}{\partial t_n}$ with $\frac{\partial t}{\partial t_n}$ to get an MPDE form:

$$\frac{\partial x}{\partial t} = \frac{\partial \hat{x}}{\partial t_1} + \frac{\partial \hat{x}}{\partial t_2} \tag{2.39}$$

$$\begin{aligned}
\frac{\partial q(x(t))}{\partial t} &= \frac{\partial q(\hat{x}(t+c_1,\cdots,t+c_m))}{\partial t_1} + \cdots + \frac{\partial q(\hat{x}(t+c_1,\cdots,t+c_m))}{\partial t_m} \\
&= f(\hat{x}(t+c_1,\cdots,t+c_m)) + \hat{b}(t+c_1,\cdots,t+c_m) \quad \text{from (2.35)} \\
&= f(x(t)) + b(t)
\end{aligned}$$

end of Proof.

2.11 Finite Difference and Multi-Finite Difference Time Domain (FDTD and MFDTD)

The FDTD method is used in time domain circuit simulation. The algorithm begins by discretizing differential equations using previous time points, resulting in a set of explicit finite difference equations. These finite difference equations are solved using root finding algorithms, such as Newton-Raphson method. Detailed explanation of FDTD method is found in [38]. Efficient approaches have been proposed by researchers [32, 31] to obtain frequency responses using FDTD. A bi-dimensional simulation method can be categorized into three types. The main difference in these types are periodicity of the excitation signals. First type is non-periodic in both directions, second type is periodic in both directions and the third type is periodic in one and non-periodic in other direction. If non-periodic signals are considered for both directions of a circuit, then traditional method(one time dimension) will be used. If periodic excitation is considered for both directions, the circuit is analyzed using MFDTD method. The n-dimensional grid is created by approximating the differential operators with a numerical differentiation formula. That is replacing the original differential equation by a finite difference approximation at each of the grid points. If there are periodic and non-periodic excitations in the circuit, it is analyzed using Time Domain Envelope following method (TD-ENV). There, the adaptive time step control is used in one time direction [2].

One of the most important open problems in circuit simulation is the prediction of strong nonlinear regimes when the input signal is composed of widely separated time constants. Multi-tone time domain is a mature technique. The previous section discussed several methods developed to analyze signals with more than one tone. Most of these methods can be formulated with a multi-time partial differential equation (MPDE) and be processed as this equation in different ways, using time domain and frequency domain. MPDE formulation was published by Roychowdhury in [2] and [13], where the derivation and assumptions of the MPDE formulation are given in detail, and the numerical methods presented are compared.

The FDTD method has rapidly become an attractive choice due to its robustness, programming simplicity and flexibility in the analysis of a wide range of structures. However, this technique has the drawback of high memory resources and computational power, especially when dealing with large grid size N as shown in Figure 2.14.

If a 2-dimensional circuit is considered, Equation (2.35) becomes

$$\frac{\partial q(\hat{x}(t_1,t_2))}{\partial t_1}+\frac{\partial q(\hat{x}(t_1,t_2))}{\partial t_2}=f(\hat{x}(t_1,t_2))+\hat{b}(t_1,t_2). \tag{2.40}$$

In this case, $\hat{x}(t_1,t_2)$ and $\hat{b}(t_1,t_2)$ are bi-periodic and the grids repeat over the whole (t_1,t_2) plane. The problem is solved using a 2-dimensional grid at every time point (t_1,t_2) found on the plane shown in Figure 2.14. Therefore, $\hat{x}((t_1+T_1),(t_2+T_2))=\hat{x}(t_1,t_2)$. If a grid size of $N\times N$ is considered with N^2 points $\{\bar{t}_{i,j}\}=(t_{1_i},t_{2_j})$. Here $t_{1_i}=(i-1)h_1$ and $t_{2_j}=(j-1)h_2$, $1\le i\le N$, $1\le j\le N$. The grid spacings in t_1 and t_2 directions are $h_1=\frac{T_1}{N}$ and $h_2=\frac{T_2}{N}$ respectively.

In MFDTD method, the n-dimensional grid is created by approximating the differential operators with a numerical differentiation formula. The original differential equation is replaced by a finite difference approximation at each of the grid points. Discretizing the differential operators in Equation (2.35) using BE method gives

$$\begin{aligned}\frac{\partial\hat{q}_{i,j}}{\partial t_1}&=\frac{\hat{q}_{i,j}-\hat{q}_{i-1,j}}{h_1}\\ \frac{\partial\hat{q}_{i,j}}{\partial t_2}&=\frac{\hat{q}_{i,j}-\hat{q}_{i,j-1}}{h_2}\end{aligned} \tag{2.41}$$

where $\hat{q}_{i,j}=q(\hat{x}(\bar{t}_{i,j}))$. This leads to a set of nonlinear algebraic equations, F(x). If Equation (2.35) is represented using the discretization (Equation (2.41)), this large system of equations can be written in matrix form

$$F_{i,j}\equiv\frac{\hat{q}_{i,j}-\hat{q}_{i-1,j}}{h_1}+\frac{\hat{q}_{i,j}-\hat{q}_{i,j-1}}{h_2}-\hat{f}_{i,j}-\hat{b}_{i,j}=0 \qquad i,j\ \in\ [1,...,N], \tag{2.42}$$

where $\hat{f}_{i,j}=f(\hat{x}(\bar{t}_{i,j}))$ and $\hat{b}_{i,j}=\hat{b}(\bar{t}_{i,j})$. Equation (2.42) contains more unknowns than number of equations due to discretized differential operators on the t_1=0 and t_2=0 lines respectively. The bi-periodic boundary conditions are used to eliminate additional unknowns. These boundary conditions provide relations between the unknowns at the boundaries, and they are seen as corner blocks in the Jacobian matrix. Then the problem can be solved numerically by means of the Newton-Raphson method.

For example if a $N \times N$ grid is used, then N^2 non-linear equations with N^2 unknowns are produced in the simulation. The size of the Jacobian Matrix in Equation (2.7) is $N^2 \times N^2$.

The MFDTD technique is well known to find the steady state solution of circuits. If the MFDTD technique is analyzed in m-dimensions instead of two, the Jacobian matrix will have $(N^m r)^2$ elements, where r and N are number of variables and grid size respectively.

2.12 MPDE Envelope Following Method

Envelope following analysis is a need for efficient simulation of stiff and highly oscillatory circuits. This method reduces the simulation time without compromising accuracy by exploiting the property that the behavior of the circuits in a given high frequency clock cycle is similar, but not identical, to the behavior in the preceding and following cycles. The 'envelope' of the high frequency clock can be followed by accurately computing the circuit behavior over occasional cycles, which accurately capture the fast transient behavior.

Proposition: To obtain faster simulation time and accurate results in Envelope Following Method (EFM), the high frequency component (small time period signal - small axis) should be greater than the highest natural frequency of the circuit. If this condition is not considered, the simulation will have some numerical problems as discussed in Section 2.4.

Time Domain Envelope following method (TD-ENV) uses a multi-tone (Fast and slow time axis) algorithm, which is an extension to a method introduced by Kundert, White and Sangiovanni-Vincentelli in [47] and [9]. A similar TD-ENV method for transient simulation is proposed by Brambilla and Maffezzoni in [6]. Details and application of TD-ENV method on transient simulation will be discussed in Chapter 3.

2.13 Time Step Control of SPICE and Spectre

SPICE has two time step control algorithms. One is based on LTE to choose the optimal time step. SPICE estimate the LTE made on every capacitor and inductor in the circuit, then chooses the time step small enough to assure that the largest LTE remains within absolute *tolerance*. The second control algorithm is based on number of iterations required for convergence at a step to decide how big to make the next time step. This second method is not commonly used due to lack of reliability.

In Spectre, to control the truncation error, the local truncation error has to be measured. It is the truncation error made on each step. The standard measure of LTE is the difference between the computed solution and the extrapolation from the previous few steps. It is assumed that the previous solutions are exact.

The following steps describe how simulators operate and control the time steps [36]:

- The user specifies a total time range, tolerances, and iteration limits.
- Compute DC analysis solution at zero time, with initial conditions.
- Some simulators have predefined breakpoint table to deal with nonlinear devices. Such is the case with the piecewise linear sources. The breakpoint table contains a sorted list of all the transition points of the independent sources. During the simulation, whenever the next time point is sufficiently close to one of the breakpoints, the time step is adjusted to land exactly on the breakpoint. This prevents unnecessary calculations.
- An internal time step control variable updates the current time, and the values of the independent sources are calculated at that time.
- Solve the system of equations through numerical integration and a finite number of root finding solver (Newton-Raphson) iterations. If the number of iterations exceeds the user defined maximum iterations per time point, then the time step is reduced by some factor. If this new time step is acceptable, then recalculate independent sources.
- Following convergence, the local truncation error is calculated, where the Trapezoidal integration method is used to estimate the error. LTE is the difference between the

computed solution and the extrapolation from the previous few steps.

- The error tolerance is compared with the value in the Local Truncation Error. If the error is within acceptable limits, the results are stored and the analysis continues at the next time point. Otherwise, the analysis is repeated with a smaller time step. This process continues to the end of total simulation time.

In SPICE the LTE is estimated at each time point and the time step is adjusted such that the LTE is maintained within reasonable bounds. Error check and Convergence check routines is incorporated and the matrices are re-calculated when the time step is changed. This is used in SPICE2 by using the time step formula

$$h_{n+1} = \sqrt{\frac{\varepsilon_r \mid x'_{n+1} \mid + \varepsilon_a}{DD_3(t_{n+1})}} \qquad (2.43)$$

where ε_r is the relative tolerance, ε_a is the absolute tolerance, and $DD_3(t_{n+1})$ is the third differential term. This time step derivation will be discussed in detail in Chapter 3.

2.14 State Variable Representation Of Diode Model

The state variable representation of the diode model improves the convergence of the system. The current equation for the diode is given by

$$i_d = I_s\left(e^{\frac{V_d}{V_T}} - 1\right) \qquad (2.44)$$

where i_d is the diode current, I_s is the reverse saturation current, V_d is the voltage applied to diode and V_T is thermal voltage.

In Equation (2.44) the diode current has an exponential dependence on voltage. This causes convergence problems when the voltage is updated during nonlinear iterations in circuit simulation as shown in Figure 2.15. At voltages greater than the threshold, small voltage increments can result in large current changes. To reduce the convergence problems a state variable x is introduced as proposed by Rizzoli [14]. This model for diode returns outputs of V_d and i_d for an input state variable x and is equivalent to Equation (2.44).

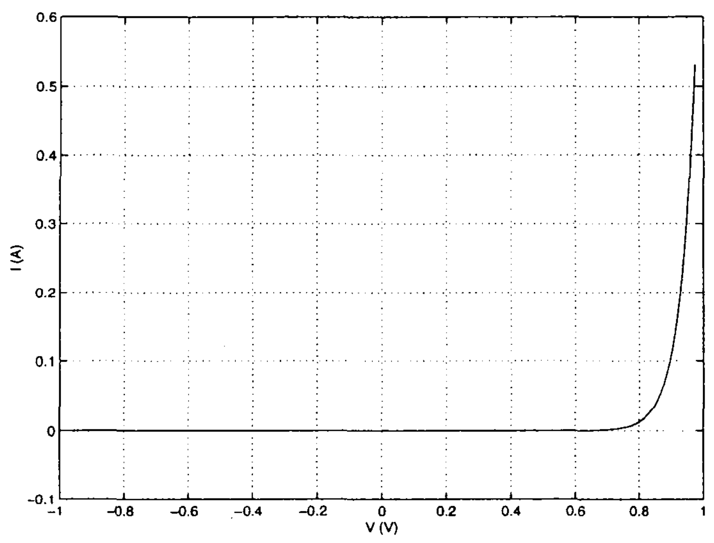


Figure 2.15: Relation between v and i in a diode.

The proposed state variable approach [14] in the HB simulation context provides great flexibility for the design of nonlinear device models. The state variables can be chosen to achieve robust numerical characteristics.

$$v(t) = \begin{cases} x(t) & \text{if } x(t) \leq v_1 \\ v_1 + \frac{1}{\alpha} I_s \left(1 + \alpha \left(x(t) - v_1\right)\right) & \text{if } x(t) > v_1 \end{cases} \tag{2.45}$$

$$i(t) = \begin{cases} I_s(e^{\alpha x(t)} - 1) & \text{if } x(t) \leq v_1 \\ I_s(e^{\alpha v_1}(1 + \alpha(x(t) - v1)) - I_s & \text{if } x(t) > v_1 \end{cases} \tag{2.46}$$

where the threshold voltage $v_1 = 0.65\ V$, $\alpha = \frac{1}{V_T}$ and $V_T = 0.026\ V$ at $T = 300\ K$. This diode model can be refined to include capacitive effects and the resistance of N and P regions. Figure 2.15 shows an exponential behavior after the threshold voltage v_1. The effect of the parameterised model is shown in Figures 2.16 and 2.17. These figures illustrate the improvement in the regular nonlinear behavior of the model for the state variable approach.

The possibility of large changes is eliminated through the use of parameterizations which ensures a smooth, well behaved current, voltage and error function variations when the state variable is updated. Thus the i - x and v - x functions are well behaved and thus circuit analysis via nonlinear iterations is also well behaved as shown in Figures 2.16 and 2.17.

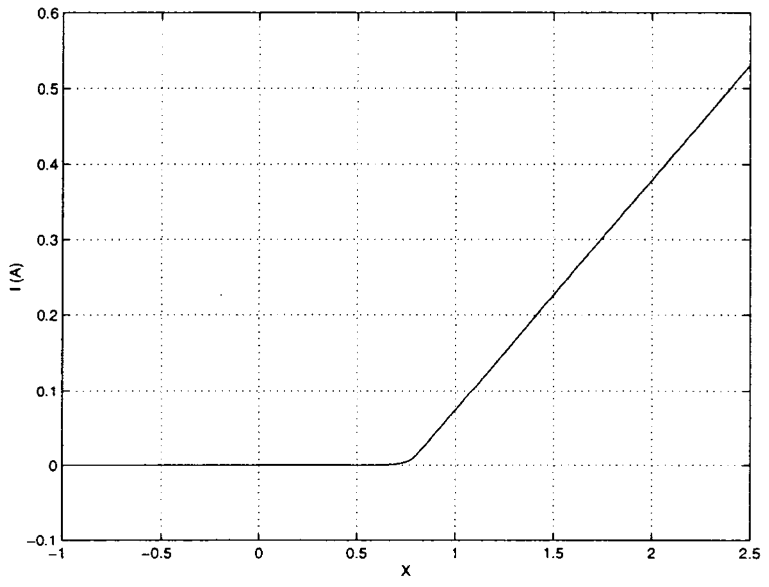


Figure 2.16: Relation between x and i in a diode.

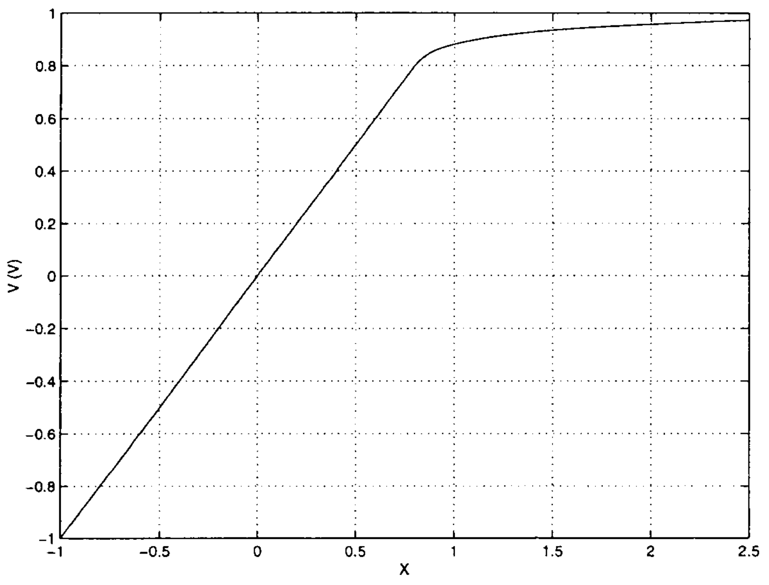


Figure 2.17: Relation between x and v in a diode.

2.15 Summary

This chapter described the concepts and techniques that will be applied in Chapter 3 and Chapter 4. Basic concepts such as interpolation methods, integration, Newton-Rapson method, LTE and stability. The details of TD-ENV method and its use for transient simulations, and the details of MFDTD method and its use for steady state simulations were explained. It was shown that the diode state variable representation improves the diode model numerical behaviour. Local Truncation Errors and stability of the numerical methods were discussed.

Chapter 3

Stiff Circuit Time Step Control

Generally widely separated time scale circuits are called stiff circuits. Any of the following can be widely separated in a stiff circuit: natural time constants, source time constants, and interval of interest. The main focus of this chapter is to develop an adaptive time step control for envelope transient analysis to improve simulation constrains such as memory and computation time. This chapter is organized as follows: Section 3.1 gives a brief introduction to stiff circuits and explains the importance of adaptive time step algorithm. Section 3.2 explains the envelope following algorithm. Section 3.3 describes the time step control algorithm for TD-ENV method. Section 3.4 is a brief summary of the chapter.

3.1 Introduction

In any transient simulation, the computation time will be proportional to the number of time steps used in the simulation plus the rejected time points. The rejected time points are due to the convergence and the acceptable LTE. A root finding algorithm, such as Newton-Raphson method, can choose a time step and try to find the root but if the root is not found then the algorithm need to reduce the step size and attempt to find the root again.

Typically, the simulation time is a multiple of the large time constant (smallest eigenvalue) associated with the linearized circuit. On the other hand, a limitation of the time step(h) depends on the smallest time constant (largest eigenvalue) of the linearized circuit. Due to the conflicts between small and large eigenvalues, a simulation will continue with small time step, which leads to a large computation time. Many literatures classified circuits with the above properties as stiff circuits [36].

When MPDE is used in this case study, an adaptive time step will be a suitable solution to overcome any sharp or smooth transition in real time (t_1) axis. MPDE will lead to have more than one time step variable in the system. For example if a two tone signal is chosen, there will be two different time axis such as fast and slow axis accordingly. Generally Runge-Kutta methods are not used in circuit simulation to solve their differential equations due to its stiff properties [42].

A dynamically changing time step increases the accuracy of simulation and reduces the simulation time by varying the value of the time step over the transient analysis sweep depending upon the rate of change of the output. An adaptive time step algorithm increases the time step value when internal nodal voltages are stable and decrease the time step value when nodal voltages are changing quickly. Another advantage of adaptive time step is that it eliminates the non-convergence problem during the transient analysis. If non-convergence occurs, the time step is reduced until the solution converges. If the number of iterations at a time point is less than the present value, then the time step is increased. However, the user has control over the maximum allowed time step, therefore the accuracy. The simulator chooses the time step to control the truncation error made at each step.

In the TD-ENV method, there is a fast axis with periodic excitation function and there is a slow axis with non-periodic excitation function. It has been reported in the literature [2, 6] that the differential equations in the real axis (slow axis t_1), which is the total simulation time, are stiff and at times present fast variations, as a result an adaptive time step is chosen. A new time step is adapted in each consequent time point in the slow axis, where adaptation is needed due to long simulation time. A future study can be done of an adaptive time step algorithm in both directions simultaneously.

3.2 Time Domain Envelope Following Algorithm (TD-ENV)

The traditional envelope following method is explained in the literature review Chapter 2. Time domain envelope transient (TD-ENV) is an envelope following method based on a bi- dimensional (or in general, multi-dimensional) representation of the time domain [2]. Circuits can be described by a system of differential equations of the form described in Equation (2.34). It has been proved in Section 2.11, if $\hat{x}(t_1, t_2)$ is the solution of Equation (2.40), then $x(t) = \hat{x}(t_1, t_2)$ is the solution of the system of Equation (2.34). For the TD-ENV problem, the boundary conditions are:

$$\hat{x}(t_1, t_2) = \hat{x}(t_1, t_2 + T_2), \tag{3.1}$$

where T_2 is the period of the oscillatory excitation and in this case study, DC-DC converter switching frequency is $\frac{1}{T_2}$. Now there are two problems to be solved:

- A boundary value problem in the direction of t_2.
- An initial condition problem in the direction of t_1.

As in [2], the solution for the first problem can be obtained in several ways, one of them is with the harmonic balance technique [3] and another one is FDTD. In this research, a FDTD approach is chosen. The second problem can be solved with standard integration techniques such as backward Euler (BE) and trapezoidal integration. At the beginning of the simulation the initial conditions are chosen to be array of zeros. Any initial conditions can be chosen by extrapolating the past time point initial conditions. The differential equations in the t_1 direction are stiff [4],[32]. A variable time step is then necessary for an efficient simulation.

As each step in the direction of t_1 involves the (relatively expensive) solution of a FDTD problem, it is practically important to minimize the number of time steps in the direction of t_1. Section 3.3.1 will provide an outline of proposed time step control algorithm that attempts to minimize the number of rejections.

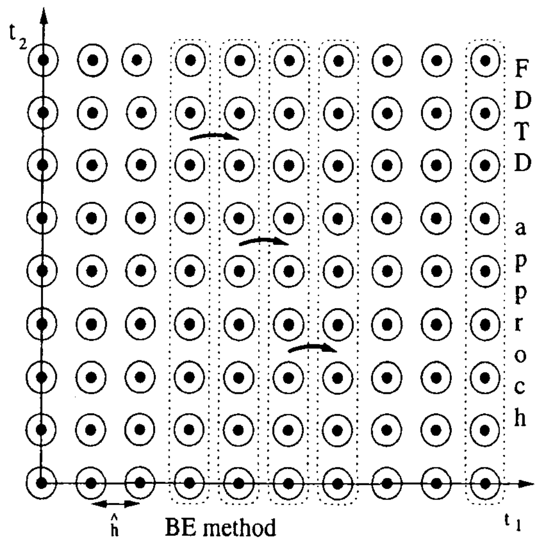


Figure 3.1: TD-ENVelope following method using MFDTD

If the MPDE Equation (2.39) is considered as an ordinary differential equation with two variables handled one at a time, and t_1 is assumed to be constant, then the differential equation will only depend on the remaining argument t_2. If an adaptive time step $\hat{h}$ is used in the t_1 direction, every value of the variable t_1 corresponds to a set of equations that are functions of t_2. The MPDE Equation (2.39) can then be written formally as a DAE in vector valued variables, using operator notation:

$$\frac{dQ(X)}{dt_1} = F(X) + B(t_1) - D_{t_2}[Q(X)], \tag{3.2}$$

where $\frac{dQ(X)}{dt_1}$ is the differential of the function $Q(X)$ in the t_1 axis, and D_{t_2} is the operator that numerically differentiates the function with respect to t_2. $F(X)$ is the vector of t_2 points as shown in Figure 3.1 $f(\hat{x}(t_1,.))$. $B(t_1)$ is the source signal with function of t_1, with values that are function of t_2. In the proposed model, an adaptive time step $\hat{h}$ is used to solve $\frac{dQ(X)}{dt_1}$ in Equation (3.2). Also the time domain FDTD method along the fast time scale t_2 is used to solve $D_{t_2}[Q(X)]$ in Equation (3.2). Instead of FDTD method in T_2 direction, shooting method or any other method could be used.

For the DC-DC converter circuit simulation, a fixed interval in the fast time axis and an adaptive time step in the slow time axis are considered. Reducing the number of Newton iterations is particularly important with TD-ENV simulations because, each step in the slow direction involves a relatively expensive solution of a nonlinear boundary type problem (FDTD) in the fast time dimension.

3.3 Time Step Control Algorithm For TD-ENV Method

In this research time step control algorithm is used in TD-ENV method. It is important to use an adaptive time step control to improve computation time and memory requirements. The procedure for the adaptive time step control algorithm using TD-ENV method is as follows:

1. The tolerance, number of points, simulation time and other simulation parameters are defined by the user. In this research extrapolation is used as an initial guess for the nonlinear equation solver.

2. The time step control algorithm uses two models for error calculations. First one is a 'coarse model' and the second one is a 'fine model'. Therefore a suitable model is chosen for the simulation at each instant. Initially the 'coarse model' is chosen, because it is cheaper (less computation time) to calculate. An initial residual $F(x_n)$ is estimated in Newton method using the 'coarse model'. Which will also give an idea how close the solution is.

3. An error function is used to determine if the trial time step is acceptable or not. An error function is defined by

$$\hat{\epsilon}_n = \| F(x_n) \| - tolerance \tag{3.3}$$

   where $F(x_n)$ is the vector of nonlinear equation values with extrapolated initial guess, and *tolerance* is the absolute tolerance defined by user. If $\hat{\epsilon}_n$ is in an acceptable range (close to zero), then the FDTD problem is solved. If not, the optimal h_n is found using dichotomy [42] to bring the time step to an acceptable range and solve the FDTD problem.

4. Then truncation error is estimated and *tolerance* is changed accordingly. Depending on the norm of LTE, the algorithm dynamically change *tolerance* and can accept or reject the calculated time step h_n.

5. The process is continued all the way to the end of the simulation time $t = t_n + h_n$.

The basic idea of the algorithm is to use a coarse model of the system to predict the optimum time step for acceptable LTE and tolerance before solving the actual FDTD problem. The error function ($\hat{\epsilon}_n$), uses current time step value - h_n and linear extrapolated value - x_n to calculate the residual. The extrapolated initial guess is found as follows:

$$x_n = x_{n-1} + \frac{h_n}{h_{n-1}} (x_{n-1} - x_{n-2}) \tag{3.4}$$

By testing extrapolated values in the nonlinear differential equations, it can be calculated to see how close the actual solution is.

3.3.1 Step Control Pseudo Code

The pseudo-code of the time step control algorithm follows:

```
1. $h = h_{last}$
2. if $|\hat{\epsilon}_n(h)| > 0.2 \times tolerance$ then
    (a) if $\hat{\epsilon}_n(h_{min}) > 0$ then $h = h_{min}$
    (b) if $\hat{\epsilon}_n(h_{max}) < 0$ then $h = h_{max}$
    (c) otherwise use dichotomy to find $h$ such that $|\hat{\epsilon}_n(h)| < 0.1 \times tol_1$
```

If the absolute value of $\hat{\epsilon}_n(h)$ is less than 20% of the tolerance, then the optimal time step size h is found as follows. If $\hat{\epsilon}_n(h_{min})$ is a positive value then h is set to be h_{min}. If $\hat{\epsilon}_n(h_{max})$ is negative, which means the time step could be larger and therefore h is set to be h_{max}. In case that the above conditions both fails ($\hat{\epsilon}_n(h)$=0), then an optimal h should be found through dichotomy by checking absolute value of $\hat{\epsilon}_n(h)$ function such that it is less than 10% of the tolerance. All of these '%' values can be chosen properly to tighten or loosen the time step control algorithm.

3. solve FDTD problem Once the time step size is determined, the actual set of equations that represent the circuit are used to solve the FDTD problem.

4. estimate norm of truncation error (ϵ) using the following equation:

$$\epsilon = \parallel X_n(:,i) - x_n(:,i) \parallel_\infty, \tag{3.5}$$

where the solution is X_n and (x_n) is the extrapolated function.

5. normalized truncation error, ϵ_N using norm of solution, X_n:

$$\epsilon_N = \parallel \frac{\epsilon}{X_n} \parallel_\infty \tag{3.6}$$

The infinity norm is used to find the truncation error, which is the norm of the difference between the solution (X_n), and the extrapolated function (x_n).

6. Calculate $\delta = \epsilon/E_{max}$, to change *tolerance* dynamically

One of the key ideas about this algorithm is dynamic tolerance changes.

7. if $\delta < 30\%$, then increase $tolerance = 3/2 * tolerance$

8. if $\delta > 50\%$, then decrease $tolerance = 1/2 * tolerance$

The parameter *tolerance* is also updated at each time step according to the estimation of the truncation error. The truncation error can be normalized by dividing the norm of the solution, $\parallel X_n(:,i) \parallel_\infty$. δ is the ratio between normalized truncation error (ϵ_N) and maximum acceptable error (E_{max}). If the ratio is relatively small, or less than 30%, then the *tolerance* can be increased by one and a half times of the original *tolerance.* If the ratio is relatively large or greater than 50%, then the tolerance can be decreased by half.

9. if $h == h_{min}$ then $\delta = 1$

10. if $\delta > 1$ then reject point and go back to step 2

11. $h_{last} = h$

12. $t_1 = t_1 + h$

13. if not finished go to 1

The value of tolerance can be adjusted to always provide a good initial guess to the subroutine that solves the FDTD problem. This is important to minimize the number of iterations to solve each nonlinear FDTD problem. This algorithm checks for error, non-convergence and varies the time step using a coarse model.

The parametric diode model is described in Chapter 2 improved the convergence of the algorithm. A linearized model is used in this case as the coarse model. In this thesis the *jdiode.m* function estimates the Jacobian matrix. The coarse model for index n is made by linearising the diode model at all grid points m in the periodic direction around the solution points at the previous point in the t_1 direction $(n-1)$:

$$\begin{aligned} i_{d_{n,m}} &\approx i_{d_{n-1,m}} + \frac{\partial i_{d_{n-1,m}}}{\partial x}\Delta x_{n,m} \\ v_{d_{n,m}} &\approx v_{d_{n-1,m}} + \frac{\partial v_{d_{n-1,m}}}{\partial x}\Delta x_{n,m}, \end{aligned}$$

where $\Delta x_{n,m}$ is the increment in the value of the state variable. Here i_d is diode current and v_d is voltage drop across voltage. The required derivatives are already available from the nonlinear solution of the previous point. From diode state space representation model Equation (2.14):

$$\frac{\partial v_{d_{n-1,m}}}{\partial x} = \begin{cases} 1 & x < v_1 \\ \frac{1}{1+\frac{x-v_1}{V_t}} & x \geq v_1 \end{cases}$$

$$\frac{\partial i_{d_{n-1,m}}}{\partial x} = \begin{cases} \frac{Is*e^{\frac{x}{V_t}}}{V_t} & x < v_1 \\ \frac{k2}{V_t} & x \geq v_1 \end{cases}$$

This algorithm checks for error, non-convergence and varies the time step using coarse model. This algorithm can follow curves better than the conventional method and have a high accuracy. However, when sharp or steep changes occur on curves, it experiences more time steps while following curves as the time-step is reduced. The other concern is the straight line interpolation does not make an accurate guess for the next time-step.

Time step and approximation error do not always have an inverse proportional relationship. Decreasing time step will not always make approximation error small. Generally small time step will results better results, but that is not always true. There is an optimum value, which will give good results. Appendix A.5 will discuss about optimum time step using Lipschitz theorem.

3.4 Conclusion

This chapter explained the details of adaptive time step algorithm. The application of TD-ENV and MFDTD methods to the simulations in this research are also discussed. MFDTD method is good for periodic excitation signals, and generally used for Steady state response. An adaptive TD-ENV technique is suitable for any circuits which has a periodic switching in one axis and non-periodic in other axis, that are difficult to simulate with traditional techniques. The techniques are further expanded in Chapter 4. The errors and stability of the systems are discussed. It is also shown how the TD-ENV method is used for transient simulation in DC-DC converter circuit and MFDTD method is used for steady state simulation in rectifier circuit.

Chapter 4

Case Study

Rectifier circuits with a low pass filter and a DC-DC converter circuit with a P and/or PI controllers are used to demonstrate the MFDTD and TD-ENV methods. In this chapter, the results of the MFDTD and TD-ENV methods are compared with the ODE results. Properties of a converter switch are studied and the simulation examples are used to verify and compare the properties of the methods. In the simulation, a DC-DC converter circuit with a PI controller is simulated using SPICE3, MATLAB and OCTAVE.

This chapter is organized as follows: Section 4.1 describes a case study of MFDTD method, where an RC circuit is used for simulations and a transformation from MPDE to ODE is also discussed. Section 4.2 gives the case study of TD-ENV method. Section 4.2 also compares the difference between the adaptive time step control algorithm and the traditional MPDE method. Section 4.3 discuss about components of DC-DC converter circuit. Nodal analysis for a DC-DC converter circuit with discritization is formulated in Section 4.4. Simulation results are shown and discussed in the Section 4.5. Fixed load and variable load simulations are performed in a P controller converter circuit in the Section 4.6.1 and Section 4.6.2 respectively. Section 4.8 is a brief summary of this Chapter.

4.1 MFDTD Case study

Switched circuits are used in the computer industry as power supplies. A regular diode is a PN junction semiconductor, which has various application usages, such as rectification, AM detector, Zener regulator, FM detector, tunnel diode oscillator and voltage doubler [33]. An input sinusoidal waveform for a forward biased diode will produce a half wave rectifier output. Using a RC filter circuit to the half wave rectifier output gives an output converter circuit shown in Figure 4.1.

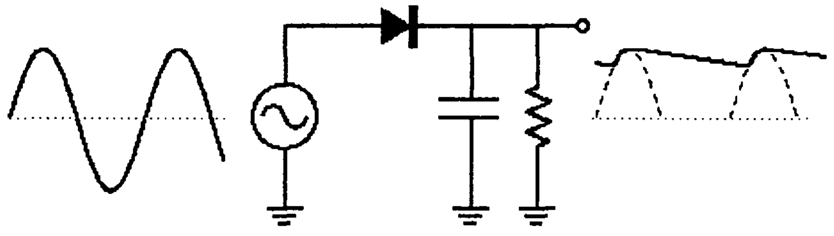


Figure 4.1: Converter Circuit

In this research, using a simple rectifier circuit, it is shown that the MPDE formulation uses less computation memory storage than the ODE formulation. Also the steady state is obtained from the MPDE results and compared with a time-marching simulation. For simulation, a pulse source is used instead of a sinusoidal input source.

The *Pulse* function act as switch with output of 1 V and 0 V. This function takes two arguments. The first is the time point at any specified switching cycle and the other is the duty cycle. The *Pulse* function has a period of T_2 and its duty cycle changes with a period T_1. It has a fall time and rise time equal to 10% of the period. It is a fast varying excitation (T_2) with slowly varying duty cycle (T_1). The *Pulse* function can be represented as

$$B(t) = pulse(\frac{t}{T_2}, 0.3 + 0.2sin(\frac{2\pi t}{T_1})) \tag{4.1}$$

where the ratio $\frac{T_1}{T_2}$=20 and $B(t)$ is a train of fast pulses where duty cycle is being modulated at a much slower rate. This circuit is shown in Figure 4.2. To obtain a faster rate, the MPDE representation of $\hat{B}(t_1, t_2)$ can be used to solve the problem. The current i_d through diode in

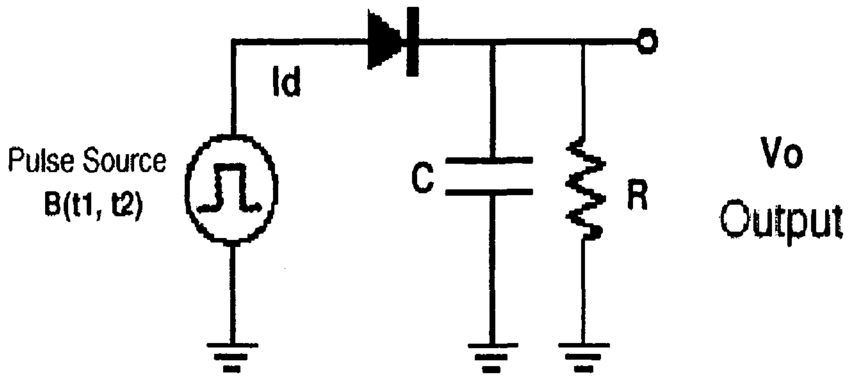


Figure 4.2: Rectifier Circuit with $B(t)$

Figure 4.2 is given by

$$i_d(pulse(t) - V_o) = \frac{V_o}{R} + C\frac{dV_o}{dt} \tag{4.2}$$

and form the nonlinear differential equation given by

$$F = i_d(Pulse(t) - V_o) - \frac{V_o}{R} - C\frac{dV_o}{dt} = 0 \tag{4.3}$$

For the simulation, the circuit parameters are chosen to be T_1 = 1 ms, T_2 = 50 μs , R = 5 kΩ and C = 1 nF.

The conventional BE method is used to formulate a set of non-linear equations that are solved using the Newton method. The discretized form of Equation (4.3) is given by

$$V_{o_i} + \frac{h}{C}\left[i_d(Pulse(t_i) - V_{o_{i+1}}) - \frac{V_{o_{i+1}}}{R}\right] - V_{o_{i+1}} = 0. \tag{4.4}$$

For the circuit given in Figure 4.2, a set of equations is derived using nodal analysis. These equations are discretized using the BE algorithm to form Equation (4.4). The simulated output V_o is shown in the Figure 4.3, where ODE is used for $B(t)$: *Pulse* function. The duty cycle of the *Pulse* function is being modulated at a much slower rate than the switching period T_2. The *Pulse* function in Equation (4.4) can be represented in bivariate form and the differential part with MPDE representation as discussed in Chapter 3. The ordinary differential form $\frac{dV_0}{dt}$

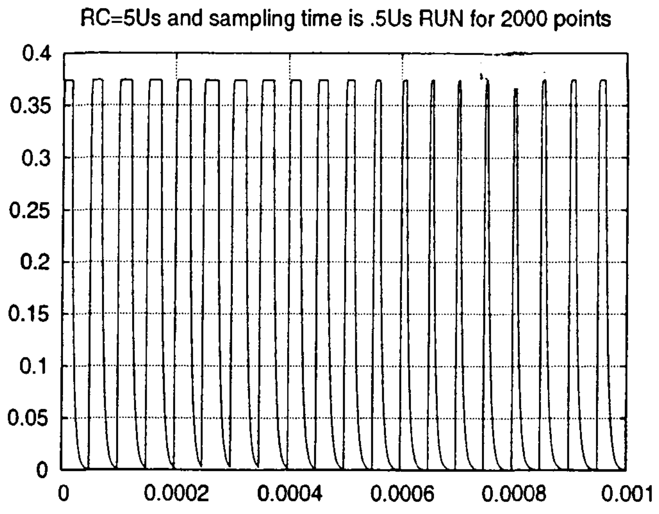


Figure 4.3: V_o: Conventional $B(t)$

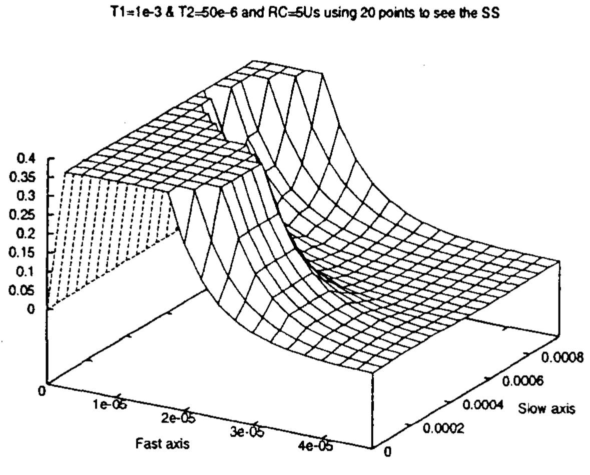


Figure 4.4: V_o: MFDTD $\hat{B}(t_1, t_2)$

can be replaced by partial differential form $\frac{\partial V_0}{\partial t_1} + \frac{\partial V_0}{\partial t_2}$. The discretized MPDE representation of Equation (4.3) becomes

$$\frac{1}{C}\left[i_d(Pulse(t_{1_i}, t_{2_j}) - V_{o_{i,j}}) - \frac{V_{o_{i,j}}}{R}\right] - \left(\frac{V_{o_{i,j}} - V_{o_{i-1,j}}}{h_1}\right) - \left(\frac{V_{o_{i,j}} - V_{o_{i,j-1}}}{h_2}\right) = 0. \qquad (4.5)$$

Figure 4.4 shows the duty cycle change of the sinusoidal signal. There the fast varying excitation (T_2) along with RC effects are shown in the Fast axis and the slowly varying sinusoidal duty cycle (T_1) is shown in the Slow axis.

Figure 4.5 shows a comparison of both simulation results. A linear interpolation method is used on MPDE as shown in Figure 4.6 to obtain the results. ODE simulation result contains the transient information, where the simulation time step is smaller than the time step used in MFDTD simulation. The MFDTD result contains steady state information, where a large time step used(1000 times as ODE time step). Increasing the grid size on Fast axis will improve the accuracy on Figure 4.5, which will lead these two graphs to super impose on each other. To reduce the discrepancies between these methods, one can use a higher order interpolation method instead of linear interpolation on Figure 4.6. On Figure 4.6 the box point represents the 'ODE' result and the circle point represents the 'MPDE' result.

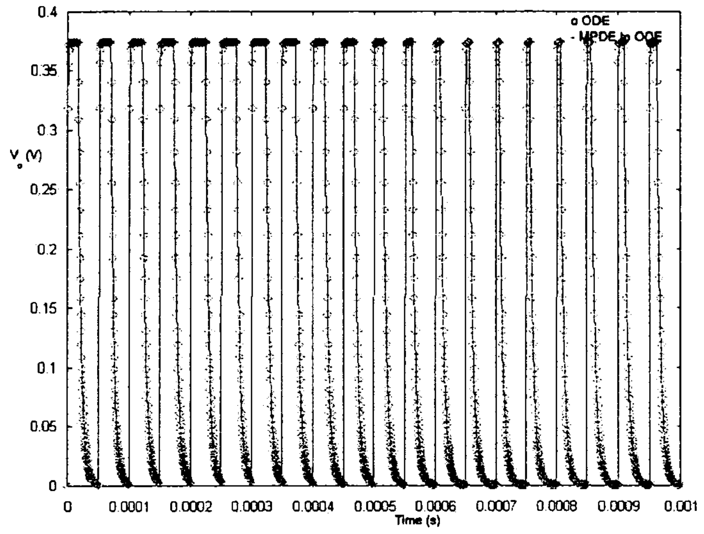


Figure 4.5: V_o: MPDE to Conventional

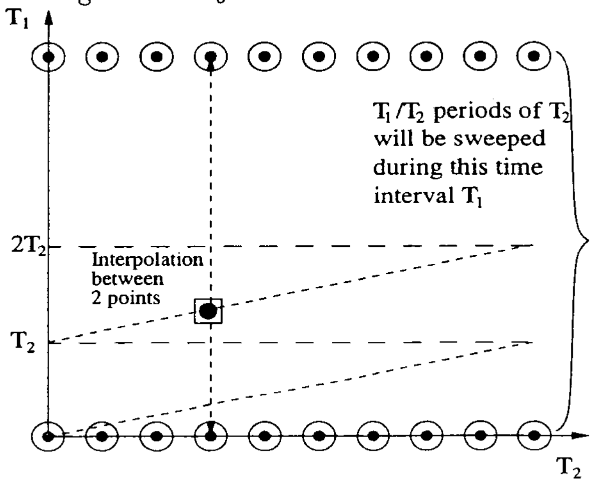

T_1
T_1/T_2 periods of T_2 will be sweeped during this time interval T_1
$2T_2$
Interpolation between 2 points
T_2
T_2

Figure 4.6: Saw tooth path

4.2 TD-ENV Case study

The TD-ENV method is commonly used to simulate communication system circuits, because it produces a more efficient and accurate prediction of envelope transient. This technique is capable of handling circuits with nonlinearities on a fast time scale such as power converters, switched capacitor filters and switching mixers. Also it is used to predict the spectral re-growth of a mixer or the transient behavior of DC-DC converter circuit.

In this research, a rectifier circuit is simulated to show the comparison between the adaptive time step control algorithm and the traditional MPDE method. The circuit is given in Figure 4.7. There are two unknown variables (x and v_L) in this circuit. Here, x represents the diode state variable and v_L represents the output voltage. The equations of the circuit are given by

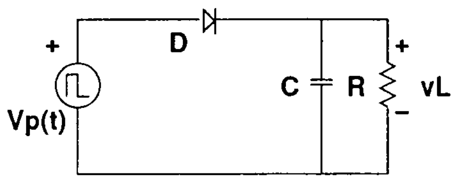


Figure 4.7: TD-ENV simulation

$$\begin{aligned} f_1(v_L, x) &= v_p(t) - v_d(x) - v_L &= 0 \\ f_2(v_L, x) &= i_d(x) - \tfrac{v_L}{R} - C\tfrac{dv_L}{dt} &= 0 \end{aligned} \tag{4.6}$$

where $v_d(x)$ is diode voltage, v_p is the pulse function with 2 V or 0 V output, and $i_d(x)$ is diode current. The PDE derived from Equation 4.6 is given by

$$\begin{aligned} v_p(t_1, t_2) - v_d(x) - v_L &= 0 \\ i_d(x) - \tfrac{v_L}{R} - C\left(\tfrac{\partial v_L}{\partial t_1} + \tfrac{\partial v_L}{\partial t_2}\right) &= 0. \end{aligned} \tag{4.7}$$

Applying the BE rule to Equation (4.7) gives the following discretization

$$\begin{aligned} v_p(t_{1i}, t_{2i}) - v_{di,j} - v_{Li,j} &= 0 \\ i_{di,j} - \tfrac{v_{Li,j}}{R} - C\left(\tfrac{(v_{Li,j} - v_{Li-1,j})}{h_1} + \tfrac{(v_{Li,j} - v_{Li,j-1})}{h_2}\right) &= 0 \end{aligned} \tag{4.8}$$

where i and j represent the index value in the t_1 and t_2 directions, respectively. The number of grid points used in the t_2 axis is represented by j_{max}. The periodic boundary conditions for the circuit are

$$\begin{cases} v_{Li,-1} = v_{Li,j_{max}} \\ x_{i,-1} = x_{i,j_{max}} \end{cases}$$

These boundary conditions are used for non linear Equation (4.8) to find the root using Newton method.

Figures 4.8 and 4.9 show the simulation results of the output voltage response and the diode state variable response respectively. It can be observed that the diode state variable presents large variations in both time scales.

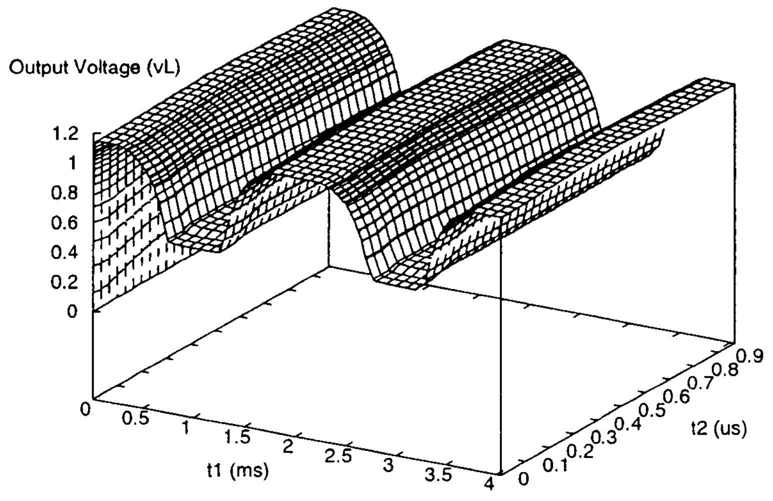


Figure 4.8: Load voltage vs. Time

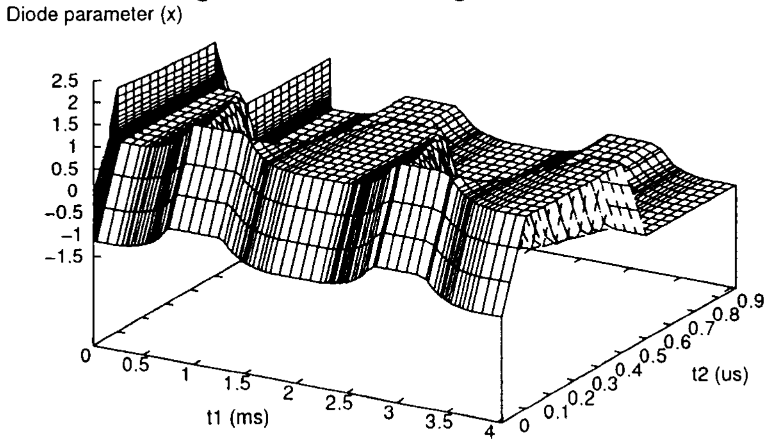


Figure 4.9: Diode state variable vs. Time

The traditional time step control method used only the MPDE formulation and LTE [37], however the proposed model used the adaptive time step control with MPDE formulation and 'coarse' model. Traditional step control uses only the truncation error to determine the next time step.

The comparison between traditional MPDE simulation and the proposed time step control algorithm is as follows: Accepted time point means, that the LTE is acceptable. The proposed

Table 4.1: Traditional vs. Proposed model computation time points

	Accepted	Rejected	Total
Proposed time step control algorithm	354	76	430
Traditional MPDE	581	324	905

method have less than half FDTD computations than the traditional method. The simulations also show that few time steps are rejected when compared to traditional time step control algorithm.

4.3 The Converter Circuit as a Controlled System

A boost converter is an electronic power supply that adjusts the voltage level from a given DC source to provide power to a variable load at a higher (fixed) DC level. Coupled with this basic operation is other functionality involving regulation and ripple control. Switching regulators offer higher efficiency than linear regulators. A switching regulator is a circuit that uses an inductor, a transformer or a capacitor as an energy storage element to transfer energy from input to output in discrete packets. Feedback circuitry regulates the energy transfer to maintain a constant voltage within the load limits of the circuit. The basic circuit can be configured to step up (boost), step down (buck), or invert output voltage with respect to input voltage.

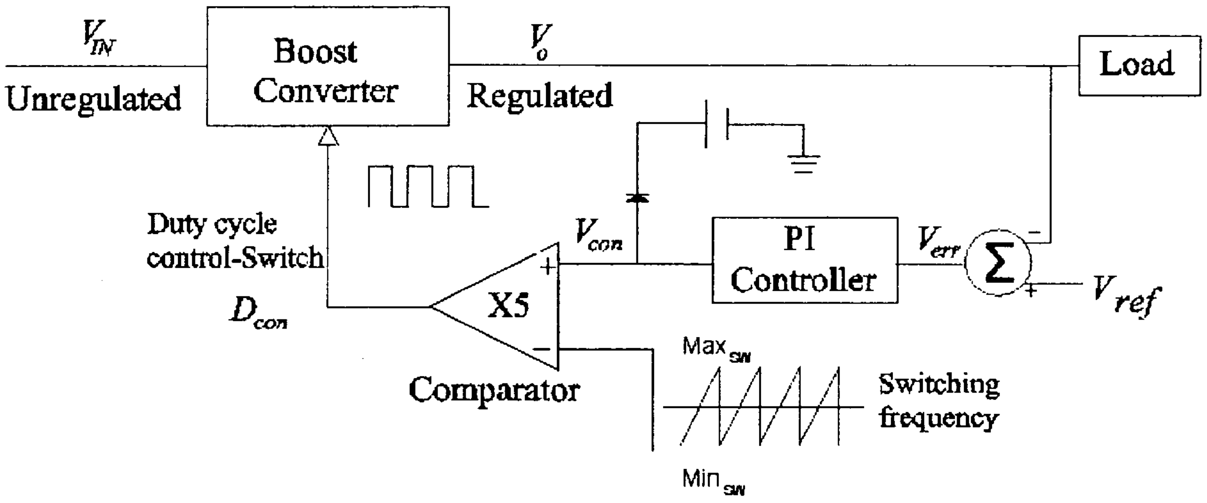


Figure 4.10: Converter Circuit Block Diagram

In order to maintain a constant output voltage under varying load conditions, feedback must be introduced as shown in Figure 4.10. This feedback must ultimately be able to directly affect the pulse width of the signal fed into the switching control signal D_{con}. By varying this pulse width, the switch can adjust to the load conditions providing more power for a larger load and vice versa. The primary components of the converter circuit are shown in Figure 4.10.

- **Boost Converter:** The boost converter converts an input voltage to a higher output voltage. Boost converters are used in battery powered devices, where the electronic circuit requires a higher operating voltage than the battery can supply, e.g. notebooks, mobile phones and camera flashes. It is important to first provide a general description of how it works. In Figure 4.11, when the switch closes, the voltage across the inductor is V_{in}. In this research, since an ideal switch and an inductor are not available, there will be some voltage drop present in the inductor internal resistance and the switch. The inductor current will ramp up linearly when the switch is closed. When the switch opens, the

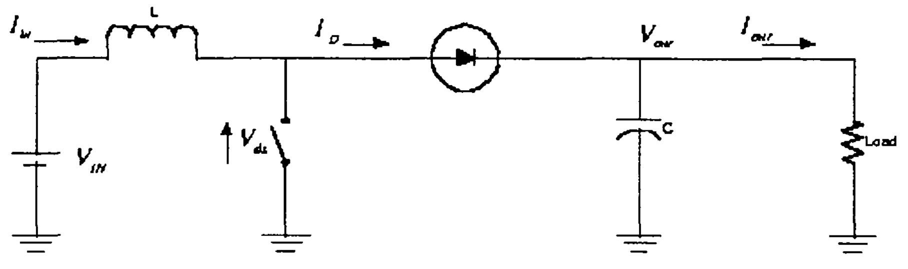


Figure 4.11: Boost converter

current flows through the diode into the capacitor and the load. When switch is open and the capacitor is charged, it will supply current to the load.

In a steady-state operating condition the average voltage across the inductor over the entire switching cycle is zero. This implies that the average current through the inductor is also in steady state, as shown in Figure 4.12 for continuous mode. When the switch is 'ON', $V_L = V_{IN}$ and when the switch is 'OFF', $V_L = V_{IN} - V_O$ for a constant V_O. The average inductor voltage in steady state must equal zero,

$$V_{IN}t_{on} = -(V_{IN} - V_O)t_{off} \tag{4.9}$$

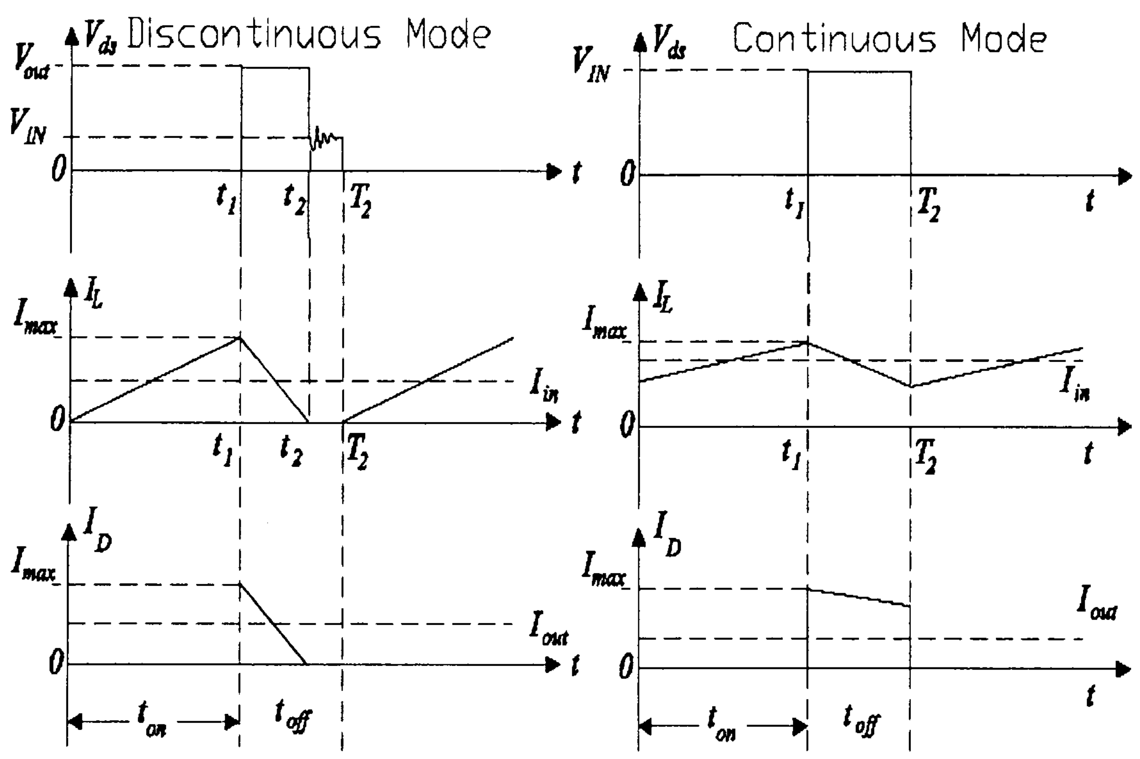


Figure 4.12: Discontinuous and Continuous mode boost converter

For the continuous mode steady state condition, the following equation

$$V_O = [1/(1 - dutycycle)]V_{IN}, \tag{4.10}$$

applies if there are no internal resistances present in L, C, switch and diode. Also the $dutycycle = \frac{t_{on}}{T_s}$ is considered to formulate Equation (4.9) with switch ON time t_{on} and switching period T_s.

If an ideal situation with continuous conduction mode is considered and V_{in} is chosen to be 6 V to achieve the 12 V output, a duty cycle equal to $\frac{1}{2}$ is required. For the simulation, a switching frequency of 100 KHz is used. In this research, this duty cycle is regulated to obtain the appropriate output voltage on the load side to minimize the error through feedback.

A distinction is drawn between discontinuous and continuous mode depending on whether the inductor current I_L reduces to zero during the off-time or not. In discontinuous mode, the inductor current I_L will go to zero every period. When the inductor current becomes zero at t_2 in Figure 4.12, the voltage V_{ds} jumps to the value of V_{in}. This is because in this case $V_L = 0$. The drain-source capacitance is in parallel with the diode-junction capacitance and forms a resonant circuit with the inductance L. This is stimulated by the voltage jump across the diode. The voltage V_{ds} then oscillates and fades away. However in this research the equation formulation is not separated as continuous nor discontinuous modes. For the simulation, general formulas for switch 'ON' and 'OFF' conditions are produced and the difference equations are solved using nonlinear equation solver.

For the simulation, the larger the value of the inductor L, the smaller the current ripple δI_L. Therefore L should be chosen to achieve an adequately small δI_L. With a larger δI_L, the voltage ripple of the output voltage V_o becomes clearly larger while the physical size of the inductor decreases marginally. The switching losses of the transistor also become larger as F_s increases.

- **Comparator:** The output voltage from the boost converter is compared with a set reference voltage V_{ref} by means of a differential amplifier to produce an error voltage $V_{err} = V_{ref} - V_o$. Here the output voltage V_o across the boost converter load is fed into the inverting side of the differential amplifier shown as X1 on Figure 4.13. The non-inverting side of the differential amplifier is fed by a 12 V DC reference voltage V_{ref} as shown in Figure 4.13. An ideal op-amp input difference voltage is zero, $V_- = V_+$, and the inputs draw no current.

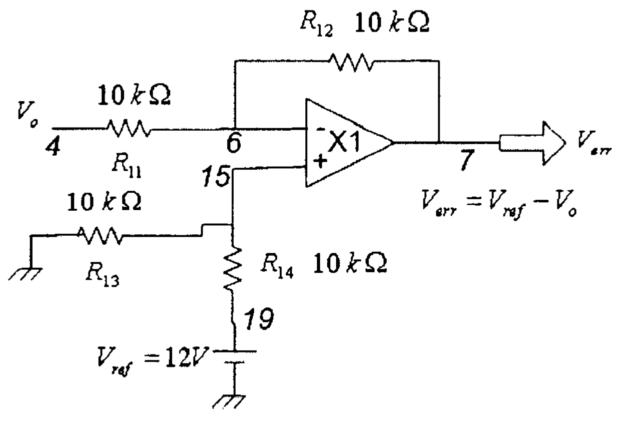
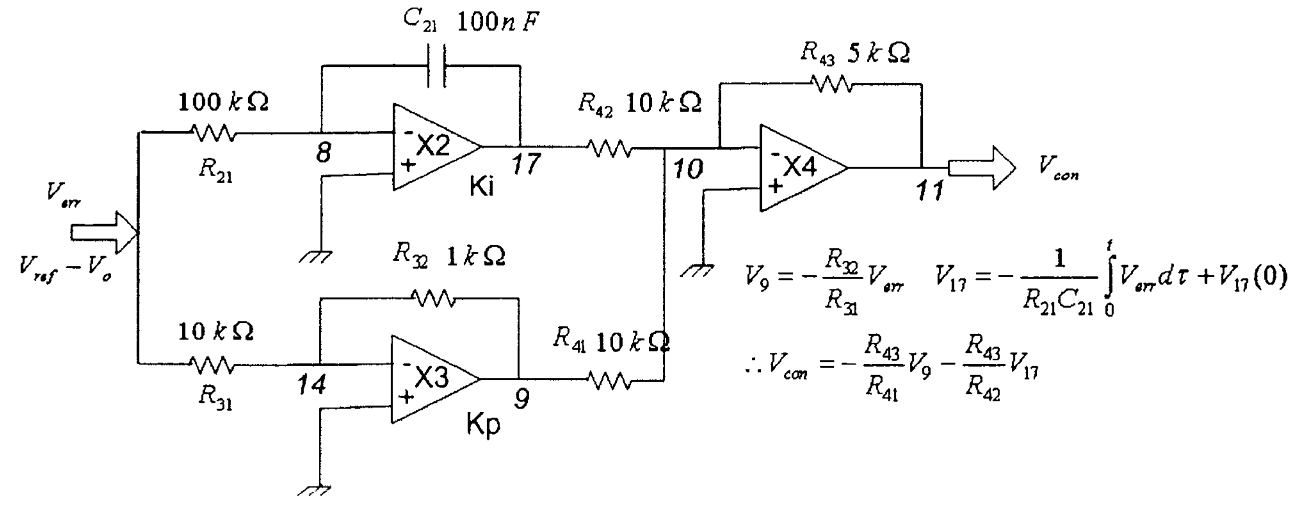


Figure 4.13: Differential Amplifier

- **PI Controller:** The error voltage V_{err} from the differential amp is fed to a PI controller. The steady state error cannot be eliminated by a P controller alone. The purpose of the integral gain is to eliminate the steady state error. The main concept of the PI controller algorithm is to cancel the largest time constant of the circuit, and to introduce an integrating effect to the circuit. The controller will respond proportionally to any change in V_{err} by producing an output signal that reflects the rate of change of the input signal. From the reference model shown in Figure 4.14, the transfer function for the PI



C_{21} $100nF$
$100k\Omega$
R_{21}
8
X2
Ki
17
R_{42} $10k\Omega$
R_{43} $5k\Omega$
10
X4
11
V_{con}
V_{err}
$V_{ref} - V_o$
R_{32} $1k\Omega$
$10k\Omega$
R_{31}
14
X3
Kp
9
R_{41} $10k\Omega$
$V_9 = -\frac{R_{32}}{R_{31}} V_{err}$
$V_{17} = -\frac{1}{R_{21}C_{21}} \int_0^t V_{err} d\tau + V_{17}(0)$
$\therefore V_{con} = -\frac{R_{43}}{R_{41}} V_9 - \frac{R_{43}}{R_{42}} V_{17}$

Figure 4.14: PI Controller Circuit with Summing Amplifier

controller can be derived using a proportional gain K_p from X3 and an integral gain K_i

from X2. K_p and K_i are

$$K_p = \frac{R_{32}}{R_{31}} \tag{4.11}$$

$$K_i = \frac{1}{R_{21}.C_{21}}. \tag{4.12}$$

Applying V_{err} into the proportional controller gives the output V_9

$$V_9 = -K_p V_{err}. \tag{4.13}$$

Applying V_{err} into the integral controller gives the output V_{17}

$$V_{17} = -K_i \int_{t_1}^{t_2} V_{err}\, d\tau + V_{17}(t_1). \tag{4.14}$$

The outputs V_9 from Equation (4.13) and V_{17} from Equation (4.14) are fed into the summing amplifier to produce,

$$V_{con} = -\left(\frac{R_{43}}{R_{41}}\right) . (-K_p V_{err}) - \left(\frac{R_{43}}{R_{42}}\right) . \left(-K_i \int_{t_1}^{t_2} V_{err} d\tau + V_{17}(t_1)\right) \tag{4.15}$$

where $\frac{R_{43}}{R_{41}}$ was chosen to be $\frac{1}{2}$. A new state variable U is introduced such that

$$U = \int_0^t V_{err}\, d\tau \Rightarrow \frac{dU}{dt} = V_{err}. \tag{4.16}$$

Substituting Equation (4.16) into Equation (4.15) gives

$$V_{con} = \frac{K_p}{2} V_{err} + \frac{K_i}{2} U. \tag{4.17}$$

The state variable U can be solved by including it to the nonlinear equations. This will be discussed in Section 4.4.

The duty cycle controller parameter D_{con} can be written as a function of V_{con} as follows

$$D_{con} = \frac{1}{2} + \left(\frac{1}{Max_{SW} - Min_{SW}}\right) . \left(\frac{K_p}{2}(V_{ref} - V_o) + \frac{K_i}{2} U\right) \tag{4.18}$$

where Max_{SW} and Min_{SW} are the maximum value and the minimum value of the saw-tooth wave form used in comparator circuit in shown in Figure 4.10. This duty cycle controller parameter D_{con} in Equation (4.18) will be discussed in detail in Section 4.4.

- **Switch - Pulse function:** Generally two sets of equations are used in a DC-DC converter circuit formulation, they are Diode/Switch ON state and OFF state model equations. Instead of two sets of equations, this study uses a switch function (*Pulse*) to have one set of equations for the converter circuit. This switch function takes an important roll in convergence. The switch transition is defined using one resistor which varies from 10 $m\Omega$ to 10 $M\Omega$. This steep transition on the resistor value gives convergence problems during the simulation time. To improve convergence, the following exponential behavior is introduced in the switch function instead of a linear rise time form R_{on} to R_{off}.

$$y = \frac{e^{k(1-\frac{t}{t_r})} - 1}{e^k - 1} \tag{4.19}$$

where t_r is rise time, y varies from 0 to 1, $k > 1$ and for the simulation it is chosen to be $k = 20$. Then R_s is calculated by summing the R_{on} and $(R_{off} - R_{on})y$. R_s is the output resistor which varies from 10 $m\Omega$ (R_{on}) to 10 $M\Omega$ (R_{off}). The exponential behavior in the switch function improves convergence because it produces a continues smooth change in the switch resistor. As a results this exponential behavior is also introduced during the fall time, that can be seen at the initial time of the Figure 4.15.

Figure 4.15 shows a complete switching cycle when the constant duty cycle is $\frac{1}{2}$. On this figure, the Y axis is in semi logarithmic scale to show the clear difference from R_{on} to R_{off} transition for R_s. The switch function without the exponential behavior gives convergence problems starting from the beginning of R_{on} transition.

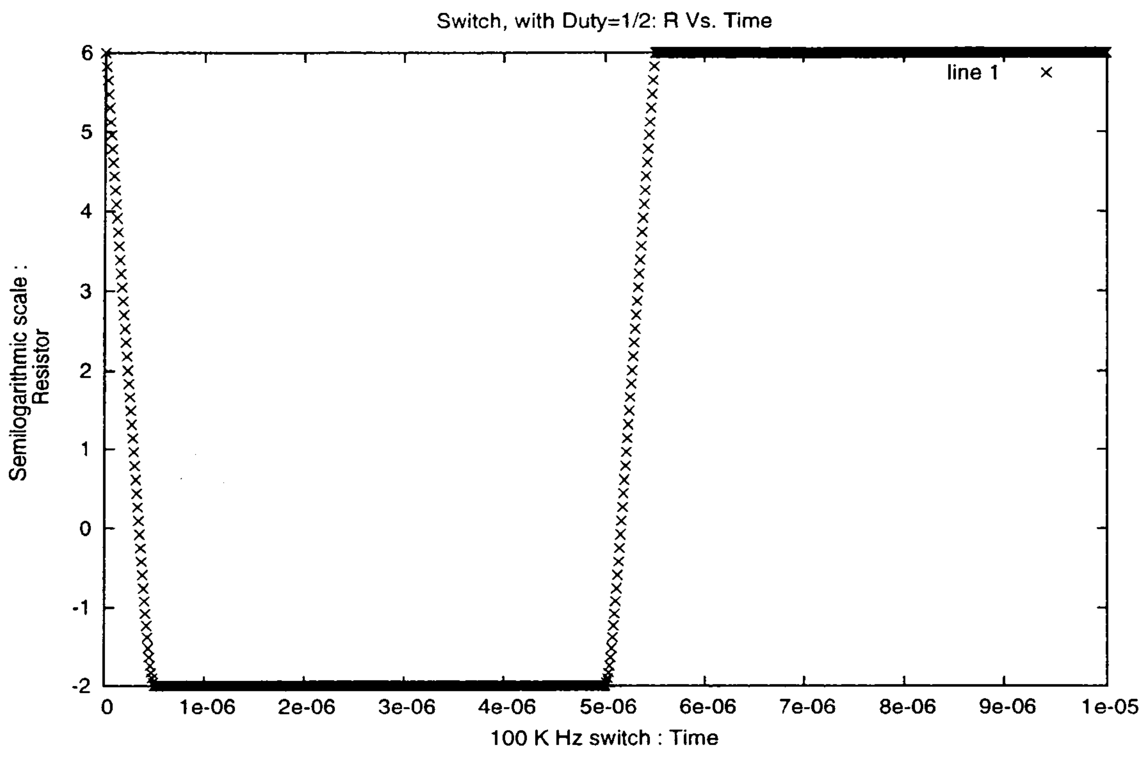


Figure 4.15: Switch with a duty cycle of $\frac{1}{2}$ for a period of T_2

4.4 TD-ENV and ODE Cicuit Formulations for the Boost Converter

The TD-ENV is based on a time domain scheme, in which the nonlinearity is caused by the switch and the diode and it is resolved by time domain integration. In this research, the set of nonlinear equations are solved using the t_2 time axis for every time point in the t_1 time axis.

The converter circuit given in Figure 4.16 is considered to formulate the circuit equations,

- application of KVL in loop 1 gives

$$F_1 = V_{in} - i_L r_{SL} - V_d - V_o - L\frac{di_L}{dt} = 0 \tag{4.20}$$

- application of KCL in Node 2 gives

$$F_2 = \frac{1}{R_s}(V_d + V_o) + i_d - i_L = 0 \tag{4.21}$$

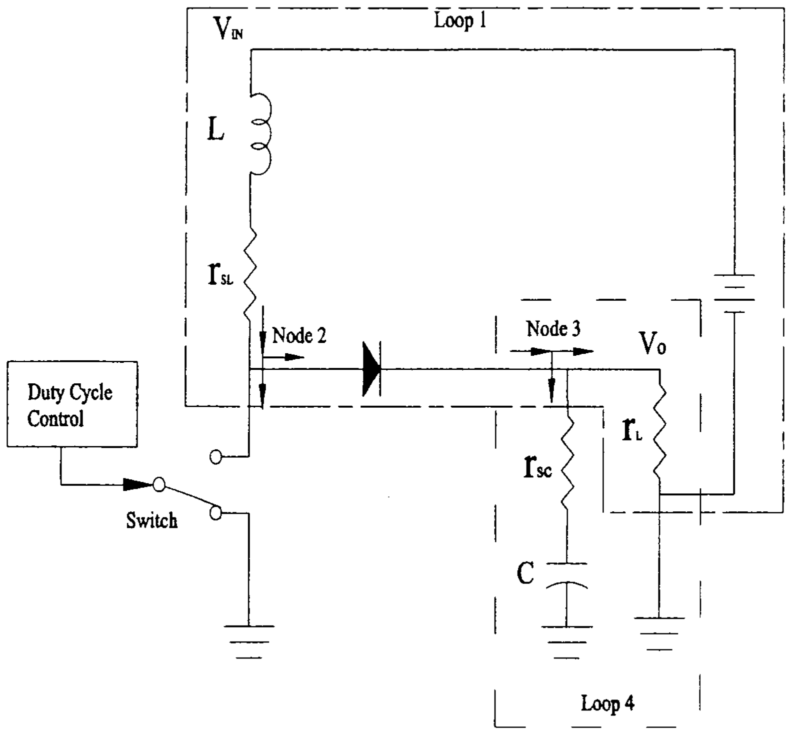


Figure 4.16: Converter Circuit

where R_s depends on the switch function. The switch function takes two arguments; D_{con} and current time point in t_2 ($i \times h_2$, $i \in (1..N)$, where N is grid size). The duty cycle D_{con} is given by

$$D_{con} = \frac{1}{2} + \mathbf{K_p} V_{err} + \mathbf{K_i} \int V_{err} d\tau \tag{4.22}$$

where $\mathbf{K_p}$ is the proportional gain ($\mathbf{K_p} = \frac{K_p}{2(Max_{SW} - Min_{SW})}$) and $\mathbf{K_i}$ is integral gain ($\mathbf{K_i} = \frac{K_i}{2(Max_{SW} - Min_{SW})}$) as discussed in Equation (4.18).

- application of KCL in Node 3 gives

$$F_3 = i_d - V_o \frac{1}{r_L} - C \frac{dV_c}{dt} = 0 \tag{4.23}$$

- application of KCL in Loop 4 gives

$$F_4 = \frac{(V_o - V_c)}{r_{SC}} - C \frac{dV_c}{dt} = 0 \tag{4.24}$$

- Equation (4.16) for the set of nonlinear equations gives

$$F_5 = \frac{dU}{dt} - V_{ref} + V_o = 0 \tag{4.25}$$

where $V_{err} = V_{ref} - V_o$ and $U = \int V_{err} d\tau$.

The discretizations used in the ODE formulation are given by

$$\begin{aligned} \frac{di_{L_i}}{dt} &= \frac{(i_{L_i} - i_{l_{i-1}})}{h_2} \\ \frac{dV_{c_i}}{dt} &= \frac{(V_{c_i} - V_{c_{i-1}})}{h_2} \\ \frac{dU_i}{dt} &= \frac{(U_i - U_{i-1})}{h_2} \end{aligned} \tag{4.26}$$

where h_2 is the time step, U_i, i_{L_i} and V_{c_i} are current state values and U_{i-1}, $i_{L_{i-1}}$ and $V_{c_{i-1}}$ are previous time point state values.

The discretizations of MPDE on slow axis t_1 and fast axis t_2 are as follows:

- On slow axis t_1:

$$\begin{aligned} \frac{\partial i_{L_j}}{\partial t_1} &= \frac{(i_{L_j} - i_{L_{j-1}})}{h_1} \\ \frac{\partial V_{c_j}}{\partial t_1} &= \frac{(V_{c_j} - V_{c_{j-1}})}{h_1} \\ \frac{\partial U_j}{\partial t_1} &= \frac{(U_j - U_{j-1})}{h_1} \end{aligned} \tag{4.27}$$

- On fast axis t_2 using BE:

$$\begin{aligned} \frac{\partial i_{L_i}}{\partial t_2} &= \frac{(i_{L_i} - i_{L_{i-1}})}{h_2} \\ \frac{\partial V_{c_i}}{\partial t_2} &= \frac{(V_c - V_{c_{i-1}})}{h_2} \\ \frac{\partial U_i}{\partial t_2} &= \frac{(U_i - U_{i-1})}{h_2} \end{aligned} \tag{4.28}$$

- On fast axis t_2 using 3-point rule:

$$\begin{aligned} \frac{\partial i_{L_i}}{\partial t_2} &= \frac{(i_{L_{i+1}} - i_{L_{i-1}})}{2h_2} \\ \frac{\partial V_{c_i}}{\partial t_2} &= \frac{(V_{c_{i+1}} - V_{c_{i-1}})}{2h_2} \\ \frac{\partial U_i}{\partial t_2} &= \frac{(U_{i+1} - U_{i-1})}{2h_2} \end{aligned} \tag{4.29}$$

The above ODE and MPDE formulations (Equations (4.26 - 4.29)) are used to simulate the circuits in this research. In MPDE formulation, BE is used on t_1 axis and the 3-point rule is used to formulate the nonlinear equations in FDTD technique on t_2 axis. The 3-point center differential formulation gives more accurate results than the regular BE rule.

4.5 Time Marching Simulation of the Boost Converter

In this research, the simulations of converter circuit are performed as follows:

1. **Matlab-Simulink:** In Matlab, variable time step with ODE15S(STIFF/NDF) integration method is used for the simulation. All other available integration methods in Matlab couldn't simulate this circuit due to the stiffness of the circuit. The disadvantage in Matlab simulation is that it takes high storage. The simulation results are given in the Appendix A.6.

2. **SPICE:** In SPICE, the gear integration method is used. SPICE does not accept any step size larger than 20 ns. This leads to many disadvantages on SPICE. One of them is high memory requirements. Therefore, the spice code is simulated on a large memory system. Figures 4.17 and 4.18 show the output voltage (V_o) and error voltage ($V_o - V_{ref}$) for the SPICE simulation. From Figure 4.17 it can be observed that the peak value of V_o is at 15.93 V. The rise time of the pulse function is 3% of T_2. However the pulse function gives 16.38 V for the same circuit and for the same rise time.

3. **ODE in Octave:** The Euler method with the ODE technique is used for the simulation in Octave. The root finding algorithm in Newton-Raphson method re tries many time steps to get the most accurate solution. All these trial time steps are considered as rejected time points. The number of rejected time points play a vital roll in convergence rate. The convergence rate is defined as the reciprocal of the number of rejected time points. Faster convergence means, that number of rejected points are less. The convergence rate of the Euler method is smaller than the convergence rate of the trapezoidal method. To achieve a good approximation tolerance of 1% in the BE and trapezoidal methods the following time steps can be used:

   - $h_{min} = \frac{T_2}{(n)2^{20}}$

- $h_{min} = \frac{T_2}{(n)2^{15}}$

where n is number of time points used in periodic t_2 axis. This is observed through trial and error. From the h_{min} differences, it can be concluded that the Trapezoidal method gives better and faster convergence for the proposed DC-DC converter circuit. This h_{min} is not acceptable because it is too small. If a small time step is chosen, the simulation will take too long to finish. As a result, the acceptable LTE tolerance is increased to 5%. Figures 4.19 and 4.20 show the output voltage (V_o) and error voltage ($V_o - V_{ref}$) for Octave code ODE simulation. The peak value of V_o for ODE in Octave varies with the rise time of the switch function. It can be observed in Figure 4.19 that the variations in V_o during 0 - 1 ms require a smaller time step than during 1 ms - 50 ms.

Table 4.2 gives the peak values and the convergence rates of V_o for various rise times of the switch function. The acceptable LTE of 5% and simulation time are the same for all rise times. When the rise time is reduced, the LTE gets larger and the convergence rate

Table 4.2: Peak values and Convergence rates of V_o for various switch function rise times.

	3% of T_2	5% of T_2	7.5% of T_2
peak value	16.38	16.08	15.62
Time taken (convergence rate)	12674 s (79 μ)	10361 s(97 μ)	9452 s (106 μ)

gets smaller or the nonlinear equations need extremely small time step to converge with an acceptable LTE. This behavior is due to the abrupt switching transitions from 1 MΩ to 0.01 mΩ. At the boundaries of this transition time, the simulation requires extremely small time step such as $\frac{T_2}{(20)2^{20}}$ for convergence.

Figures 4.21 and 4.22 show the inductor current (i_L) and the diode state variable x output for Octave simulation. It can be observed that during the start time (0 to 1 ms) the inductor current becomes zero and the diode state variable: X experiences an abrupt variation.

In the simulation, the DC-DC converter circuit is investigated using different state variables. When the number of state variables are reduced to four from five, such as [i_L, X, V_o, V_c] from [i_L, X, V_o, V_c, U], the computation time is reduced as well. For the four state variable representation, the trapezoidal integration method is used to obtain the response of the integral controller. The integral state variable U is an important unknown state variable, because it depends on the time step. There are some computation time difference between Euler and Trapezoidal integration methods, but they are not significant.

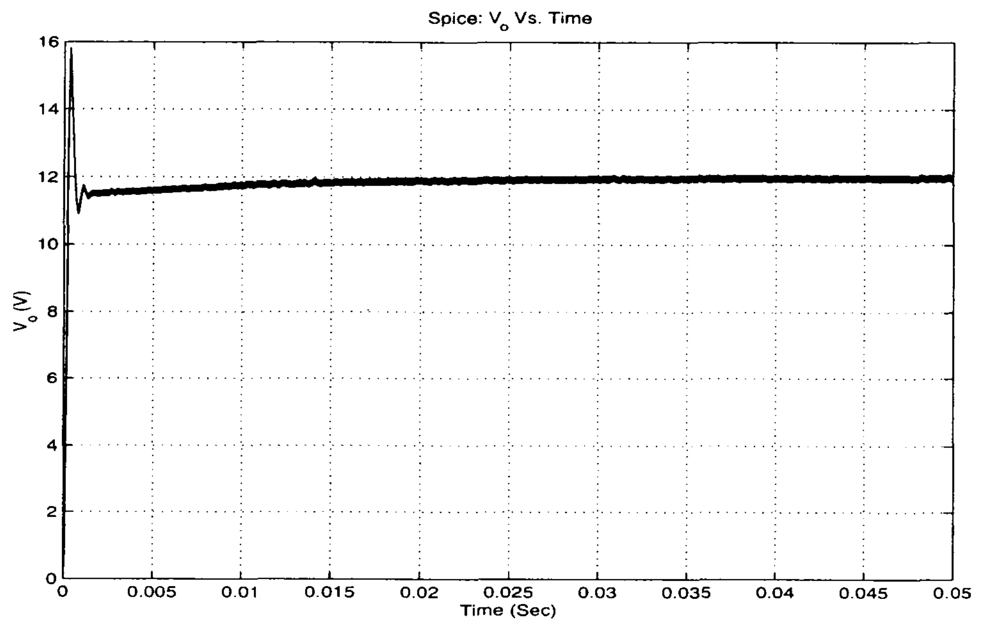


Figure 4.17: V_o vs. Time

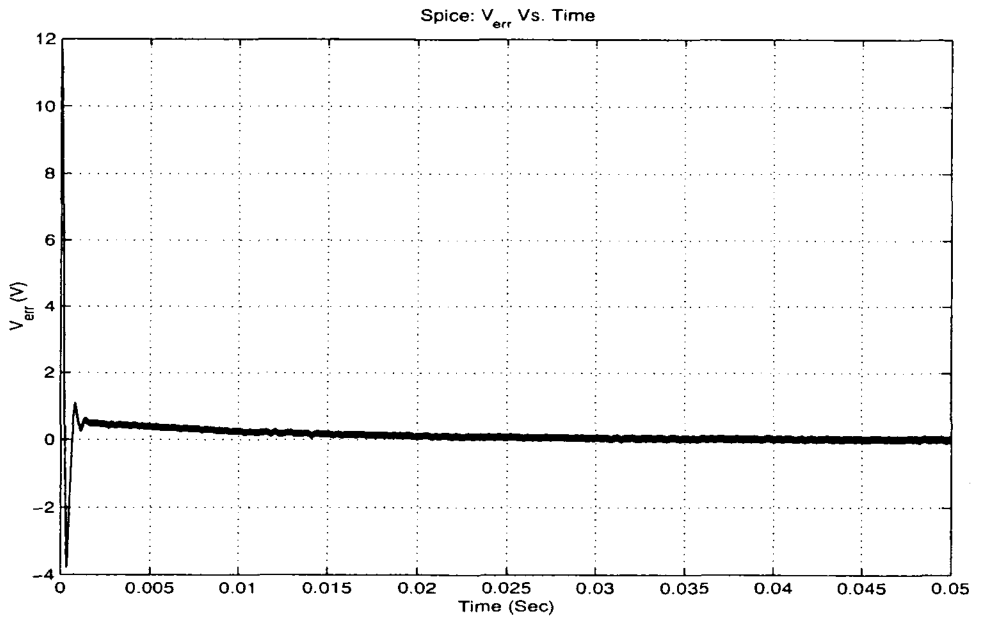


Figure 4.18: V_{err} vs. Time

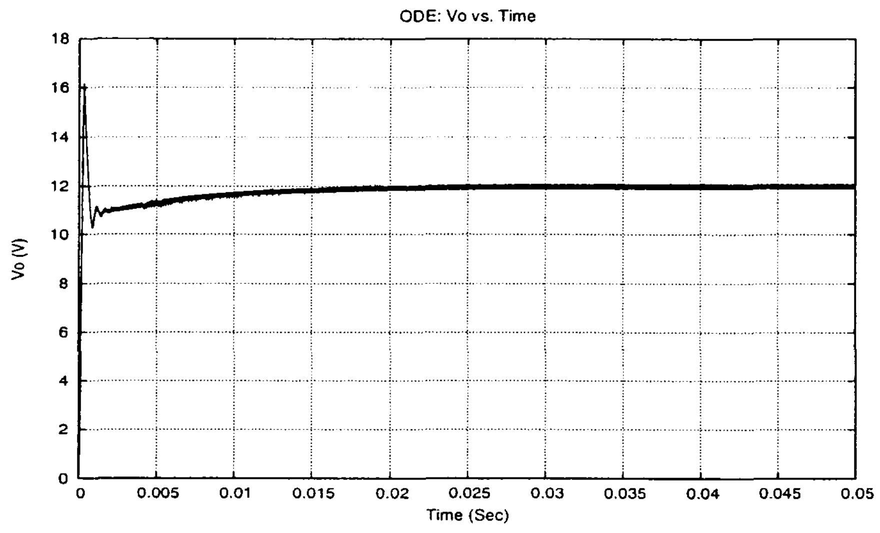


Figure 4.19: V_o vs. Time

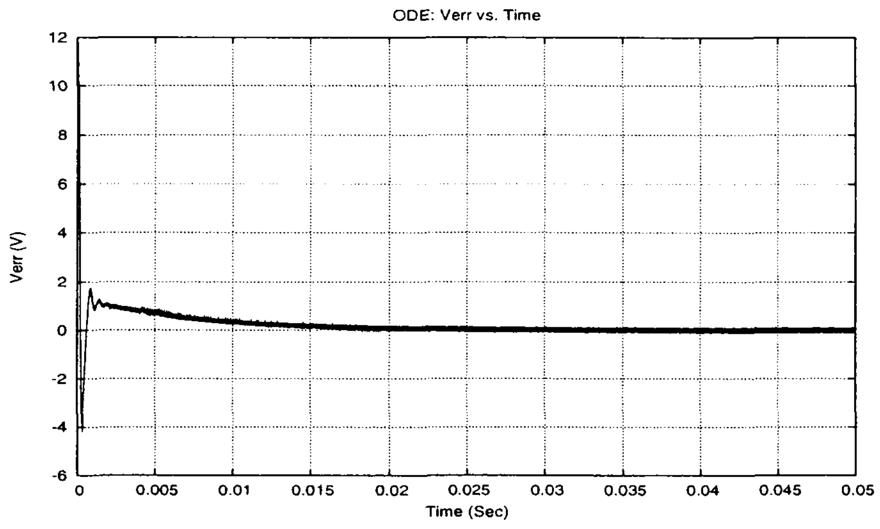


Figure 4.20: V_{err} vs. Time

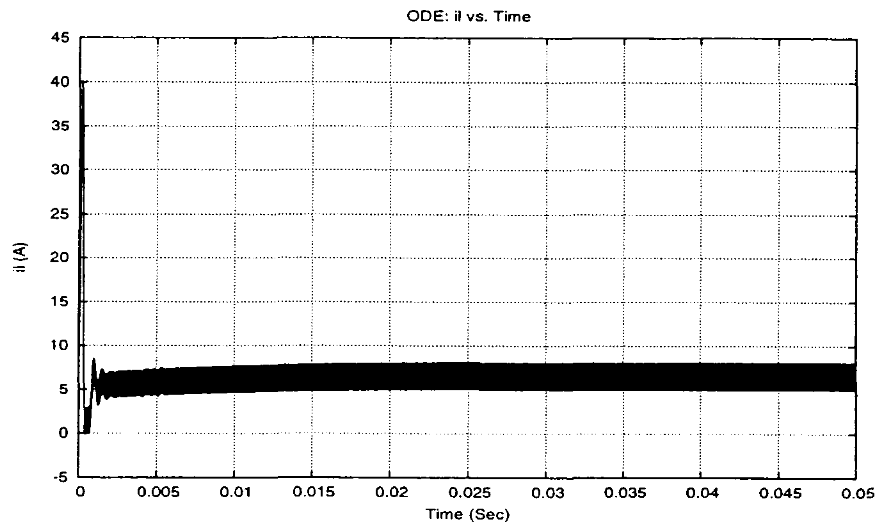


Figure 4.21: I_L vs. Time

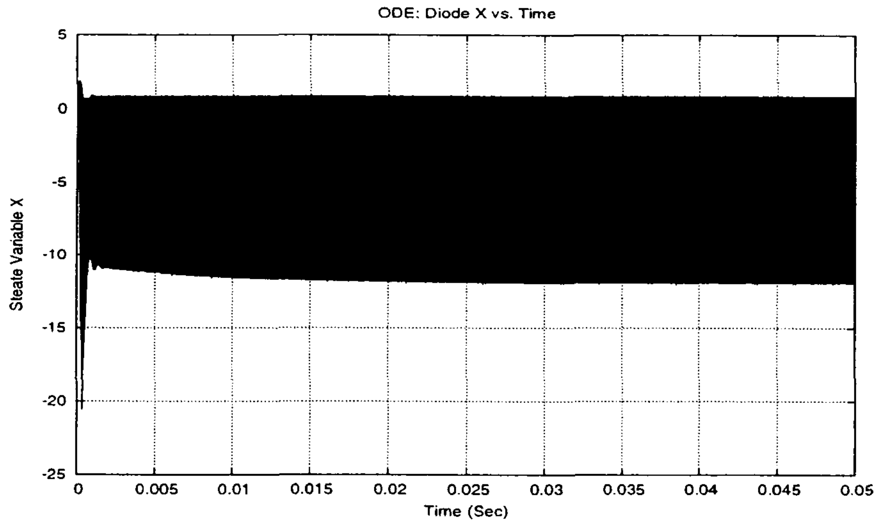


Figure 4.22: Diode X vs. Time

All the parameters are kept the same for all three simulations and an acceptable LTE of 5% is considered for the results. The settling times for MATLAB, SPICE and OCTAVE simulations are considered to be 50 ms. The over-shoot peak values of the simulations vary due to the nonlinear characteristics of the switch function. The MATLAB simulation results are not comparable with the other two simulations because MATLAB uses the ideal circuit components.

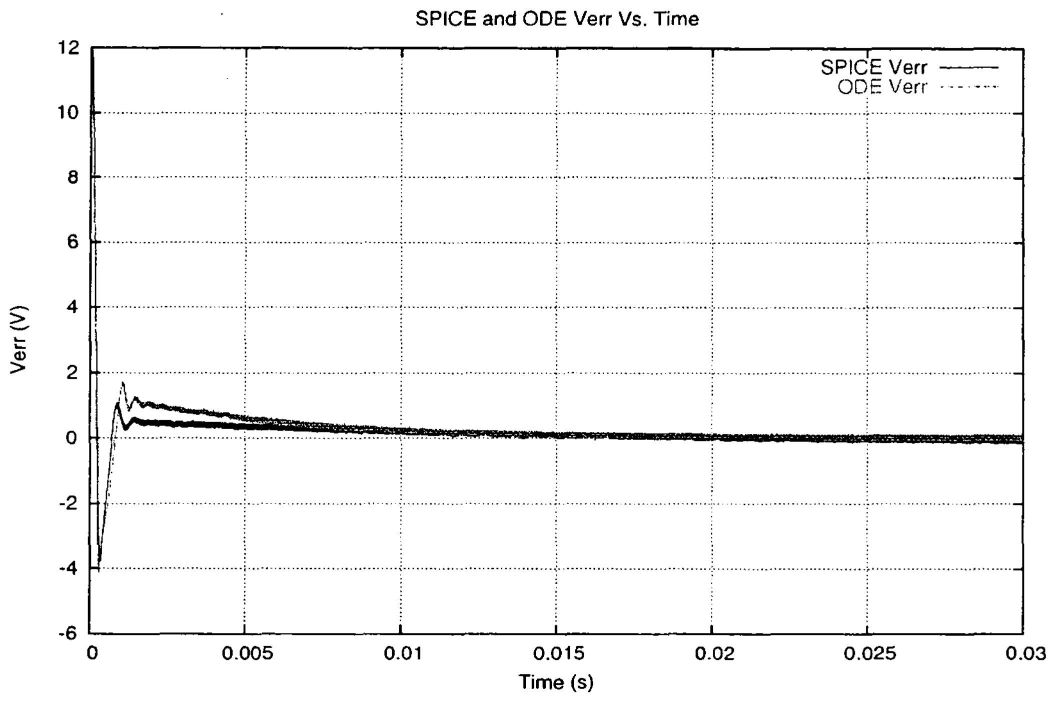


Figure 4.23: V_{err} comparison between SPICE and OCTAVE

Figure 4.23 shows the SPICE and OCTAVE simulations for V_{err} as shown on figure legend. Although they both show similar characteristics, the acceptable LTE for ODE in Octave is 5% but for SPICE, the LTE tolerance range is less than 1% and minimum time step is 20 ns. The LTE in SPICE can not be increased without experience convergence problems. Another major difference is the control circuit components, such as operational amplifiers and switches are different in SPICE and ODE simulation. The simulation codes for SPICE, MATLAB and OCTAVE are given in the Appendix B.

4.6 Simulation of the Boost Converter using TD-ENV

ODE and TD-ENV methods are based on the same nonlinear Equations (4.20 - 4.25). The TD-ENV method solves the set of nonlinear equations for the full switching period of T_2 along t_2 axis with N grid points.

On the proposed TD-ENV method, an adaptive time step is used on the t_1-axis and fixed grid on the t_2-axis. In this research, the Euler integration is used in t_1-axis and 3-point rule and/or BE method is used in the t_2-axis. The advantage of the TD-ENV is that it receives less computation time and storage for all the circuits simulated in this research (except for the PI controller circuit, as it will be seen). To avoid the convergence problem and high LTE during the transient time on PI circuits, the interpolated ODE results are used as initial condition in t_2 direction [4]. However from the simulation it is observed that the use of ODE results do not improve the computation time. This is because the duty cycle D_{con} in the PI controller changes each time point of t_1 direction, where step size is kept in the minimum. A small change in D_{con} produces a huge difference in the switching resistor and takes longer time for the simulation. For PI controller, D_{con} oscillates in a different frequency and includes an other widely separated time scale to the system. A minimum time step of $\frac{T_2}{N}$ is chosen for the complete simulation to reduce the higher LTE. If a time step smaller than $\frac{T_2}{N}$ is used, even smaller LTE is achieved. However for the smaller time step, TD-ENV method takes longer than the ODE method.

To improve simulation time and reduces LTE in P controller, small time steps are considered during the transient and large time steps are considered during the steady state.

The steps followed to obtain a better convergence rate and to reduce LTE are summarized as follows.

- An exponential behavior is considered during the fall-rise transition time in the switch function as explained in Section 4.3.

- Reduced the number of state variables in the circuit and increased the grid size in the t_2

axis.

- Relaxed the acceptable LTE during the simulation time.
- Used interpolated ODE results [4] for the first 2 cycles in t_2 direction to have better LTE and convergence rate.

Two cases are considered for the TD-ENV simulations, and they are:

1. Circuit with fixed load
2. Circuit with variable load

4.6.1 Circuit with Fixed Load

- **Application Of Constant Duty Cycle Control:** Here the converter circuit with fixed load is simulated using constant duty cycle D_{con} control of $\frac{7}{12}$. Where D_{con} is applied to the switch function of the converter circuit as shown in Figure 4.16. The fixed load is equal to 4 Ω, the inductance is 10 uH, the internal inductor resistance is 10 mΩ, internal capacitor resistance is 20 mΩ, and the capacitor is 200 uF. Table 4.3 shows the methods used and the computation times for constant duty cycle. On this table, the ODE method uses one time variable which is discretized using BE. Traditional TD-ENV uses two time variables which are discretized using BE. However this method doesn't use the complete proposed control. Traditional method only adjust the time step size using dichotomy, where dynamic tolerance are not adjusted as discussed in Chapter 3. TD-ENV methods with BE and 3-point are discretized using BE and center derivatives respectively, and the proposed adaptive time step control is used in the t_1 time direction. It is observed that the proposed method TD-ENV is almost four times faster than the ODE solution. Figures 4.24 - 4.27 show the simulation results of output, state variable representation,

Table 4.3: Converter circuit with constant duty cycle control, using 40 grid points in T_2

	method used	Rejected points	Accepted points	Total points (Time)
ODE	BE	N/A	N/A	10000+ (9452 s)
TD-ENV Traditional time-step controller	BE (LTE only)	108	364	472 (2641 s)
TD-ENV with BE Proposed time-step controller	BE	15	137	152 (683 s)
TD-ENV with 3-point Proposed time-step controller	Center derivatives	12	106	118 (557 s)

inductor current and time step variation. For simulation, $h_{max} = 500T_2$ and $h_{min} = \frac{T_2}{N}$ are used as step limitations.

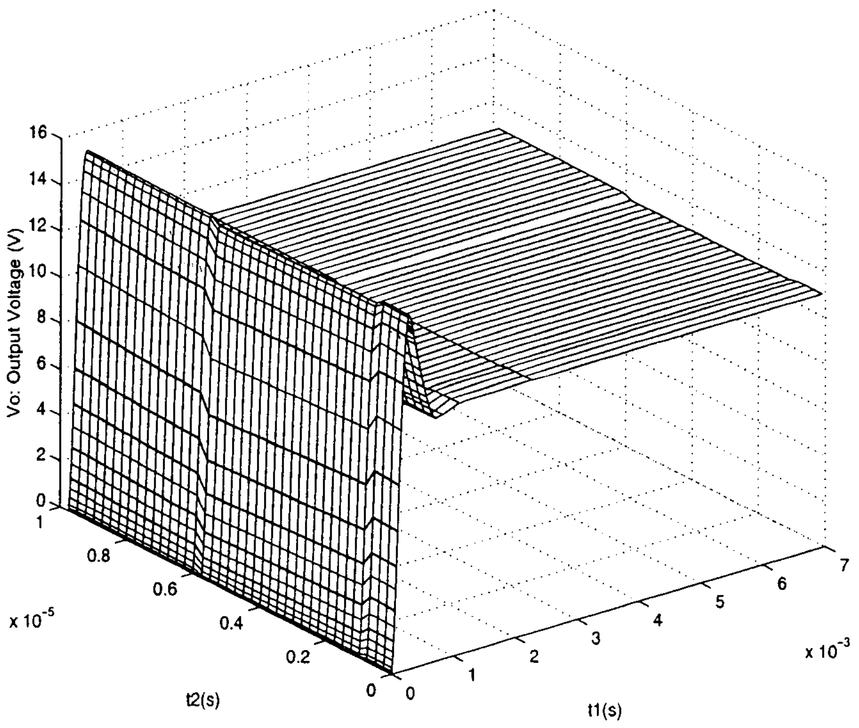


Figure 4.24: V_o vs. Time without Feedback Controller

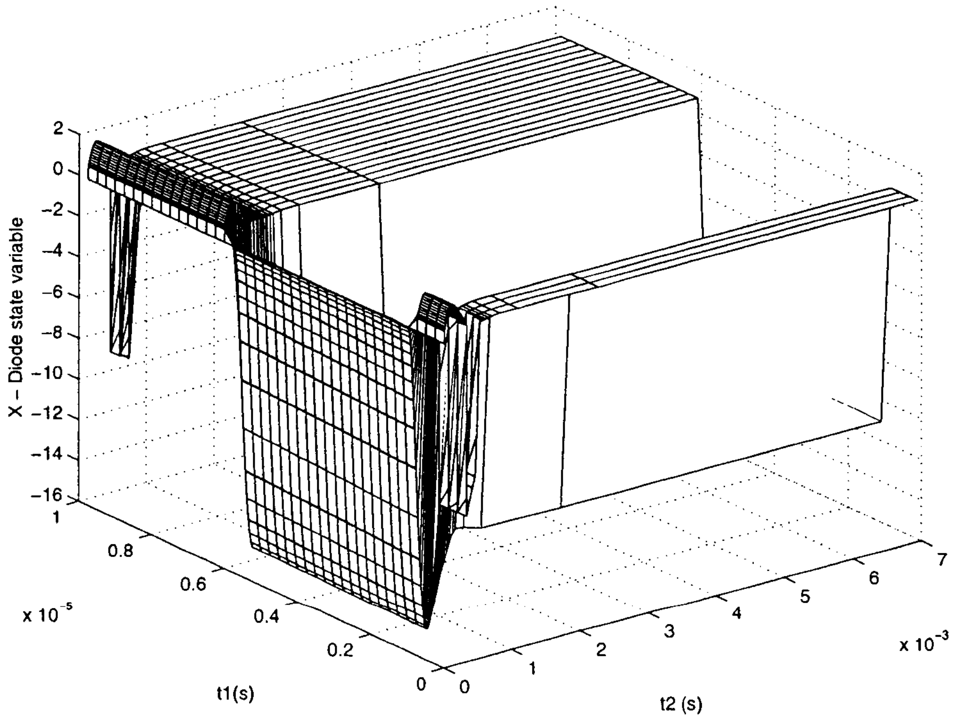


Figure 4.25: X vs. Time without Feedback Controller

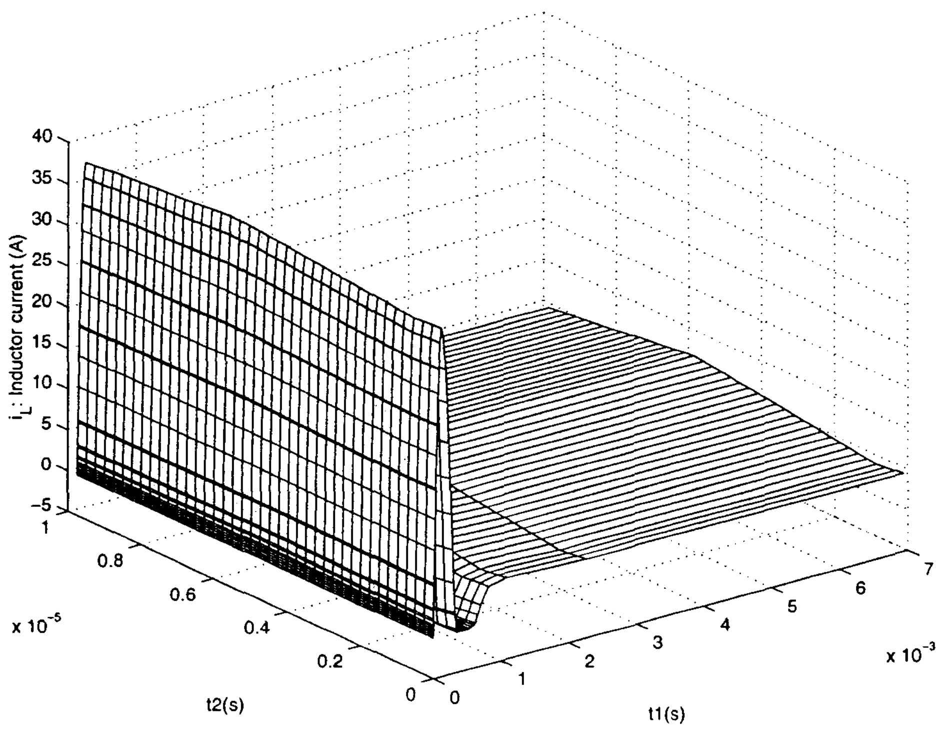


Figure 4.26: i_L vs. Time without Feedback Controller

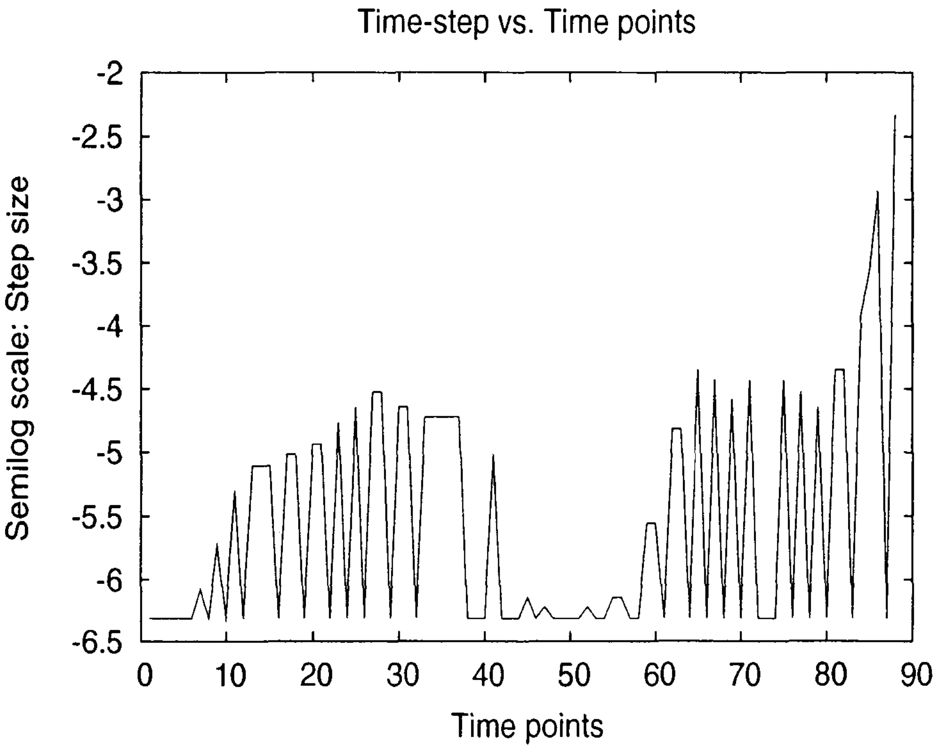


Figure 4.27: Time step size vs. Time points without Feedback

- **Application of P Controller**

The P controller is applied on the converter circuit. The maximum time step for this simulation is $500T_2$. When the simulation is performed for 30 ms, the initial transient was not visible. Therefore a semi-log scale is used to show the time step.

Figures 4.28 and 4.29 show the output voltage (V_o) and inductor current (i_L) respectively for OCTAVE code TD-ENV simulation using a P controller. It can be observed that the P controller produces a steady state error on Figure 4.28. Table 4.4 compares the computation time for this P controller circuit. Figure 4.31 shows the adaptive step size for initial transient. After 3 ms the step size is increased very much which can be seen at the end time points of the Figure 4.31.

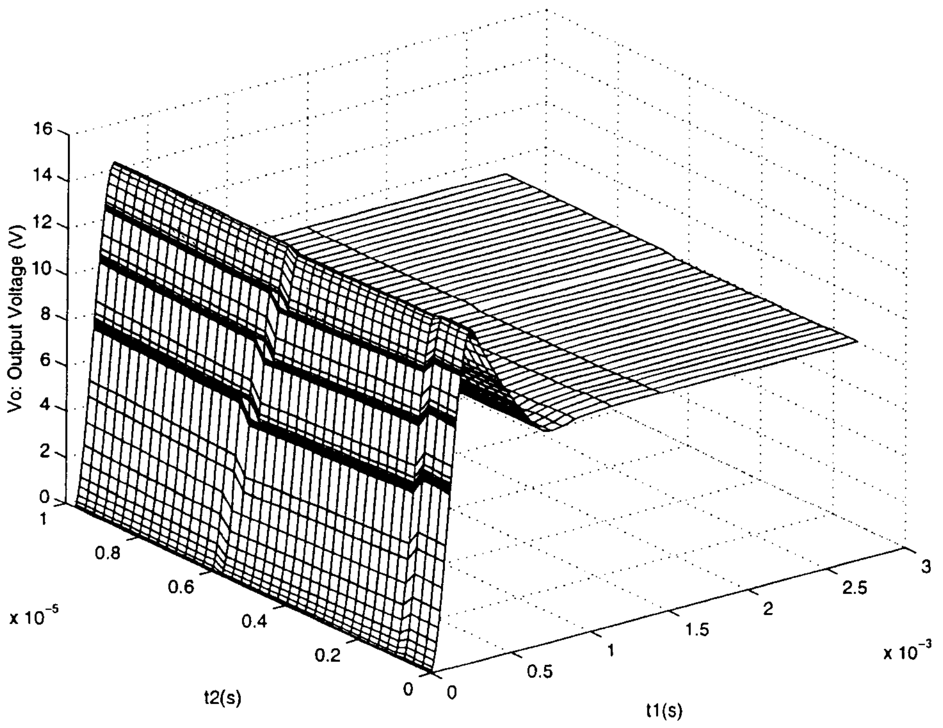


Figure 4.28: Transient response with P controller. V_o vs. Time

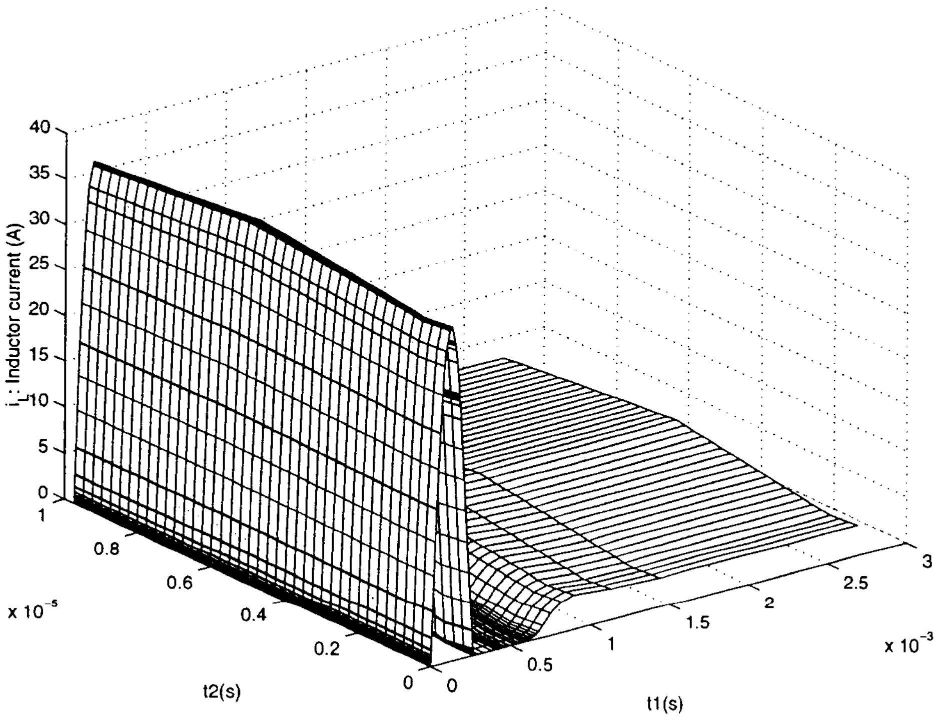


Figure 4.29: Transient response with P controller. i_L vs. Time

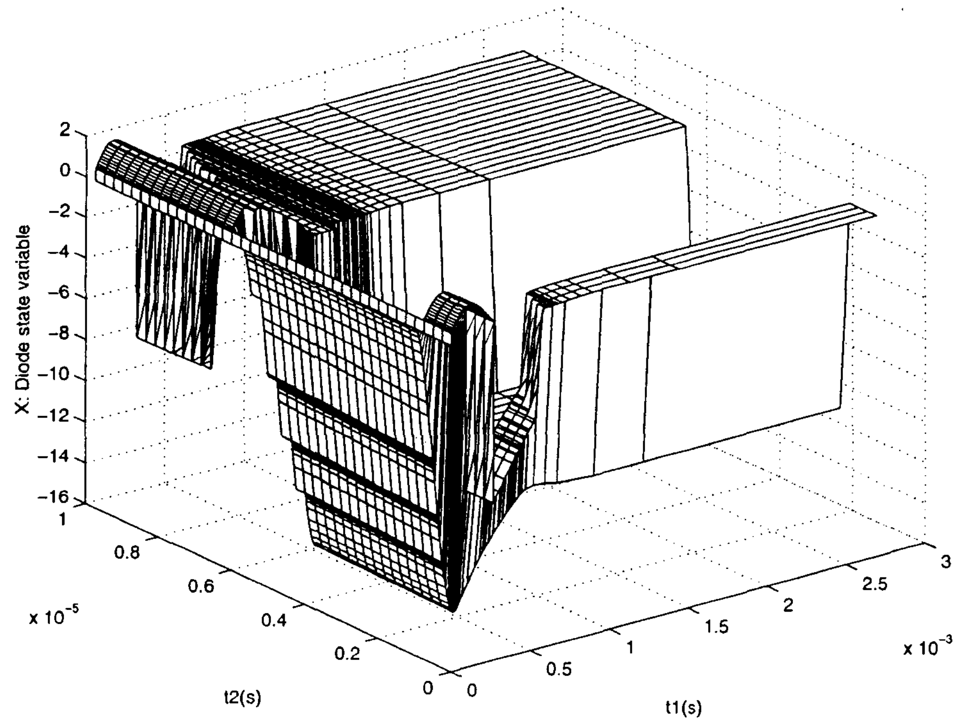


Figure 4.30: Transient response with P controller. State variable X vs. Time

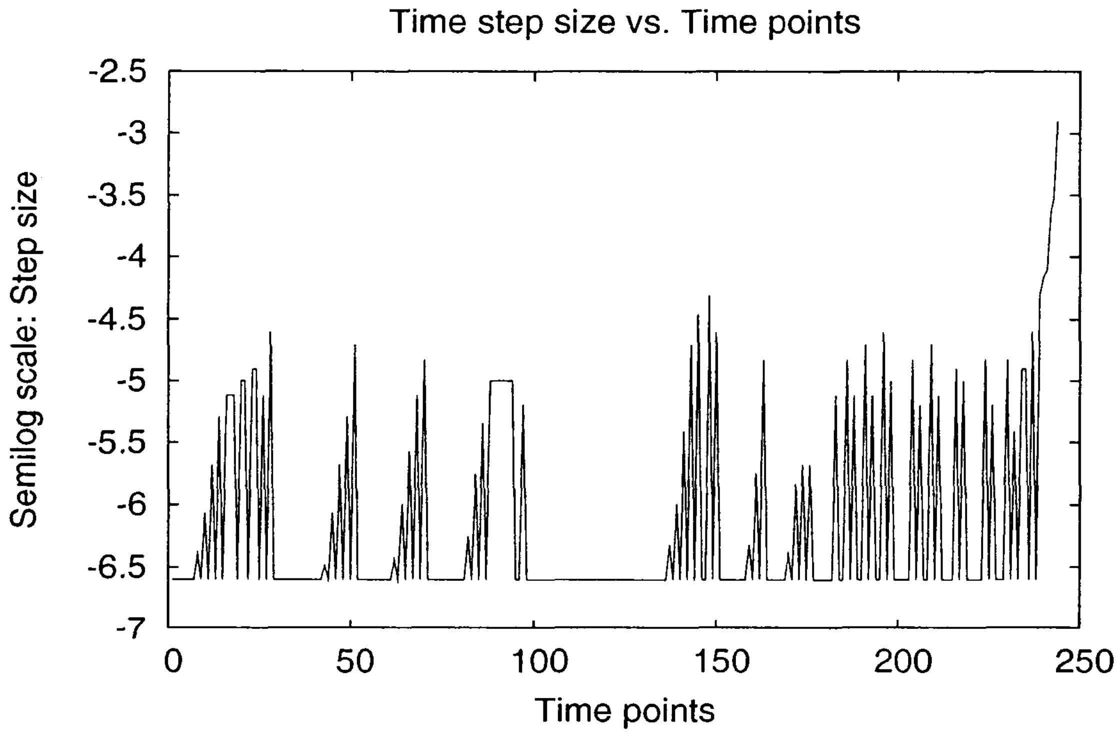


Figure 4.31: Transient response with P controller. Time step size vs. Time

- **Application PI Controller:**

A P controller alone cannot remove the steady state error. Therefore a PI controller is applied on the converter circuit to remove the steady state error of V_o. The time step on Figure 4.32 is kept at a minimum value of $\frac{T_2}{N}$ most of the time during the simulation, which leads to a long simulation time. If the time step continues to behave with this minimum time step, the PI computation time gets as high as $N^3 \times$ ODE simulation time. At every time point along t1 direction, proposed model will simulate $N\text{X}N$ grid points of FDTD problem. For the worst scenario lets consider the minimum step size all along the t1 axis, then simulation will repeat $N^2\text{X}N$ more computations than ODE simulation time.

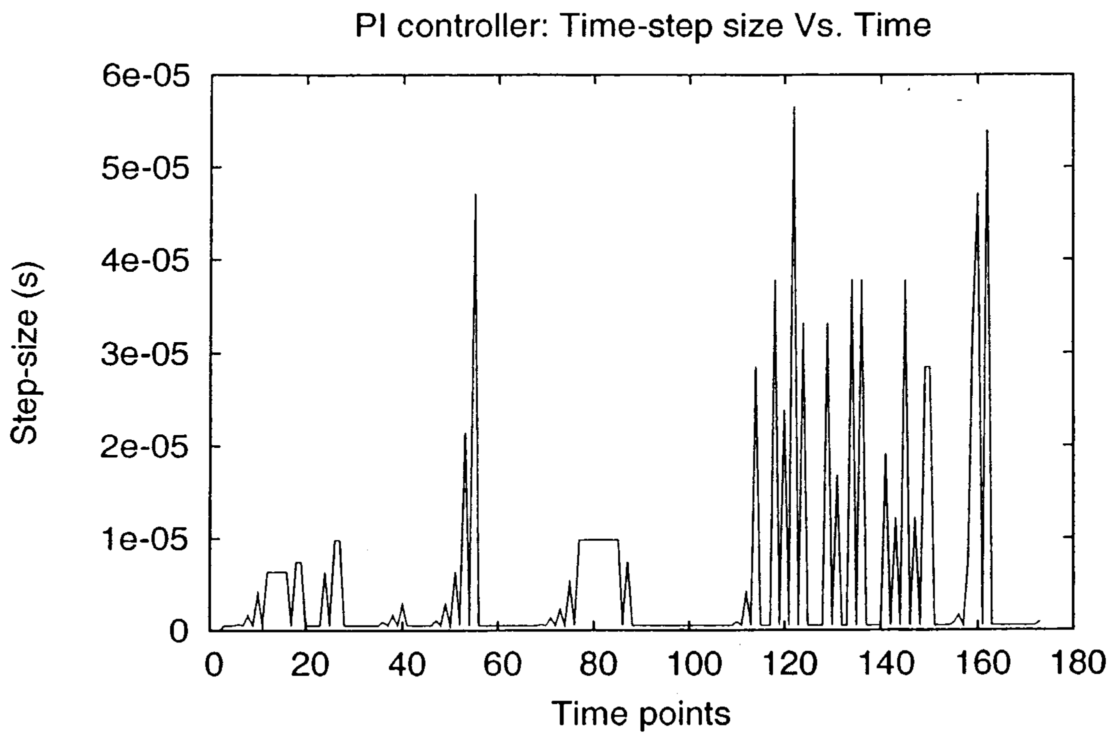


Figure 4.32: Time step size vs. Time with PI controller

Figure 4.33 shows the PI controller duty cycle variation. Zoomed Figure 4.33 shows an other exciting frequency in the system. There a sinusoidal variation in duty cycle control can be observed. The sinusoidal variation in D_{con} creates a huge difference in switching

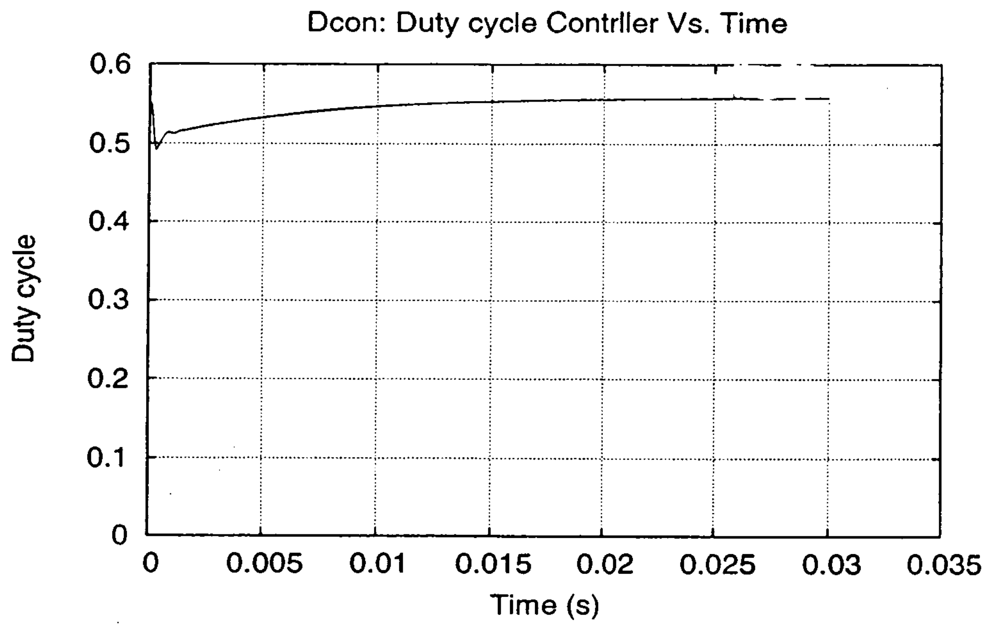


Figure 4.33: D_{con} oscillation for PI

Table 4.4: PI controller computation times for various methods

	Integration method used	P time (s)	PI time (s)
SPICE	Gear	850	1400
Matlab	Variable time step ODE15S	1550	2110
ODE	BE	30942.11	33701.99
Traditional TD-ENV	BE	11962.11	-
Proposed TD-ENV	BE with center derivatives	3612.81	-

resistor R_s for adjacent time points in the t_1 scale. This produces a large LTE and keeps the time step at minimum size.

Table 4.4 compares the computation times for the simulations with P and PI controllers. The total simulation time is 30 ms. A LTE of 5% is used for all simulations. For SPICE, a time step of 20 ns is used as a minimum time step. SPICE couldn't finish the computation for time steps greater than 20 ns. From the results in Table 4.4, it follows that the proposed model gives better simulation times than the other methods.

4.6.2 Circuit with Variable Load

- **Application of P controller:** The next set of simulations is made for a variable load with a P controller. The time dependance of the variable load is shown in Figure (4.34). The load resistor is given by

$$R_L = 4 + 2sin(2\pi\tau_1 t_1), \tag{4.30}$$

where τ_1 is 5e-3.

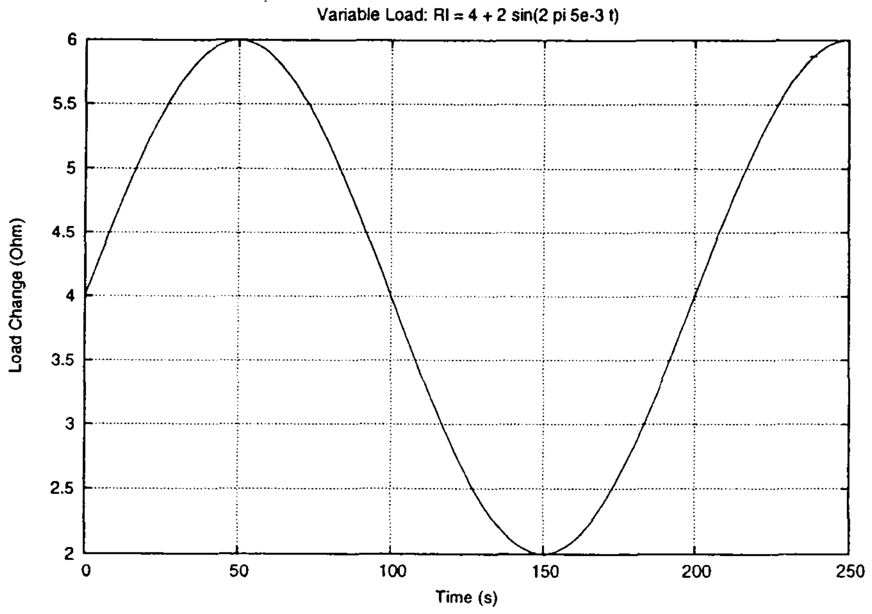


Figure 4.34: Variable Load

Figures 4.35, 4.36, 4.37 and 4.38 show the transient simulation results for output voltage, inductor current, diode state variable and time-step size respectively. The continuation of this simulation is shown for full simulation time in Figures 4.39-4.42. Figure 4.42 shows the adaptive step size on $y - axis$, where maximum allowed step size is 20 s. The front moving problem in the diode state variable and the discontinuous mode can be seen during this time on Figure 4.37. During this discontinuous mode time, the switching resistor (R_s) is plotted and made sure that the R_s doesn't have any change other than t_{on} and t_{off} transitions during T_2 time period.

Table 4.5: Total computation time for the boost converter with P controller, variable load and 40 grid points

	Integration method (Tol: 5%)	Accepted points	Rejected points	Total simulation for 250 s points (time s)
Traditional	BE	541	374	915 points (54,130 s)
Proposed	Euler with 3 point rule in T_2	236	13	249 points (11,660 s)

Table (4.5) shows a clear advantage for the proposed method. The CPU time for a time-marching simulation would be too long to be practical. The proposed model is almost six times faster than the traditional method. ODE implementation is not feasible for the above table comparison, because it will take a long time and tons of points to complete the simulation. The boost converter with P controller settle around 9.5 V, and it has a constant steady state error. However in PI controller the output Voltage settles exactly at 12 V, with small ripple due to switching and output capacitor r_{SC} and C. The integral controller removes the steady state error as discussed in Section (4.3). The proposed method gives better results for a constant duty cycle system, P controller system and variable load with P controller system. Figure 4.42 shows the full scale of time step control for variable load using P controller. During the initial transient time, step size is less than 6×10^{-6} as shown in Figure 4.38. After that time step size is increased gradually up to 20 s as shown in Figure 4.42. More grid points imply more points during the fall-rise edge on every switching period during our simulation. The fall-rise time on every switching transition has sharp changes and this produces convergence problems during the simulation.

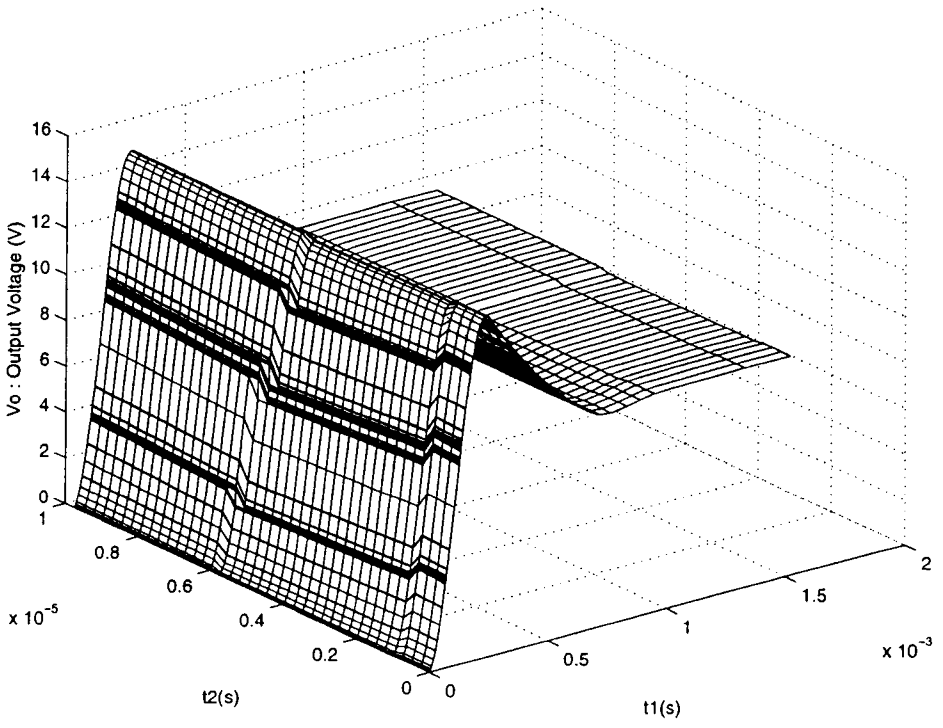


Figure 4.35: Transient response of circuit with variable load using a P controller. V_o vs. Time

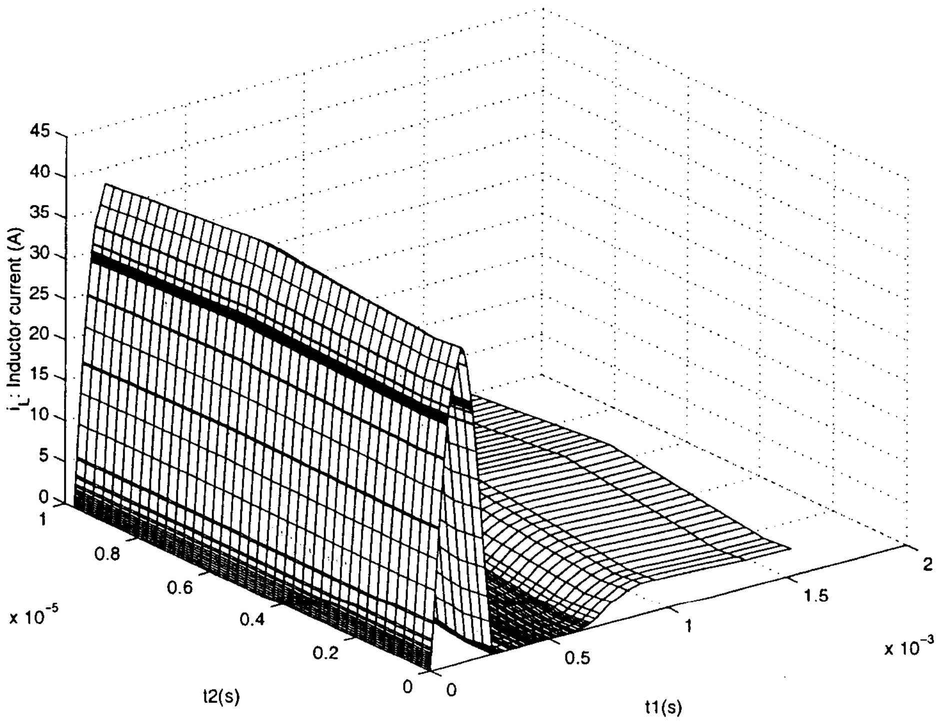


Figure 4.36: Transient response of circuit with variable load using a P controller. i_L vs. Time

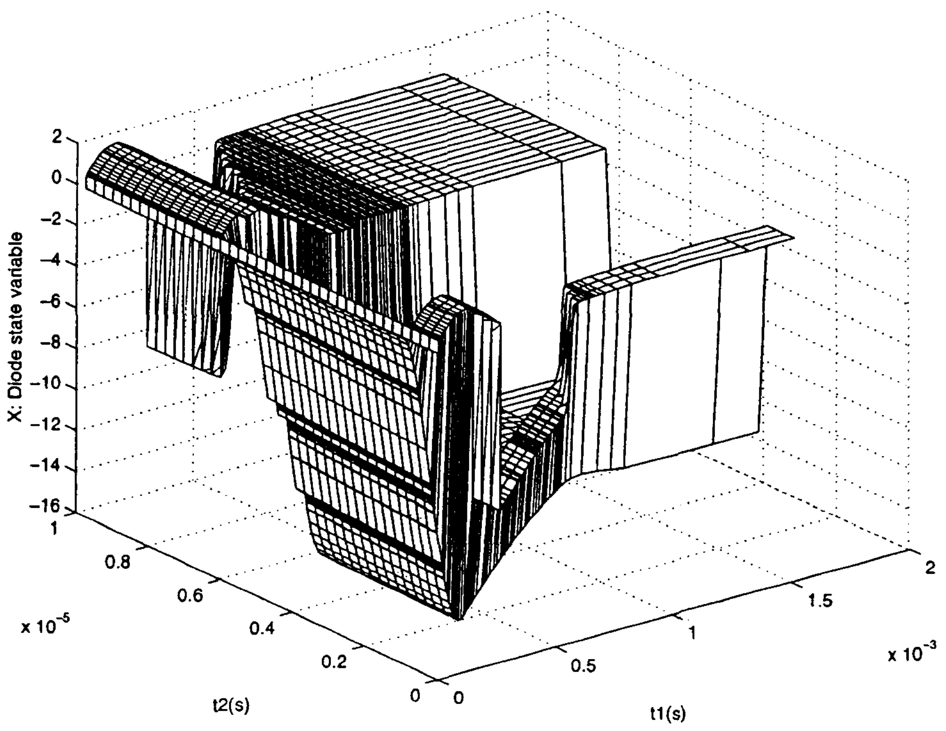


Figure 4.37: Transient response of circuit with variable load using P controller. State variable X vs. Time

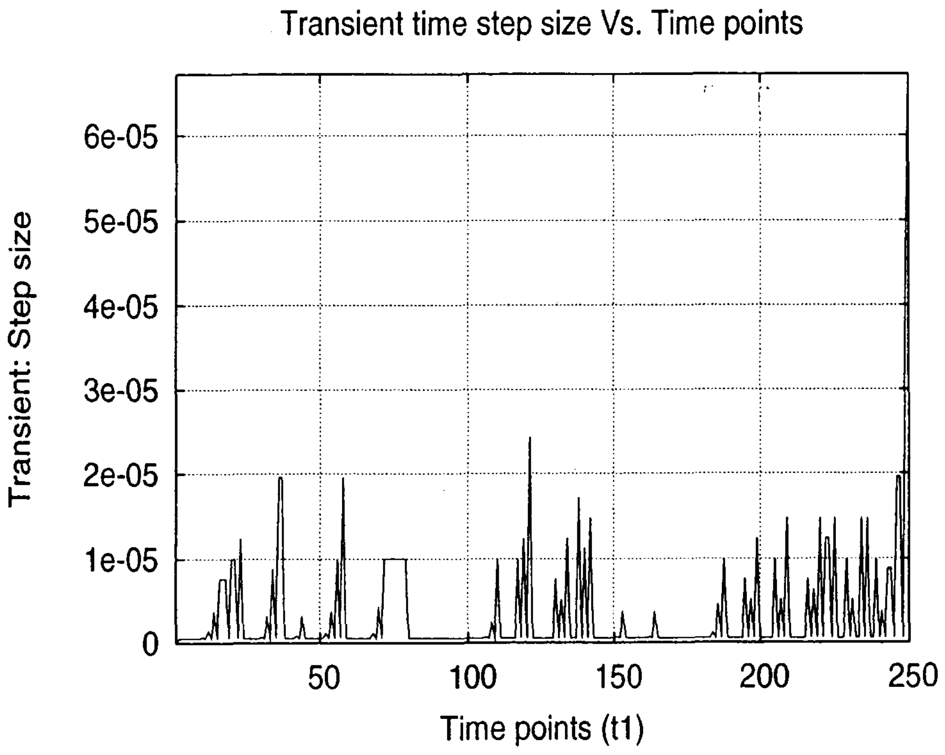


Figure 4.38: Transient response of circuit with variable load using P controller. Step size vs. Time points

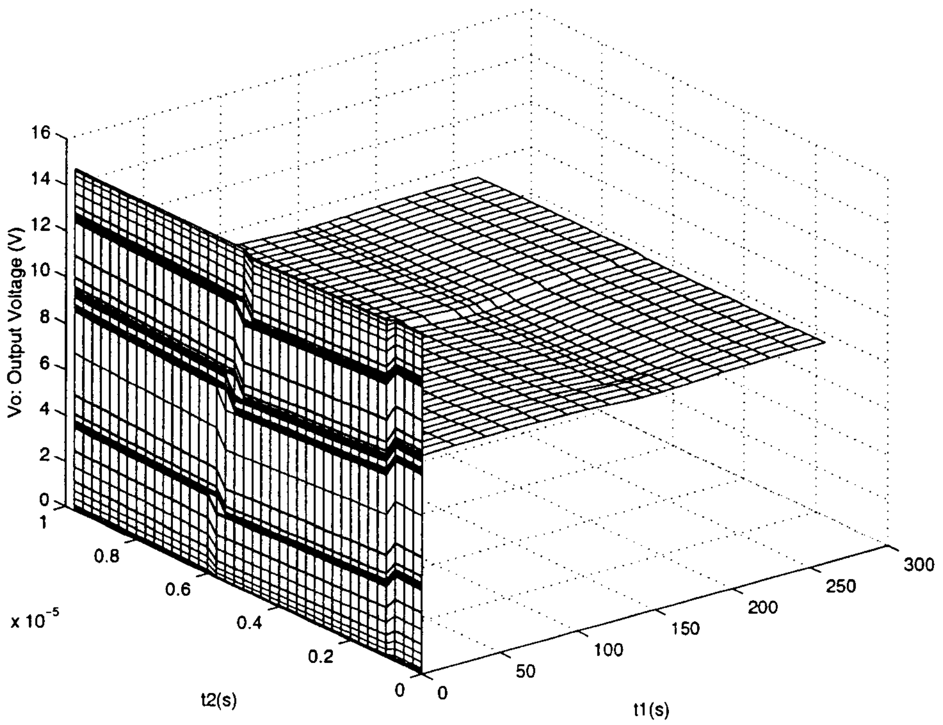


Figure 4.39: Response of transient and steady state for the circuit with variable load using P controller. V_o vs. Time

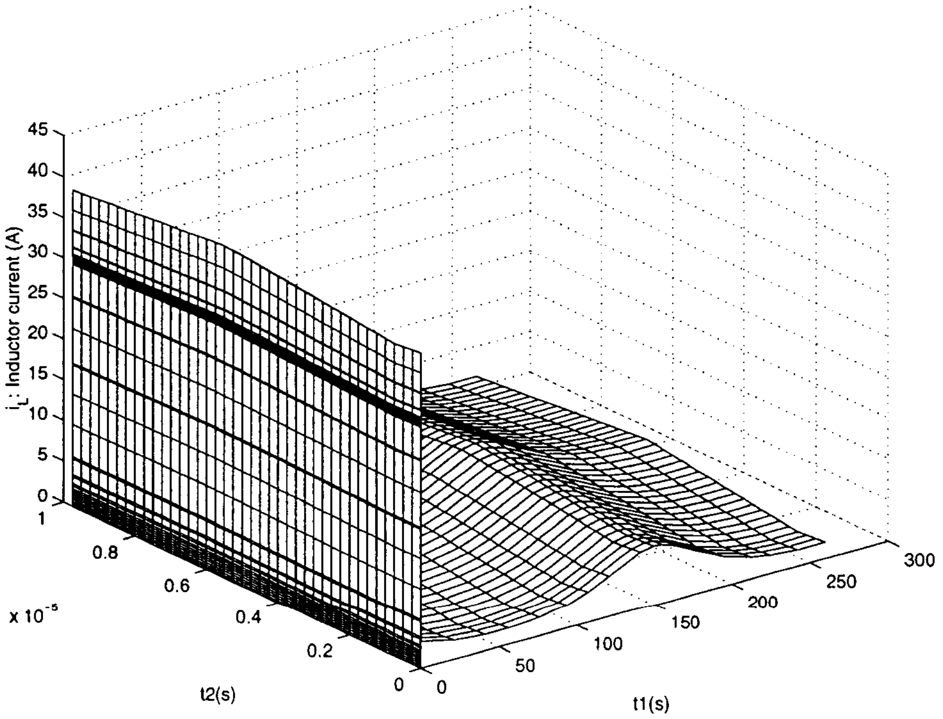


Figure 4.40: Response of transient and steady state for the circuit with variable load using P controller. i_L vs. Time

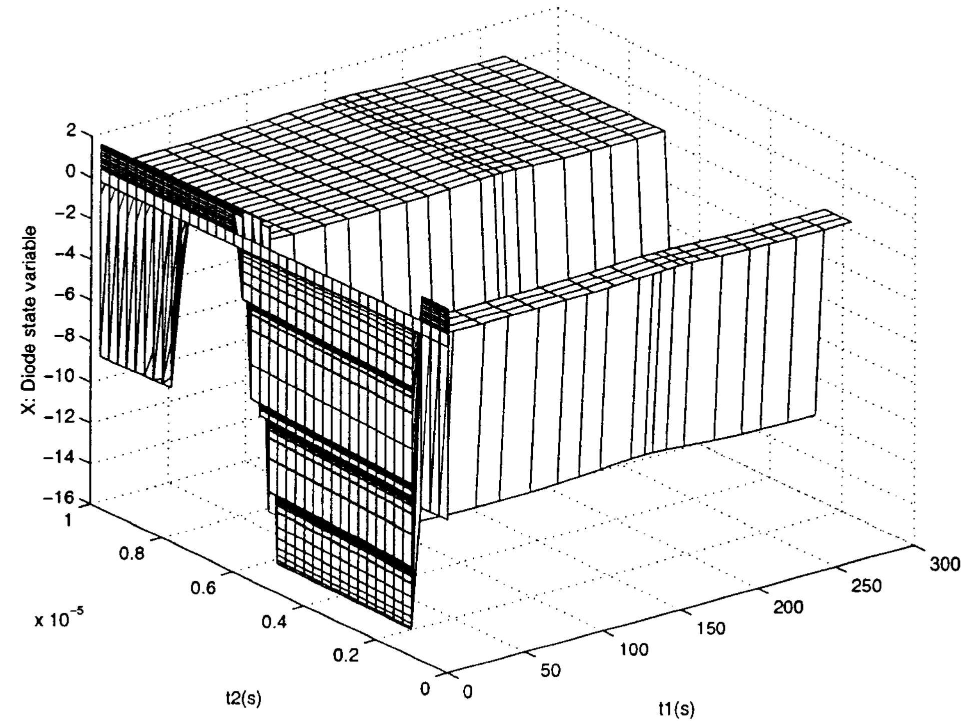


Figure 4.41: Response of transient and steady state for the circuit with variable load using P controller. State variable X vs. Time

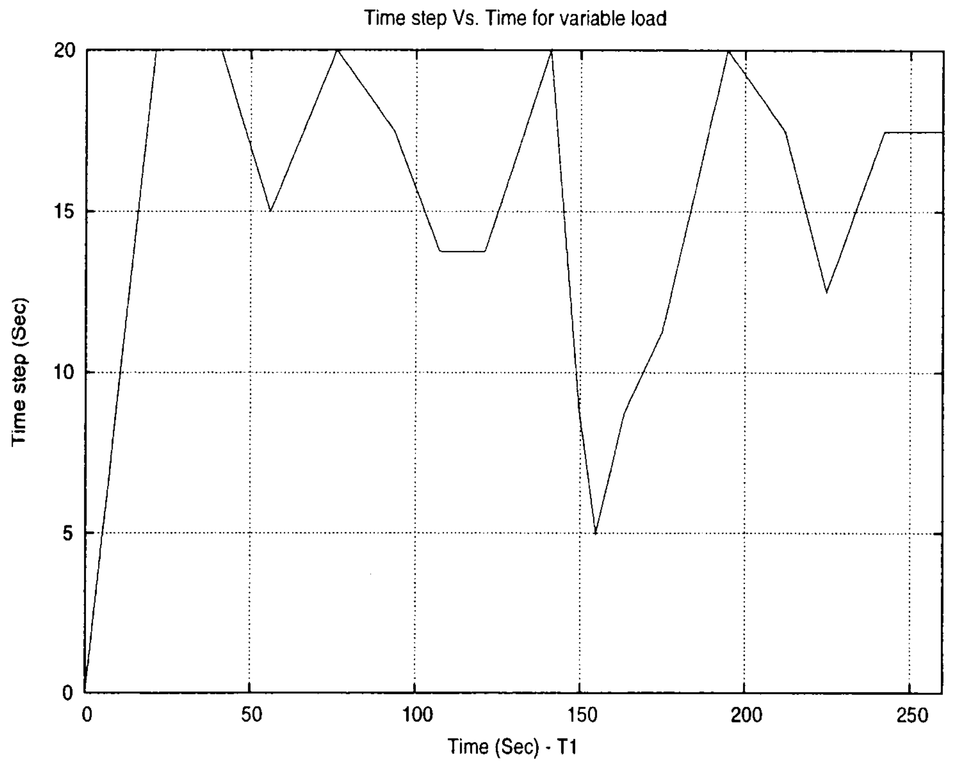


Figure 4.42: Response of transient and steady state for the circuit with variable load using P controller. Time step vs. Time

4.7 Comparison of initial conditions in Time-Marching and TD-ENV methods

In TD-ENV simulation, the first few switching results are way off from the ODE simulation results. This is because on TD-ENV simulation, a column of zeros is used as initial condition for the first two periods in the t_2 direction. This gives improper extrapolated results at the beginning of the simulation. To solve this problem, TD-ENV method uses the first 2 periods of the ODE result and interpolates them to get the initial condition for TD-ENV simulation where the rest of the parameters and initial conditions of ODE and TD-ENV are kept the same. This improved the LTE during transient response.

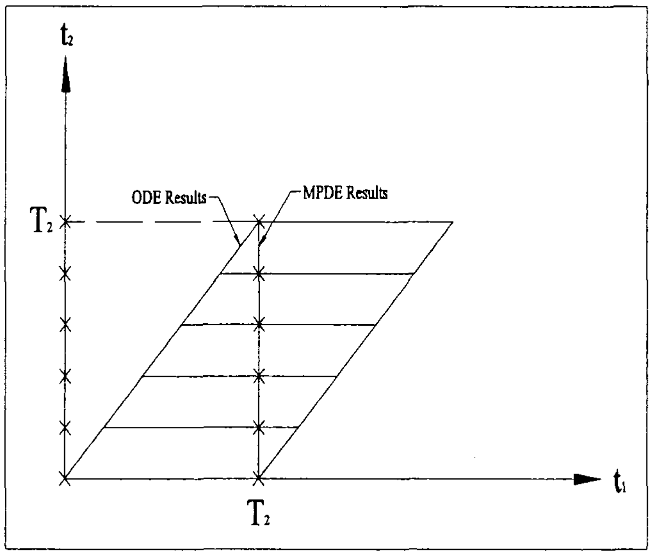


Figure 4.43: Interpolation method to determine initial conditions for TD-ENV

The comparison between the results of time-marching and TD-ENV are mapped to one time dimension, as discussed in Section 4.1. On the edge of switching transition there are some glitches due to the 'switching' and linear interpolation.

Figure 4.43 shows the interpolation method used to formulate the initial conditions for TD-ENV simulation. If the the time step is assumed to be same in t_1 direction as in switching period T_2, then the elements in the vertical line at $t_1 = T_2$ can be found by linear interpolation.

4.8 Conclusions

The bivariate form using TD-ENV and MFDTD can require far fewer points to represent a signal than a set of samples, yet it contains all the information needed to recover the original signal completely [1].

The MFDTD and TD-ENV methods are used to simulate a rectifier circuit and a converter circuit respectively. It is shown how the TD-ENV method is used for transient simulation in a DC-DC converter circuit and the MFDTD method is used for steady state simulation in a rectifier circuit.

Section 4.5 shows the ODE output results from SPICE, MATLAB and OCTAVE. They agree within some tolerance. However, they are not superimposed on each other, because the models are not exactly equal and the tolerances are different. Section 4.5 explains the TD-ENV method using an adaptive time step, which gives better computation time and storage than all the other traditional methods discussed in this Chapter for transient simulation. However, the PI controller circuit experiences a higher LTE and convergence problem due to fast variations and oscillations in the duty cycle D_{con} as shown in Figure 4.33. The duty cycle control has an oscillation in Figure 4.33 is continuing throughout the simulation, that makes the time-step to be minimum all the time. As a result we don't have good computation time during the PI simulation.

Chapter 5

Summary and Future Work

5.1 Summary

The TD-ENV method is used with an adaptive time step algorithm to simulate a truly nonlinear circuit in time domain. Possibly one of the most important contributions of this work is the time step control algorithm for stiff circuits.

The TD-ENV and MFDTD methods solve transient and steady state respectively of a circuit particularly in the case where signals at two very different frequencies are used. However, it is impossible to declare any method superior. For a particular problem, one method can give a better results than the other and in another example same method can perform poorly.

The main drawback of the MFDTD method is the large size of the matrix. To increase the accuracy of the simulation, the grid size can be increased. As grid size increases, matrix size and computation time also increases. Sparse matrix properties can be used to remove some computation time. Therefore, the details of matrix solving are crucial. Sparse matrix solving algorithms should be used in MFDTD method.

The main drawback of the TD-ENV method is that it requires a small step size during the beginning of transient. Each time point in the true time axis T_1, the nonlinear equations are solved using Newton iterations. The larger the circuit size and number of unknown variables, the larger the number of nonlinear equations for the system. For example, for a circuit with k unknown variables and N grid points at each T_2 switching cycle, the number of nonlinear equations that has to be solved in the Newton iteration is $k \times N$.

The TD-ENV method has $O(kN)^3$ relationship between the grid size and the computation time, when dense matrices are considered during simulation. Nevertheless, the TD-ENV method gives faster computation than ODE method for a system with widely separated time constant values in it.

Reference [4] shows the importance of the initial condition during a transient simulation. An interpolated ODE results are used as initial condition for TD-ENV method, as discussed in Chapter 4. When the ODE results are applied as initial condition in the TD-ENV method, the TD-ENV method gives a faster computation time than a column of zeros as initial conditions for all the variables.

5.2 Conclusion

Stiff circuits have extreme range of operating frequencies or time scales, which are difficult to simulate. The MPDE formulation gives faster computation when there are widely separated time constants. MPDE formulation is the solution to the problem of the transient analysis of circuits with widely different time constants (example: thermal-electrical). On this thesis the system of ODE's that describe a circuit is converted into a system of PDE's using multiple time variables, and solved through difference equations in MFDTD and TD-ENV.

The rectifier circuit gives a promising computational time for adaptive time step control algorithm. The traditional method is twice as slow than the proposed adaptive time step model, as shown in Table 4.1. The proposed TD-ENV approach is also faster for the DC-DC converter circuits, except the circuit with PI controller because of the duty cycle oscillations. Another problem in this circuit is the sharp transition on the switching. Even though an exponential behavior is used on the switch function, the sharp transition still exists and gives convergence problems. During this transition time, the LTE is very high, and to compensate this error the adaptive time step algorithm takes the smallest time step, which increases the simulation time. Due to the smallest time step and D_{con} oscillation as discussed in Section 4.6, computation time is too long for a PI controller system.

The proposed model has to be modified to handle a PI controller, because it gives higher LTE on the transition edge of the switch, even with a small time step. The LTE is computed using the difference between the extrapolated value and the nonlinear solution, as discussed in Section 2.5. To find an accurate LTE for the proposed method, more research can be done in future.

The D_{con} oscillation shown in Figure 4.33 produces a moving front, mainly in the diode state variable during the simulation. This causes a large value in the estimated LTE that reduces the time step. Due to the presence of the PI controller, D_{con} oscillates close to the switching frequency and the time step is kept as the minimum value during the simulation time, which is shown in Figure 4.32.

5.3 Limitations on MFDTD and TD-ENV methods

- **MFDTD method:** Computation time is high, however this is a popular method to find steady state solution for Electro Magnetic circuits. In this research, the MFDTD method is tested by choosing various grid size for more accuracy. The higher the grid sizes, the higher the accuracy. Higher accuracy will superimpose ODE and 'MPDE to ODE' comparison plots on each other as shown in Figure 4.5, however it will take long computation time. Also, this method can only be used for periodic and quasi-periodic signals to find the steady state solution and not for transient solution.

- **TD-ENV method:** The system must have a periodic excitation to analyze TD-ENV method. Furthermore, the system must have widely separated time scales on it.

5.4 Future Work

- Most of the topics treated in this research are left open to new research. One possible topic for future work is to use the adaptive grid in t_2 axis for faster computation in TD-ENV method. This would allow the simulation to have less points on t_2 axis, which will reduce the computation time by a large factor. One possible method to achieve adaptive griding is called Adaptive Base Functions (ABF) given in [8].
- Another future work is the implementation of sparse matrix techniques during our simulation [39]. Sparse matrices are a basic tool of computational science and engineering. Sparse matrices are a special class of matrices that contain a significant number of zero-valued elements. In this research, the Jacobian matrix in Newton iterations is a sparse matrix, but it is treated as a dense matrix by the simulation code. This sparse matrix property can be used to speed up the simulation.
- The PI controller DC-DC converter circuit experiences more convergence issues and high LTE. Two approaches can be considered to overcome these problems. The first is to use a different methods during each segment of the simulation time and the second is to use three time scales. Due to the limitation of time, these are not executed in this thesis. First approach is to formulate the circuit equations to improve convergence rate and reduce LTE by using 'ON' state model and 'OFF' state model switch equations. This will remove the transition edge on the switch function and may give better simulation results. Also, different methods can be used during the simulation time, such as ODE for the initial transient and then TD-ENV for the rest of the simulation time. From the simulation results it is clearly observed that there are convergence and LTE issues, during the start time to settling time. However, after settling time, there was no changes on the circuit to

use TD-ENV with large time steps. The computation time can be improved also by using three time scales, one for switching (t_2), one of PI controller oscillation (t_3) and one for the real time (t_1). These techniques will improve the adaptive time step algorithm and give better computation time even on PI controller system.

Appendix A

Time Domain methods: Time-step

A.1 FDTD in EM circuits

The FDTD algorithm provides a means to numerically solve Maxwell's equations in the time domain [38]. This technique is mostly used in Electro Megnetic (EM) circuit simulation. FDTD calculates the E and H fields within a gridded computational domain using grids that are small compared to the smallest wavelength and model feature. Therefore, far fields and some models with with extended features such as wires may not be applicable due to large domains with excessive computational times.

A.2 Wavelets methods

Wavelets have dominant applications in signal processing and image processing, such as smoothing, recognition of features and compression. Wavelets offer a means of approximating functions that allows selective grid refinement [33].

A.3 Numerical Solution of Differential Equation

A differential equation (DE) is an equation involving an unknown function and its derivatives. It can have one or more variables. A DE can be either linear or nonlinear. Generally a linear equation is linear with respect to the dependent variable (y) and its derivatives. Def. of a linear DE is: if (a) every dependent variable (y) and every derivative involved, $d^n y/dx^n$, occur to the first degree only, and (b) no products of dependent variables, and/or derivatives, and/or nonlinear function occur [45].

Example of Non-linear partial differential equation	Example of Linear partial differential equation
$\frac{\partial^2 y}{\partial x^2} + 5\left(\frac{\partial y}{\partial x}\right)^3 + 6y = 0$	$6\,\frac{d^3 x}{dt^3} + 2\,\frac{dx}{dt} + x = 6\,t$
$\left(\frac{\partial y}{\partial t}\right)^2 = \frac{\partial^2 y}{\partial x^2}$	$\frac{\partial y}{\partial t} = \frac{\partial^2 y}{\partial x^2}$
$\frac{dy}{dt} = log(y)$	$\frac{d^2 y}{dx^2} + y = 0$

An Ordinary differential equation, ODE, has the form of $y' = f(t, y)$, where $\partial f/\partial y$ is a non-singular matrix. ODEs are encountered when dealing with initial value problems. The order of the differential equation is the order of the highest derivative of the unknown function involved in the equation. An ODE is a relation which involves one or several derivatives of an unspecified function y of x. The following equation: $i_d = V_o/R + C\,dV_o/dt$, is a first order differential equation. Usually, in circuit simulation, the order of a differential equation will be dependent on the number of capacitors and inductors used in the circuit. Partial differential equations arise in connection with two or more independent variables in an unknown function.

A linear differential equation of order n is a differential equation and can be written in the following form:

$$a_n \frac{d^n y}{dx^n} + a_{n-1} \frac{d^{n-1} y}{dx^{n-1}} + ... + a_1 \frac{dy}{dx} + a_0 y = f(x)$$

where a_n is not a zero function

Physical systems have special meaning for order of differentials. For example in displacement function, we can get velocity from the first order of the differential and the second order of differential will result the acceleration. There are many methods to find the solution for linear differential equation. A corresponding system of n nonlinear equations in n variables does not have any such theory and it may have any number of solutions.

A first order homogeneous differential equation involves only the first derivative of a function and the function itself, with constants only as multipliers. The equation is of the form:

$$\frac{df(x)}{dx} = af(x), \text{where } a \text{ is a constant} \tag{A.1}$$

Solution for this differential equation is $f(x) = A_0 e^{ax}$. The general solution to a differential equation must satisfy both the homogeneous and non-homogeneous equations.

A.4 Runge-Kutta method

[42] A 4th order Runge-Kutta method for time stepping gives disastrous results for $h = 0.1$, but gives accurate approximation for when $h = 0.05$ as evidenced in Table A.1. On the above

Table A.1: Stiff Solution using Runge-Kutta-Forth-Order Method

Time	W_1(t)	W_1(t)	U_1(t)	W_2(t)	W_2(t)	U_2(t)
(t)	h=0.05	h=0.1	Unique sol^n	h=0.05	h=0.1	Unique sol^n
0.1	1.712219	-2.645169	1.793061	-0.8703152	7.844527	-1.032001
0.2	1.414070	-18.45158	1.423901	-0.8550148	38.87631	-0.874681
0.3	1.130523	-87.47221	1.131575	-0.7228910	176.4828	-0.724998
0.4	0.909276	-934.0722	0.909409	-0.6079475	789.3540	-0.608214
0.5	0.738751	-1760.016	0.738788	-0.5155810	3520.999	-0.515658

Table A.1 W_1 and W_2 represent the solution of U_1(t) and U_2(t) respectively using different time steps. For example the first W_1 column represents the solution of U_1(t) using the step size of 0.05.

A.5 Optimum time step

[42] Decreasing time step and approximation error doesn't have a proportional relationship. Decreasing time step will not always make approximation error small. It has an optimum limit. Differential equations are written in the change form of some variable with respect to another. Most of these problems require the solution to differential equation that satisfies a given initial condition. Generally small time step will results better results, but that is not always true. There is an optimum value $h = \sqrt{2\delta/M}$, will give us good results and if we go beyond this step, we will not get better results. This section is discribing about optimum time step using Lipschitz theorms.

definition Lipschitz constant

A function $f(t, y)$ is said to satisfy a Lipschitz condition in the variable y on a set $D \subset R^2$, provided a constant $L > 0$ exists with the property that:

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|$$

whenever $(t, y_1), (t, y_2) \in D$. L is called Lipschitz constant for f.

Theorem

Suppose that $D = (t, y)| \quad a \leq t \leq b, \quad -\infty < y < \infty$, and that $f(t, y)$ is continuous on D. If f satisfies a Lipschitz condition on D in the variable y, then the initial-value problem:

$$y = f(t, y), \qquad a \leq t \leq b, \qquad y(a) = \alpha$$

has a unique solution y(t) for $a \leq t \leq b$.

Theorem

Suppose f is continuous and satisfies a Lipschitz condition with constant L on

$$D = (t, y)| \quad a \leq t \leq b, \quad -\infty < y < \infty,$$

and that a constant M exists with the property that

$$\ddot{y}(t) \leq M, \quad \forall \in [a, b]$$

Let $y(t)$ denote the unique solution to the initial-value problem:

$$y = f(t, y), \qquad a \leq t \leq b, \qquad y(a) = \alpha$$

and $u_0, u_1, u_2, \cdots, u_n$ be the approximations obtained using Euler's method for some positive integer N:

$$\begin{aligned} u_0 &= \alpha + \delta_0 \\ u_{i+1} &= u_i + hf(t_i, u_i) + \delta_{i+1}, \qquad \forall i = 0, 1, \cdots, N-1 \end{aligned}$$

Error associated with backward Euler method denotes δ_i, the round off error associated with u_i. Using above both theorem and Lipschitz conditions:

$$|y(t_i) - u_i| \leq \frac{hM}{2L}[e^{L(t_i-a)} - 1], \tag{A.2}$$

for each $i = 0, 1, 2, \cdots, N$.

Generally, we consider that small time-step will results better results. However, if we consider equation A.2 the error bound is nolonger linear in h and, in fact, since

$$lim_{h\to 0}\left(\frac{hM}{2} + \frac{\delta}{h}\right) = \infty,$$

the error would be expected to become large for sufficiently small values of h. The main problem is we don't have the values for M and L. We can only get the relation in a certain time range. Letting

$$E(h) = hM/2 + \delta/h$$

with h suffiently small and using L'Hospital's Rule will results

$$E'(h) = M/2 + \delta/h^2$$

If $h > \sqrt{2\delta/M}$, then $E'(h) < 0$ and E(h) is decreasing. If $h < \sqrt{2\delta/M}$, then $E'(h) > 0$ and E(h) is increasing. The minimum value of E(h) occurs when $h = \sqrt{2\delta/M}$. If we decrease h beyond this value, then the total error in will increase. Normally the value of δ is sufficiently small that this lower bound for h does not affect the operation of Euler's method.

A.6 Matlab DC-DC converter PI results

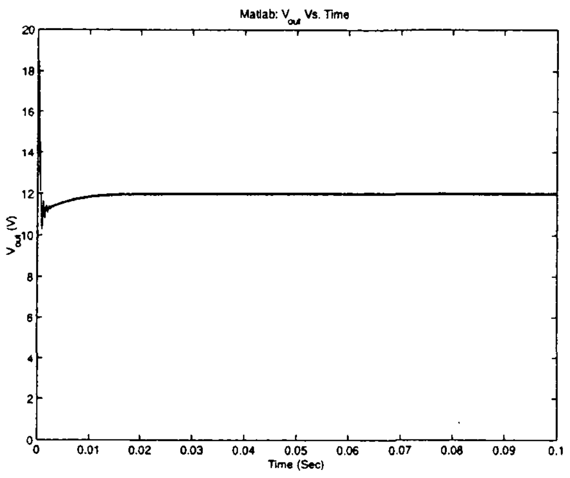


Figure A.1: V_o Vs. Time

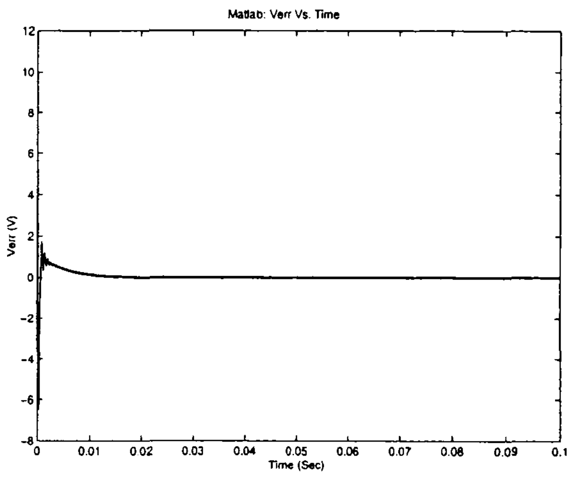
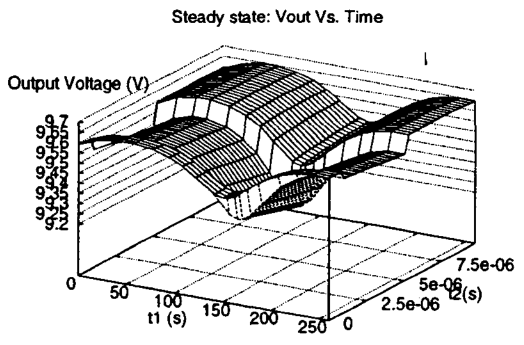
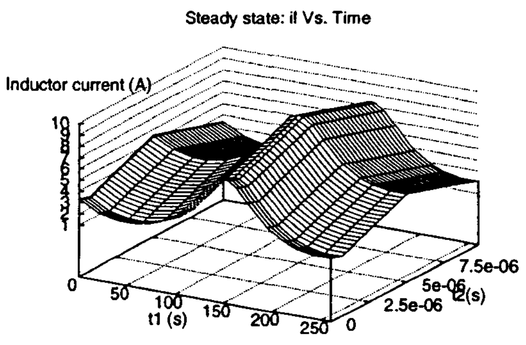


Figure A.2: V_{err} Vs. Time

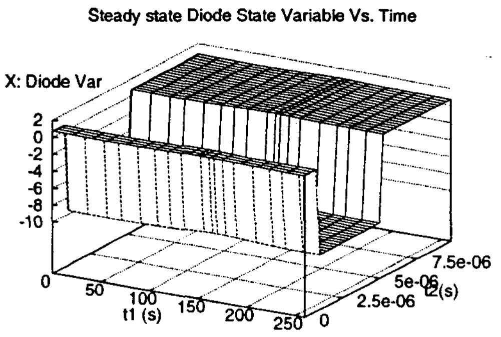
A.7 Steady state Analysis P controller circuit with variable load



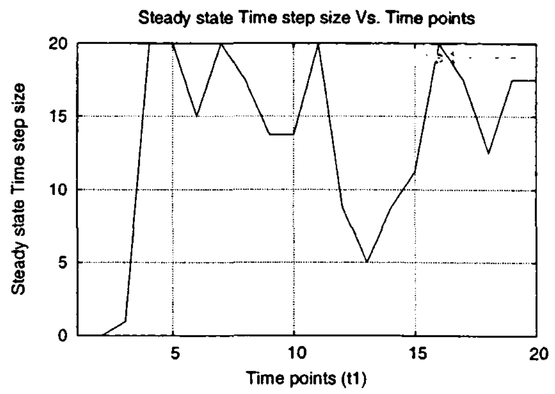
(a)



(b)



(c)



(d)

Figure A.3: This Figures (a), (b), (c), and (d) are Steady state simulation results for variable load converter circuit using P controller alone Vo, il, X, and Time step respectively.

Appendix B

Code

B.1 SPICE Code

```
*** Boost Converter Circuit ***

*.options reltol=.01 abstol=1e-10 chgtol=1e-14 vntol=1e-4 opts

.options method="gear"

.tran 10ns 50ms 0 50ns uic

*** Operational amplifier model ***

.subckt opamp1 1 2 4

b1 3 0 v=12*tanh(100*v(1,2))

rout 3 4 100

cout 3 4 100ff

.ends

.model diodemod d(is=10pA rs=10e-3)

*.model switchmod sw(vt=2V ron=10m roff=1MEG)

.model mosn nmos(level=1 kp=10 rd=10e-3)

*** Power Circuit

vcc 1 0 dc 0 exp 0 6V 0 50us 1s 50us
```

```
l1 1 2 10uH

rsl 2 3 10m

d1 3 4 diodemod

rsc 4 5 20m

c1 5 0 200uF

rl 4 0 4

*s1 3 0 13 0 switchmod off

cswitch 3 0 1nF

m1 3 13 0 0 mosn

roff 3 0 1MEG

*.ic v(4)=0 v(5)=0

*** Control Circuit:

r11 6 4 10k

r12 7 6 10k

x1 15 6 7 opamp1

r13 15 0 10k

r14 19 15 10k

vref 19 0 12V

* Integral gain

r21 17 7 100k

c21 8 17 100nF

x2 0 17 8 opamp1

*.ic v(8)=0 v(17)=0

* Proportional Gain

r31 14 7 10k

r32 9 14 1k
```

```
x3 0 14 9 opamp1

* Adder

r42 10 8 10k

r41 10 9 10k

*r44 10 18 10k

*vadd 18 0 0V

r43 11 10 5k

x4 0 10 11 opamp1

* Limiter and PWM

d2 11 16 diodemod

vlim 16 0 5V

vtr 12 0 dc 0 pulse(6V -6V 0 9.7us .3us 0 10us)

b1 13 0 v=2+2*tanh(10*v(11,12))

.save 4 7 11

.end
```

B.2 Octave Code: ODE

Main Code

```
%~/shared/users/jude/ode/te/trp/5var$ test2.m

% ------- Begin main loop ------

while t1 < t1max;

 to = t1; t1 = to+h2;

 xguess=old_xvec(:,iii-1) + (h2/ho).* (old_xvec(:,iii-1) -

 old_xvec(:,iii-2));

 [old_xvec(:,iii), info] = fsolve("equa2",xguess);
```

```
extraplated_value = old_xvec(:,iii-1) + (h2/2)*((1/h_trap(iii-1)).*
                    (old_xvec(:,iii-1) - old_xvec(:,iii-2))+
(1/h_trap(iii-2)).* (old_xvec(:,iii-2) - old_xvec(:,iii-3)));
absolute_error=(old_xvec(:,iii) -  extraplated_value)
./old_xvec(:,iii);
TE=norm(abs(absolute_error), inf);
while ((info!=1) || (TE>Max_TE)) %False will exit
 if (h2 > hmin)
  h2 =h2/2;
  t1=to+h2;
  xguess=old_xvec(:,iii-1) + (h2/ho).* (old_xvec(:,iii-1) -
         old_xvec(:,iii-2));
  [old_xvec(:,iii), info] = fsolve("equa2",xguess);
  extraplated_value = old_xvec(:,iii-1) + (h2/2)*((1/h_trap(iii-1))
                     .*(old_xvec(:,iii-1) - old_xvec(:,iii-2))+
  (1/h_trap(iii-2)).* (old_xvec(:,iii-2) - old_xvec(:,iii-3)));
  absolute_error=(old_xvec(:,iii) -  extraplated_value)
  ./old_xvec(:,iii);
  TE=norm(abs(absolute_error), inf);
 else
  h2=hmin;
  t1=to+h2;
  xguess=old_xvec(:,iii-1) + (h2/ho).* (old_xvec(:,iii-1) -
  old_xvec(:,iii-2));
  [old_xvec(:,iii), info] = fsolve("equa2",xguess);
  err(iii)=norm(equa2(old_xvec(:,iii)),inf);
```

```
    info=1;

    extraplated_value =old_xvec(:,iii-1) + (h2/2)*((1/h_trap(iii-1)).*

                       (old_xvec(:,iii-1) - old_xvec(:,iii-2))+

    (1/h_trap(iii-2)).* (old_xvec(:,iii-2) - old_xvec(:,iii-3)));

    absolute_error=(old_xvec(:,iii)-extraplated_value)./old_xvec(:,iii);

    TE=(Max_TE+Act_TE)/2;

   endif

  endwhile

  lte(iii)=TE;

  vsum=vsum+Uc;

  ho=h2;

  h_trap(iii)=h2;

  step(iii)=h2;

  time1(iii) = t1;

  check_d(iii)=duty1;

  Switch(iii)=Rs;

  if ((h2<hmax) && (info==1) && (TE<Act_TE))

      h2=2*h2;

  end

  iii=iii+1;

 endwhile

 time_taken=toc;

 % ---------- End main loop ----------
```

Non-linear equation Code: equa2.m function

```
% ~/shared/users/jude/ode/te/trp/5var$ equa2
% DEFINE VARIABLES FOR BOOST Converter

% Soft start

% http://newton.ex.ac.uk/teaching/CDHW/Electronics2/userguide/sec3.html

V1=0; V2=6; TD1=0; TAU1=50e-6; TD2=1; TAU2=50e-6;

if t1<TD2

Vin=V1+(V2-V1)*(1-exp(-(t1-TD1)/TAU1));

else

Vin= 1+(V2-V1)*(1-exp(-(t1-TD1)/TAU1))+(V1-V2)*(1-exp(-(t1-TD2)/TAU2));

endif

%Define all circuit variables

% Neet to save these Control Values for next iteration

Uc=(h2/2)*(Vref-xvec(4)+Vref-old_xvec(4,iii-1));

Vcon=0.5*(Kp*(Vref-xvec(4)) + Ki*(xvec(5)));

if (Vcon>5.7)%5.7

     Vcon=5.7;

endif

duty1=(1/Vref)*Vcon+1/2; %PI controller

%Clip duty between 1 to 0

if (duty1<0)

     duty1=1/2;

endif

Rs=pulse(t1,duty1);

[Vd , id] = diode(xvec(2));

[Vdo, ido]= diode(old_xvec(2,iii-1));

D_il2_old= (old_xvec(1,iii-1)-old_xvec(1,iii-2))/h_trap(iii-1);

D_Vc2_old= (old_xvec(3,iii-1)-old_xvec(3,iii-2))/h_trap(iii-1);
```

```
D_U2_old = (old_xvec(5,iii-1)-old_xvec(5,iii-2))/h_trap(iii-1);

D_il2=2*(xvec(1)-old_xvec(1,iii-1))/h2-D_il2_old;%T2 periodic direction

D_Vc2=2*(xvec(3)-old_xvec(3,iii-1))/h2-D_Vc2_old;%T2 periodic direction

D_U2 =2*(xvec(5)-old_xvec(5,iii-1))/h2-D_U2_old; %T2 periodic direction

%xvec = [il; X; Vc; Vo; U];

% Calculate switch current

vs = Vd+xvec(4);

is = vs / Rs;

% Add switch saturation

if (is > 50)

  is = 50 + log(vs/Rs - 49);

end

vso=Vdo+old_xvec(4,iii-1);

new_ic= C_p*(vs-vso)/h2;

F(1)= (Vin-xvec(1)*Rsl-Vd-xvec(4))*1/L -(D_il2);%F1

F(2)= is + id-xvec(1)+new_ic;                  %F2

F(3)= (id - xvec(4)*(1/Rl))*1/C - (D_Vc2);     %F3

F(4)= ((xvec(4)-xvec(3))/Rsc)*1/C -(D_Vc2);    %F4

F(5)= (D_U2)-Vref+xvec(4);                     %F5

endfunction
```

Diode function

```
# Diode equation

function [Vd , id] = diode(x)

  Vt = 0.02585;

  Is = 10e-12;
  rs = 10e-3;

  v1 = 0.65;

  k2 = Is * exp(v1/Vt);

  k1 = k2 - Is;

  if (x < v1)

    Vd = x;

    id = Is * (exp(x / Vt) - 1);

  else

    Vd = v1 + log(1 + (x - v1)/Vt) * Vt;

    id = k1 + k2 * (x-v1) / Vt;

  endif

  Vd = Vd + id * rs;

endfunction
```

Pulse function

```
function R = pulse(t1,duty1)  # pulse function take time & duty cycle

  global T1;

  global T2;

  global npoints;

  % Output  1 - Low Resistor   0 - High Resistor

  R_on=10e-3;%3

  R_off=1e6;

  tuse= rem(t1,T2);

  fallrise = 0.05*T2; % 5% of T2

  dutyt = duty1*T2;

  if (tuse < fallrise && tuse < dutyt)
```

```
    y = tuse / fallrise; # linear rise
  else
    if (tuse < dutyt)
      y=1;
    else
      if (dutyt > fallrise)
    if (tuse < dutyt+fallrise)
      y = (dutyt+fallrise-tuse) / fallrise; # linear fall
        else
      y=0;
    endif
      else
    if (tuse <  dutyt)
       y = (dutyt-tuse) / fallrise; # linear fall
    else
      y = 0;
    endif
      endif
    endif
  endif
  % Make logarithmic variations
  k = 20;
  y = (exp(k * (1-y)) - 1) / (exp(k) - 1);
R=R_on+(R_off-R_on)*y;
endfunction
```

B.3 Matlab Simulink Block

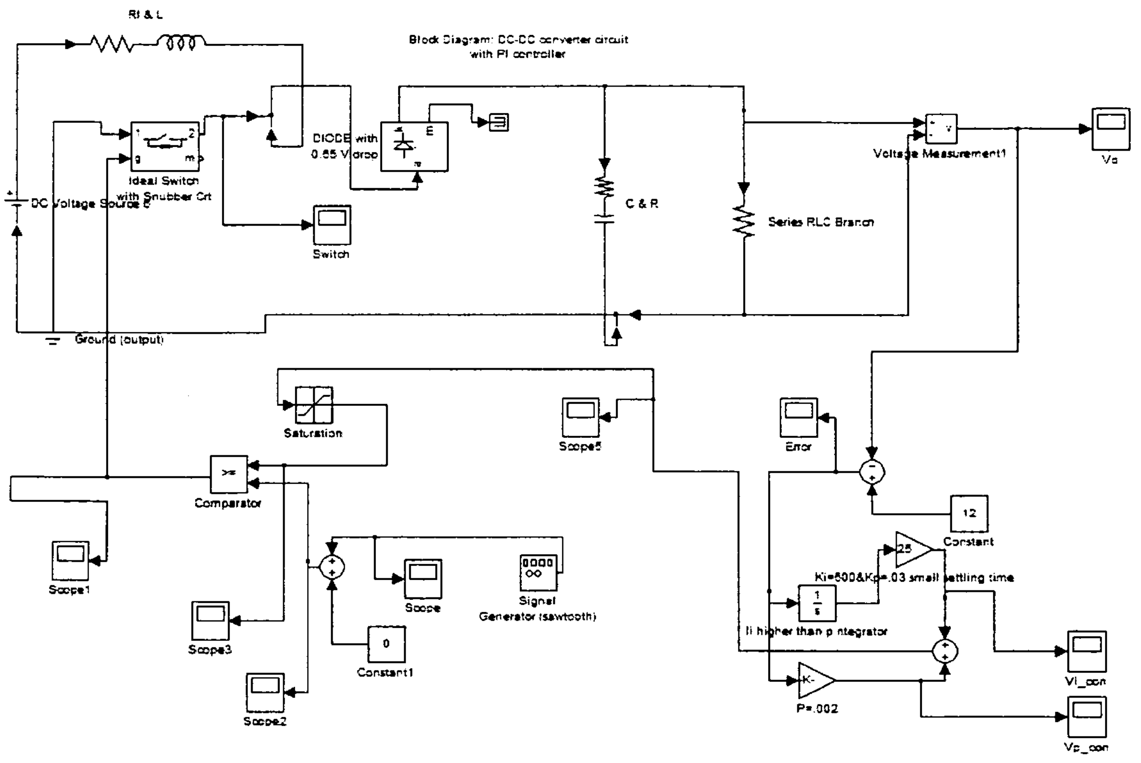


Figure B.1: Matlab Boost converter

Bibliography

[1] J. Roychowdhury and O. Narayan, **"Analyzing Oscillators Using Multitime PDEs"**, *IEEE Transactions on Circuits and Systems - I:* July 2003, pp 894–903.

[2] J. Roychowdhury, **"Analyzing Circuits with Widely Separated Time Scales Using Numerical PDE"**, *IEEE Transactions on Circuits and Systems - I:* May 2001, pp 578–594.

[3] J. C. Pedro and N. B. de Carvalho, **"Simulation of RF circuits driven by modulated signals without bandwidth constraints"**, *2002 IEEE MTT-IMS Digest,* pp. 2173–2176.

[4] J. Roychowdhury, **"Making Fourier-envelope simulation robust,"** *2002 IEEE/ACM ICCAD,* pp. 240-245.

[5] C. E. Christoffersen, M. Ozkar, M. B. Steer, M. G. Case and M. Rodwell, **"State variable-based transient analysis using convolution,"** *IEEE Transactions on Microwave Theory and Techniques,* vol. 47, June 1999, pp. 882–889.

[6] A. Brambilla and P. Maffezzoni, **"Envelope Following Method for the Transient Analysis of Electrical Circuits"**, *IEEE Trans. on Circuits and Systems—I: Fundamental Theory and Applications,* Vol. 47, No. 7, July 2000, pp. 999–1008.

[7] W. Batty, C. E. Christoffersen, A. J. Panks, S. David, C. M. Snowden and M. B. Steer, **"Electro-Thermal CAD of Power Devices and Circuits with Fully Physical

Time-Dependent Compact Thermal Modelling of Complex Non Linear 3-D Systems," *IEEE Transactions on Components and Packaging Technology,* Vol. 34, No. 4, December 2001, pp. 566–590.

[8] A. Wenzler and E. Lueder, "**Analysis of the periodic steady-state in nonlinear circuits using an adaptive function base,"** *1999 IEEE Int. Symposium on Circuits and Systems Digest,* Vol. 6, pp. 1–4.

[9] K. Kundert, "**Introduction to RF Simulation and Its Application**", *IEEE Trans. of Solid-State Circuits,* September 1999, Vol. 34, No. 9, pp. 1298–1319.

[10] A. Howard, "**Applications of an Envelope Simulator**", *Automatic RF Techniques Group 52nd conference,* December 1998, pp. 39–54.

[11] C. E. Christoffersen, M. B. Steer and M. A. Summers, "**Harmonic balance analysis for systems with circuit-field interactions,"** *1998 IEEE Int. Microwave Symp. Dig.*, June 1998, pp. 1131–1134.

[12] E. Ngoya and R. Larcheveque, "**Envelop transient analysis: A new method for the transient and steady state analysis of microwave communication circuits and systems,** *IEEE Microwave Theory and Techniques Symp. Dig.,* pp. 1365–1368, June 1996.

[13] P. Feldmann and J. Roychowdhury,"**Computation of circuit waveform envelopes using an efficient, matrix-decomposed harmonic balance algorithm**", *Digest of Technical Papers on ICCAD,* pp. 295–300, November 1996.

[14] V. Rizzoli, A. Lipparini, A. Costanzo, F. Mastri, C. Ceccetti, A. Neri and D. Masotti, "**State-of-the-Art Harmonic-Balance Simulation of Forced Nonlinear Microwave Circuits by the Piecewise Technique,"** *IEEE Trans. on Microwave Theory and Tech.,* Vol. 40, No. 1, Jan 1992.

[15] D. Feng, J. Phillips, K. Nabors, K. Kundert, and J. White, **"Efficient computation of quasi-periodic circuit operating conditions via a mixed frequency/time approach"**, *Proceedings of 36th Design Automation conference,* pp. 635–640, June 1999.

[16] A. Lehtovuori, J. Virtanen, and M. Valtonen, **"GMRES Preconditioners for Multivariate Steady-State Time-Domain Method,"**, *Proceedings of IMS 2003, Philadelphia,* June 8-13, 2003, pp. 2129–2132.

[17] M. Valtonen, P. Heikkila, A. Kankkunen, K. Mannersalo, R. Niutanen, P. Stenius, T. Veijola and J. Virtanen, **"APLAC - A new approach to circuit simulation by object orientation,"** *10th European Conference on Circuit Theory and Design Dig.*, 1991.

[18] A. R. Djordjevic and T. K. Sarkar, **"Analysis of time response of lossy multiconductor transmission line networks,"** *IEEE Trans. on Microwave Theory and Tech.*, Vol. MTT-35, Oct. 1987, pp. 898–908.

[19] J. E. Schutt-Aine and R. Mittra, **"Nonlinear transient analysis of coupled transmission lines,"** *IEEE Trans. on Circuits and Systems*, Vol. 36, Jul. 1989, pp. 959–967.

[20] P. Stenius, P. Heikkilä and M. Valtonen, **"Transient analysis of circuits including frequency-dependent components using transgyrator and convolution,"** *Proc. of the 11th European Conference on Circuit Theory and Design*, Part II, 1993, pp. 1299–1304.

[21] R. Griffith and M. S. Nakhla, **"Mixed frequency/time domain analysis of nonlinear circuits,"** *IEEE Trans. on Computer Aided Design*, Vol.11, Aug. 1992, pp. 1032–1043.

[22] C. M. Arturi, A. Gandelli, S. Leva, S. Marchi and A. P. Morando, **"Multiresolution analysis of time-variant electrical networks,"** *1999 ISCAS Symp. Digest*, 1999.

[23] D. Zhou, W. Cai and W. Zhang, **"An adaptive wavelet method for nonlinear circuit simulation,"** *IEEE Trans. on Circuits and Systems—I: Fundamental Theory and Appl.*, Vol. 46, pp 931–938, Aug. 1999.

[24] M. S. Nakhla and J. Vlach, **"A Piecewise Harmonic Balance Technique for Determination of Periodic Response of Nonlinear Systems,"** *IEEE Trans. on Circuits and Systems*, Vol CAS-23, No. 2, Feb 1976.

[25] T. J. Brazil, **"A new method for the transient simulation of causal linear systems described in the frequency domain,"** *1992 IEEE MTT-S Int. Microwave Symp. Digest*, June 1992, pp. 1485–1488.

[26] C. E. Christoffersen, S. Nakazawa, M. A. Summers, and M. B. Steer, **"Transient analysis of a spatial power combining amplifier"**, *1999 IEEE MTT-S Int. Microwave Symp. Dig.*, June 1999, pp. 791–794.

[27] A. Brambilla, D. D'Amor and M. Pillan, **"Convergence improvements of the harmonic balance method,"** *Proceedings IEEE Int. Symposium on Circuits and Systems*, June 1993, Vol. 4 pp. 2482–2485.

[28] T. W. Nuteson, H. Hwang, M. B. Steer, K. Naishadham, J.W.Mink, and J. Harvey, **"Analysis of finite grid structures with lenses in quasi-optical systems,"** *IEEE Trans. Microwave Theory Tech.*, pp. 666–672, May 1997.

[29] M. B. Steer, M. N. Abdullah, C. Christoffersen, M. Summers, S. Nakazawa, A. Khalil, and J. Harvey, **"Integrated electro-magnetic and circuit modeling of large microwave and millimeter-wave structures,"** *Proc. 1998 IEEE Antennas and Propagation Symp.*, pp. 478–481, June 1998.

[30] C. E. Christoffersen, and J. Alexander,"**An Adaptive time step Control Algorithm for Nonlinear Time Domain Envelope Transient**", *2004 IEEE Canadian Conference on Electrical and Computer Engineering Digest*, June 2004, pp. 791–794.

[31] C. Britt, "**Solution of electromagnetic scattering problems using time domain techniques**", *IEEE Trans. Antennas Propagat.* Vol. 37, 1181–1192 (1989).

[32] C. Furse, S. Mathur, and O. Gandi, "**Improvements to the finite-difference time-domain method for calculating the radar cross section of a perfectly conducting target**", *IEEE Trans. Microwave Theory Tech.* 38, 919–927 (1990).

[33] C. E. Christoffersen, *Global Modeling of Nonlinear Microwave Circuits,* Ph.D. Dissertation. North Carolina State University, 2000. pp. 19–22.

[34] Laurence W. Nagel, *SPICE2: A computer program to simulate semiconductor circuits*, PhD Thesis, University of California, Berkeley. 1975.

[35] B. Troyanovsky, *Frequency Domain Algorithms for Simulating Large Signal Distortion in Semiconductor Devices*, PhD Thesis, Stanford University, 1997.

[36] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhold, 1994. pp 263-282, 100–129.

[37] P. J. C. Rodrigues, *Computer Aided Analysis of Nonlinear Microwave Circuits*, Artech House, 1998.

[38] A. Taflove, *Computational Electrodynamics: The Finite Difference Time Domain Method*, Artech, Boston, 1995.

[39] K. S. Kundert and A. Songiovanni-Vincentelli, *Sparse user's guide - a sparse linear equation solver*, Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, Calif. 94720, Version 1.3a, Apr 1988.

[40] T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, McGraw-Hill Book Company, 1990.

[41] S. Hayken, *Communication System,* Fourth Edition, John Wiley and Sons, 2000.

[42] R. L. Burden and D. J. Faires, *Numerical Analysis,* McGraw-Hill, 1993. pp 157–178, 555.

[43] C. G. Dahlquist, *A Special Stability Problem for Linear Multistep Methods,* BIT. 3, 1963, pp.27–43.

[44] C. K. Cheng, *CSE 245 Slides and Lecture Notes* Department of Computer Science and Engineering. University of California, San Diego.

[45] S. L. Ross, *Differential Equations.* WCB McGraw-Hill, 1955. pp. 3–20.

[46] L. O. Chua, Charles A. Desoer and Ernest S. Kuh., *Linear and Nonlinear Circuits,* McGraw Hill, 1987.

[47] K. S. Kundert, J. K. White and A. Sangiovanni-Vincentelli, *Steady-state methods for simulating analog and microwave circuits*, Boston, Dordrecht, Kluwer Academic Publishers, 1990.