A Study in the Frequency Warping of Time-Domain Methods

by

Kai Gao

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

This thesis develops a study for the frequency warping introduced by time-domain methods. The work in this study focuses first on the time-domain methods used in the classical SPICE engine, that is the Backward Euler, the Trapezoidal Rule and the Gear's methods. Next, the thesis considers the newly developed high-order method based on the Obreshkov formula. This latter method was proved to have the A-stability and L-stability properties, and is therefore robust in circuit simulation. However, to the best of the author's knowledge, a mathematical study for the frequency warping introduced by this method has not been developed yet.

The thesis therefore develops the mathematical derivation for the frequency warping of the Obreshkov-based method. The derivations produced reveal that those methods introduce much smaller warping errors than the traditional methods used by SPICE. In order to take advantage of the small warping error, the thesis develops a shooting method framework based on the Obreshkov-based method to compute the steady-state response of nonlinear circuits excited by periodical sources. The new method demonstrates that the steady-state response can be constructed with much smaller number of time points than what is typically required by the classical methods.

Acknowledgements

I would like to express the deepest appreciation to my supervisor, Dr. Emad Gad, for his patient guidance, caring, encouragements and kind help and providing me with an excellent atmosphere for doing research. Without his valuable suggestions and immediate help this thesis would not have been possible.

I would also like to thank Dr. Michel Nakhla for the essential background knowledge learned in his courses.

Specially, I would like to thank my parents and my girlfriend who were always cheering me up and supporting me with their best wishes.

Contents

List of Figures

List of Algorithms

Nomenclature

BE	Backward Euler method
TR	Trapezoidal Rule method
BDFs	Backward differentiation formulas or Gear's methods
LMS	Linear multistep methods
LTE	Local truncation error
DE	Differential equation
MNA	Modified Nodal Analysis
NR	Newton-Raphson method
$\boldsymbol{G}$	MNA matrice describing the memoryless elements in a circuit
$\boldsymbol{C}$	MNA matrice describing the energy storage elements in a circuit
$\boldsymbol{f}(\boldsymbol{x}(t))$	A vector consists of the nonlinear elements in MNA formulation
$\boldsymbol{b}(t)$	A vector consists of independent stimuli in MNA formulation
$\boldsymbol{x}_n$	Approximation for exact value of $\boldsymbol{x}(t)$ at time $t = t_n$
h	Time step size

h_n	Time step size between t_{n-1} and t_n
$\mathbb{R}$	Real variable space
$\mathbb{C}$	Comoplex variable space
α_i, β_i	Coefficients in Obreshkov-based high-order method
$\tilde{\boldsymbol{G}}, \tilde{\boldsymbol{C}}, \tilde{\boldsymbol{f}}, \tilde{\boldsymbol{b}}, \boldsymbol{\xi}$	Augmented matrices and vectors describing the circuit after applying the Obreshkov-based method
$\tilde{\boldsymbol{J}}$	Block Jacobian matrix
$\boldsymbol{J_{i,j}}$	The entry (i, j) of matrix $\tilde{\boldsymbol{J}}$

Chapter 1

Introduction

1.1 Problem Description and Motivation

Time-domain circuit simulation plays a very important role in the analysis and design of electronic circuits. The behavior of the circuits can be numerically modeled by a set of differential equations (DEs). Circuit simulators adopting some numerical integration formulas solve a system of differential equations approximately at a set of discrete time points. The set of nonlinear differential equations can be transformed into a set of nonlinear algebraic equations by using numerical integration methods. In time-domain circuit simulation, the class known as linear multistep methods (LMS) has been most commonly used and examples from this class include the Backward Euler (BE) method, the Trapezoidal Rule (TR) method and backward differentiation formulas (BDFs, also known as Gear's methods).

Reference [1] introduced a new type of error measure, denoted as the warping error. Unlike the local truncation error (LTE), the warping error copes with the error introduced at each time instance accumulating in the process of integration. Warping error is formulated as a multiplication to show how the eigenvalues of the circuit characterized

by differential equations get shifted from the original ones. Reference [1] showed that the warping error depends on the order of the accuracy of the LMS integration method. It also showed that for a fixed integration time step size, the higher order LMS integration methods introduce smaller warping error. As a result, higher order linear multistep methods are more suitable for circuit simulation in order to limit the warping effects.

Typically, time-domain simulators are expected to be very accurate while requiring limited resources of computation and shorter execution time. However, in most cases, these requirements are in conflict with each other due to the conflict between the order and stability of the integration method used in the process of solving the system of differential equations. Germund Dahlquist proved that the highest order of absolutely stable (i.e., circuits are not stable if the length of the chosen integration time step is too long) implicit LMS method is 2 [2]. In order to avoid the instability issue, most circuit simulators adopt either trapezoidal rule or the gear method of order 2 (a backward differentiation formula [3] [4]).

As stated in the previous paragraph, choosing a higher order integration method is better for circuit simulation in terms of limiting the warping error. Reference [5] addressed the above conflict by developing a new integration method, which is based on the Obreshkov formula [6]. This new integration method can be made of high order while maintaining the desired A-stability (or L-stability) property in solving the DEs. This integration method enables the transient analysis of electronic circuits to take larger step size, leading to shorter execution time compared with the classical low-order integration methods (i.e., TR or BDFs) adopted in most conventional simulators today. Furthermore, recent publications [7] [8] [9] [10] also showed that the improvement of the new A-stable and L-stable high-order method for time-domain transient simulation.

1.2 Thesis Objectives

The first objective of this thesis is to explore whether the new A-stable and L-stable high-order integration method can reduce the warping error compared with the classical low-order methods such as the Backward Euler method, the Trapezoidal Rule method and backward differentiation formulas. More specifically, the thesis seeks to derive the warping error expression of the high order method and compare it with those of classical low-order methods. Theoretical analysis and numerical examples are presented for this objective.

The second objective is to develop a shooting method framework based on the Obreshkov method to compute the steady-state response of nonlinear circuits exited by periodic sources.

1.3 Thesis Contributions

The first contribution of this thesis is showing that the new A-stable and L-stable high-order integration method indeed introduce much less warping error compared with the traditional low-order methods such as BE, TR, Gear method of order 2, which are most commonly adopted in circuit simulator. As a result, adopting the high-order integration method can not only achieve better accuracy but also maintain both the A-stability and L-stability.

The second contribution presents a shooting method framework based on the Obreshkov-based high-order method that achieves superior accuracy while requiring shorter execution time to compute the steady-state response of nonlinear circuits.

1.4 Thesis Organization

This thesis is organized as follows. Chapter 2 presents the numerical solutions of differential equations. Chapter 3 presents the derivation of the warping error for the classical integration methods. Chapter 4 presents the derivation of the warping error for the the A-stable and L-stable high-order integration method. Chapter 5 presents derivation of the shooting method based on the A-stable and L-stable high-order integration method. Chapter 6 presents numerical simulations for both low-order and high-order methods adopted in shooting method.

Chapter 2

Numerical Solution of Differential Equations

This chapter presents a general background for both classical integration methods and A-stable and L-stable high-order integration methods in time-domain circuit simulation. In particular, two central concepts of these integration methods, the order and the stability of the method, will be discussed in the chapter as well as some overviews of transient simulations in these methods.

2.1 Introduction

We discuss both the classical and high-order integration methods for solving a system of nonlinear differential equations in the following form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, t) \tag{2.1}$$

where $\boldsymbol{x} \in \mathbb{R}^N$ is a vector of unknowns and $\boldsymbol{f}(x,t) : \mathbb{R}^{N+1} \rightarrow \mathbb{R}^N$ is a nonlinear mapping. The main objective in any integration method employed to solve the above system of

differential equations is to approximate the exact solution $\boldsymbol{x}(t)$ at discrete time points $t_0, t_1, t_2, \ldots$ In this thesis, $\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ refer to the approximations generated by the integration method and $\boldsymbol{x}(t_0), \boldsymbol{x}(t_1), \boldsymbol{x}(t_2), \ldots$ denote the exact values of the solution.

2.2 Order and Local Truncation Error and Stability

Order Order is an important characteristic for a numerical integration method [11] [12]. We call an integration method order of k if

$$\boldsymbol{x}_n = \boldsymbol{x}(t_n) + Ch_n^{k+1} \frac{d^{k+1}}{dt^{k+1}} \boldsymbol{x}(t)\Big|_{t=t_n} + O(h_n^{k+2}). \tag{2.2}$$

where the coefficient C is called the error constant of the method and its magnitude reflects the accuracy of the integration method [13]. The increment h_n is the time step and $h_n = t_n - t_{n-1}$. $O(h^w)$ denotes the terms that are proportional to powers of h that are larger than or equal to w.

Local Truncation Error All integration methods introduce errors, to different degree, in the process of finding numerical approximations to the solutions of differential equations. The Local Truncation Error (LTE) [14] is one possible way to measure the error introduced by a given method although we discuss a another type of error measure (the warping error) throughout this thesis. LTE is defined as the $(k+1)^{th}$ term in the above expansion (2.2) as follows

$$\text{LTE} = Ch_n^{k+1} \frac{d^{k+1}}{dt^{k+1}} \boldsymbol{x}(t)\Big|_{t=t_n}. \tag{2.3}$$

LTE describes the difference between the approximation and the exact solution after only one time step when applying an integration method, assuming that no previous

error has been made, i.e., $x_{n-1} = x(t_{n-1})$. It can be seen that, for the same step size, an integration method with higher order introduces less LTE and this indicates that higher order methods bring more accurate approximations.

Stability Another important characteristic of the numerical integration methods is stability. Testing an integration method for stability is usually via the following scalar test equation [15].

$$\begin{aligned} \frac{dx(t)}{dt} &= \lambda x(t), \\ \lambda \in \mathbb{C}; \quad \Re(\lambda) &\in \mathbb{R}^{-}; \end{aligned} \tag{2.4}$$

where λ is a complex parameter and denotes the eigenvalue of a system. $\Re(\lambda)$ describes the real part of λ. The analytical solution to the scalar test equation (2.4) is as follows

$$x(t) = x_0 e^{\lambda t}, \tag{2.5}$$

where $x(t_0) = x_0$ is the initial condition at the time point t_0. The analytical solution is decaying exponentially if $\Re(\lambda) < 0$.

Naturally, we expect an integration method deployed to approximate the solution of the system can also satisfy the following condition (A-stable requirement) for the two successive time points (i.e., t_n, t_{n+1}),

$$|x_n| \geq |x_{n+1}| \quad n = 1, 2, ... \tag{2.6}$$

A numerical integration method is called A-stable if its approximations to (2.4) satisfies the absolute stability requirement (2.6).

The stability region of a given integration method refers to a region in the complex λh-plane where the approximate solutions generated by the method to the scalar test

equation (2.4) satisfy the absolute stability requirement (2.6). In other words, when deploying an integration method to approximate solutions for the above scalar test problem, the step size h must be chosen to ensure that λh lie on or inside the stability region.

2.3 Classical Integration Methods and Their Stability

This section reviews briefly some classical integration methods and their stability characteristics. We focus on the linear multistep (LMS) integration methods, since LMS is an important class of existing integration methods and most commercial circuit simulation softwares are based on them in the time-domain circuit simulation. In particular, we consider the Forward Euler (FE) method, the Backward Euler (BE) method and the Trapezoidal Rule (TR) method as well as the backward differentiation formulas (BDFs, also known as Gear's methods) [16].

To simplify the discussion of using these integration methods, we will first denote by x_n the approximation obtained by a given method to $x(t)$ at $t = t_n$.

2.3.1 Forward Euler Method and Its Stability Analysis

Let us consider the Forward Euler method first for the sake of simplicity. The FE method is defined by

$$x_{n+1} = x_n + hx'_n \tag{2.7}$$

where h denotes the time step size and $h = t_{n+1} - t_n$ and x'_n refers to the approximation of $\frac{dx}{dt}$ at $t = t_n$. In other words, FE method approximates $x(t)$ at $t = t_{n+1}$ in terms of the approximation of $x(t)$ and the approximation of its derivative, both obtained at $t = t_n$.

To study the stability characteristics of the FE method, inserting the scalar test equa-

tion (2.4) into the FE formula (2.7) for the derivative term, we have the approximation at $t = t_{n+1}$, i.e. x_{n+1} as follows

$$x_{n+1} = (1 + h\lambda)x_n \tag{2.8}$$

Calculating the values of x at time points from t_0 to t_n through (2.8) as follows

$$\begin{aligned}
&t_0 = 0 \longrightarrow x(t_0) = x_0 \\
&t_1 = h \longrightarrow x(t_1) = x_1 = (1 + h\lambda)x_0 \\
&t_2 = 2h \longrightarrow x(t_2) = x_2 = (1 + h\lambda)x_1 = (1 + h\lambda)^2 x_0 \\
&\vdots \\
&t_n = nh \longrightarrow x(t_n) = x_n = (1 + h\lambda)^n x_0
\end{aligned} \tag{2.9}$$

If the following condition is satisfied by the FE scheme

$$|1 + h\lambda| < 1 \tag{2.10}$$

we can have $n \to \infty \Rightarrow x_n \to 0$. Considering the definition of the stability region discussed above, it can be seen that (2.10) is the stability condition for the FE method. Introducing a new variable $z = \lambda h$ and substituting λh with the complex polar form, $x + jy$, lead to

$$|1 + x + jy| \leq 1 \quad or \quad (1 + x)^2 + y^2 \leq 1 \tag{2.11}$$

This relation depicts a region inside a circle with center at $x = -1$, $y = 0$ passing through the origin in the complex plane and it can be seen in Figure 2.1.

It can be seen from the above figure that the stability region of the FE method is inside the circle and that FE method is stable for some small values of time step size h. If the

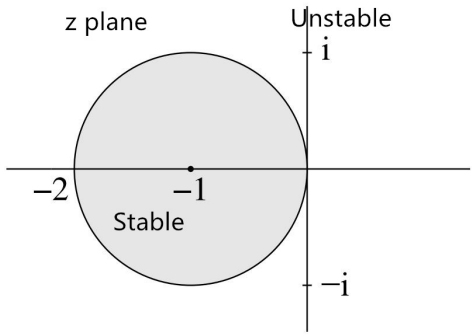

Figure 2.1: The shaded area represents the stability region of the FE method

magnitude of λ is large, the time step h must be reduced to make $z = \lambda h$ fall inside the unit circle.

2.3.2 Backward Euler Method and Its Stability Analysis

The Backward Euler (BE) formula is based on the following formula [17]

$$x_{n+1} = x_n + hx'_{n+1} \tag{2.12}$$

to approximate $x(t)$ at $t = t_{n+1}$ given an approximation for $x(t)$ at $t = t_n$, i.e. x_n. Here x'_{n+1} refers to the approximation of $\frac{dx}{dt}$ at $t = t_{n+1}$.
Inserting the scalar test equation (2.4) into the BE formula (2.12) for the derivative, we have the approximation x_{n+1} expressed in terms of x_n as follows

$$x_{n+1} = (1 - h\lambda)^{-1}x_n = (1 - z)^{-1}x_n \tag{2.13}$$

then x_n can be expressed in terms of the initial condition x_0

$$x_n = (1 - z)^{-n}x_0 \tag{2.14}$$

Proceeding in similar steps as with FE method leads to the following stability condition for the BE method

$$|(1-z)^{-1}| \leq 1 \tag{2.15}$$

then substituting z with the complex polar form, $x + jy$, leads to

$$(1-x)^2 + y^2 \geq 1 \tag{2.16}$$

The above relation describes a region outside a circle with center at $x = 1$, $y = 0$ passing through the origin in the complex plane as shown in Figure 2.2. It can be seen that the Backward Euler method yields decaying approximations to the test equation (2.4) for all λ, regardless of the value of step size h, in the left half plane. Thus, the BE method is an A-stable integration method.

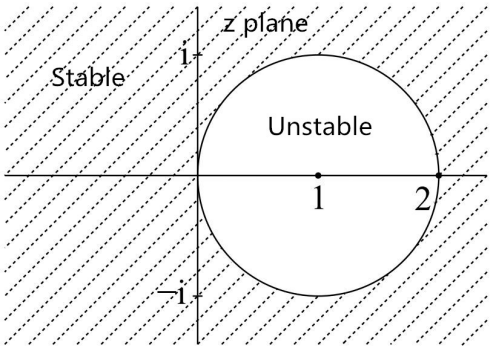

Figure 2.2: The shaded area represents the stability region of the BE method

2.3.3 Trapezoidal Rule Method and Its Stability Analysis

The Trapezoidal Rule (TR) method is defined by

$$x_{n+1} = x_n + \frac{h}{2}(x'_n + x'_{n+1}) \tag{2.17}$$

Inserting the scalar test equation (2.4) into the TR formula (2.17) for the derivative terms, x'_n and x'_{n+1}, we have the approximation x_{n+1} expressed in terms of x_n as follows

$$x_{n+1} = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} x_n \tag{2.18}$$

Proceeding in similar steps as with BE method leads to the stability region for the TR method as shown in Figure 2.3. It can be seen that the imaginary axis is the boundary of the stability region for the TR method and that the TR yields decaying approximations to the test equation (2.4) for all values of λ, regardless of the step size h, in the left half plane. In other words, the left half plane of the complex plane is the stability region of the TR method. Therefore the TR is also an A-stable integration method.

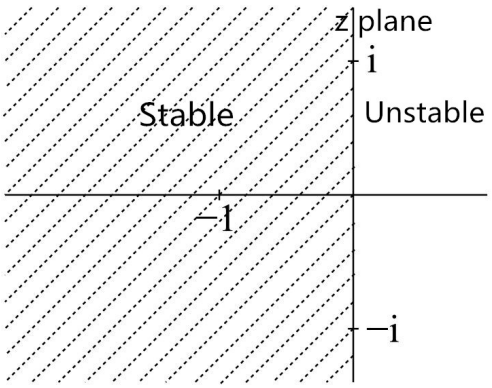

Figure 2.3: The shaded area represents the stability region of the TR method

2.3.4 Backward Differentiation Formulas and Their Stability Analysis

The general form of a linear k-step method [18] can be described as follow

$$\sum_{i=0}^{k} \alpha_i x_{n+k-i} = h \sum_{i=0}^{k} \beta_i x'_{n+k-i} \tag{2.19}$$

where h is fixed time step size, k is the order of the LMS method and α_i and β_i are the coefficients that determine the LMS method [16] [3]. Some coefficients may be zero to simplify the method.

The Gear's methods (also known as the backward differentiation formulas) is a family of linear multistep methods with $\beta_1 = \beta_2 = \cdots = \beta_k = 0$, and Gear's method of order k can be represented by the following formula

$$\sum_{i=0}^{k} \alpha_i x_{n+i} = h\beta_0 x'_{n+k} \tag{2.20}$$

where the coefficients α_i and β_0 are chosen such that the Gear's method achieves order k. The k-step Gear's method is an order k method.

The simplest Gear's method is Backward Euler method (Gear's method of order 1) and the following are some other specific formulas of Gear's method.

Gear's method of order 2:

$$x_{n+2} - \frac{4}{3}x_{n+1} + \frac{1}{3}x_n = \frac{2}{3}hf(x_{n+2}, t_{n+2}) \tag{2.21}$$

Gear's method of order 3:

$$x_{n+3} - \frac{18}{11}x_{n+2} + \frac{9}{11}x_{n+1} - \frac{2}{11}x_n = \frac{6}{11}hf(x_{n+3}, t_{n+3}) \tag{2.22}$$

Gear's method of order 4:

$$x_{n+4} - \frac{48}{25}x_{n+3} + \frac{36}{25}x_{n+2} - \frac{16}{25}x_{n+1} + \frac{3}{25}x_n = \frac{12}{25}hf(x_{n+4}, t_{n+4}) \tag{2.23}$$

Figure 2.4 shows the stability regions of the Gear's methods of order 1 to 6, and all curves close in the right half plane of the complex plane. Dahlquist proved that implicit

LMS methods of order higher than 2, are not A-stable [2] [19], which means that the highest order of A-stable Gear's methods is 2.

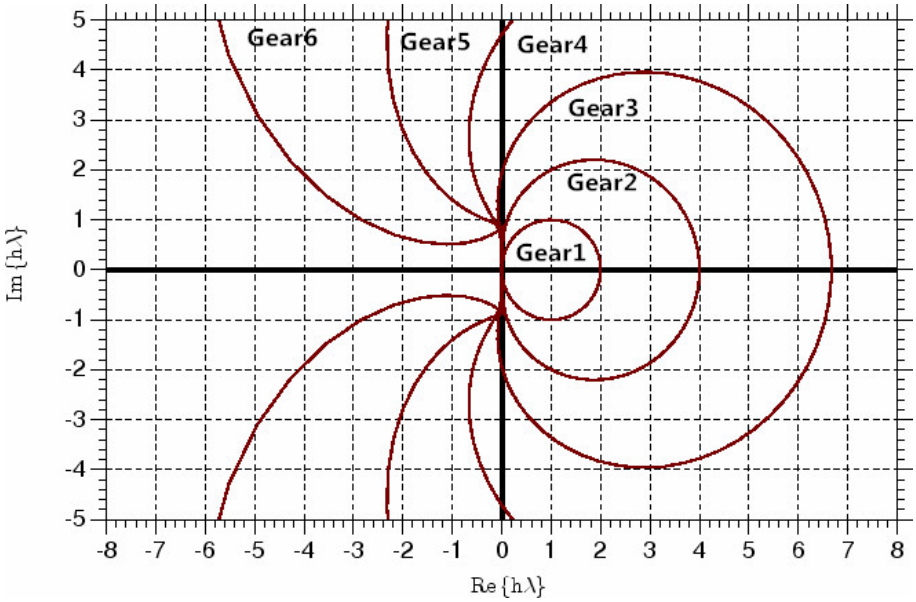

Figure 2.4: Stability regions of Gear's method of order 1 to 6 are outside the curves and all curves close in the right half plane of the complex plane

2.3.5 A-Stability and L-stability

The A-stability [11] has been discussed above and it is a necessary property for any integration method in circuit simulation, since simulations of a stable circuit needs to be stable [20]. For example, the stability region of FE method in Figure 2.1 shows that in

order to yield decaying approximations to the test function (2.4), the value of $h\lambda$ must lie in the unit circle. If $h\lambda$ lies outside the unit circle, the approximations resulting from FE are bound to become unstable. Figure 2.1 shows that the stability region of FE method can not cover the whole left half plane of the complex plane, so the FE method is not A-stable. Both the BE and TR methods are A-stable since their stability regions (shown in Figures 2.2 and 2.3) cover the entire left half plane.

L-stability [15] [21]is also another important concept of stability in circuit simulation, particularly in the so-called stiff circuits. A stiff system typically has some eigenvalues close to the imaginary axis and some eigenvalues lie very far away from them, all in the left half plane. We still consider the test function (2.4), an integration method is said to be L-stable if the following condition is satisfied by its two successive approximations (i.e., x_n and x_{n+1}) [22] [23] [24]

$$\left|\frac{x_{n+1}}{x_n}\right| \to 0 \quad as \quad |\lambda| \to \infty \tag{2.24}$$

For example, the BE method is L-stable, whereas the TR method is not L-stable.

2.4 Obreshkov-based High-Order Method

Reference [5] presented a new integration method based on the Obreshkov formula [6] [25] for time domain circuit simulation. This method is constructed by including information from high-order derivatives. The Obreshkov formulas were first proposed in 1942 [6] as follows

$$\sum_{i=0}^{k} \alpha_{i,k}(-1)^i h^i \boldsymbol{x}_{n+1}^{(i)} = \sum_{i=0}^{k} \alpha_{i,k} h^i \boldsymbol{x}_n^{(i)} \tag{2.25}$$

this is a general form of one-step methods and it is used to approximate $\boldsymbol{x}_{n+1}$. $\boldsymbol{x}_n^{(i)}$ above denotes the approximation to $\frac{d^i \boldsymbol{x}(t)}{dt^i}$ at the time point $t = t_n$,

$$\boldsymbol{x}_n^{(i)} \approx \left. \frac{d^i \boldsymbol{x}(t)}{dt^i} \right|_{t=t_n} \tag{2.26}$$

assuming that $\boldsymbol{x}_n^{(i)}, i = 0, 1, ..., k$ are known. $\alpha_{i,k}$ are the constant coefficients.

The above set of formulas were modified in [5][26] as follows,

$$\sum_{i=0}^{k} \alpha_i h^i \boldsymbol{x}_{n+1}^{(i)} = \sum_{i=0}^{m} \beta_i h^i \boldsymbol{x}_n^{(i)} \tag{2.27}$$

where $m = k - 1$ or $m = k - 2$ and α_i and β_i are the coefficients of $\boldsymbol{x}_{n+1}^{(i)}$ and $\boldsymbol{x}_n^{(i)}$ respectively, given by

$$\alpha_i = (-1)^i \frac{(m+k-i)!}{(m+k)!} \frac{k!}{i!(k-i)!} \tag{2.28}$$

$$\beta_i = \frac{(m+k-i)!}{(m+k)!} \frac{m!}{i!(m-i)!} \tag{2.29}$$

Our goal is to use this modified Obreshkov formula (2.27) to approximate the unknown vector $\boldsymbol{x}(t)$ in (2.31) in the next section at discrete time instances $t_0, t_1, t_2, ...$ In order to accomplish this, the modified Obreshkov formula (2.27) is to be satisfied by the approximations of $\boldsymbol{x}(t)$ at two successive time instances (i.e., t_n, t_{n+1}).

As mentioned above in (2.26), we can see that $\boldsymbol{x}_n^{(0)} \approx \boldsymbol{x}|_{t=t_n}$. So similarly, we can have $\boldsymbol{x}_{n+1}^{(0)} \approx \boldsymbol{x}|_{t=t_{n+1}}$. $\boldsymbol{x}_n^{(i)}, i = 0, 1, ..., m$ are assumed to be approximated already and $\boldsymbol{x}_{n+1}^{(0)} \approx \boldsymbol{x}|_{t=t_{n+1}}$ is the approximation for the actual value of the waveform in the vector $\boldsymbol{x}(t)$ at the time point $t = t_{n+1}$.

The order and stability characteristics of the Obreshkov-based high-order formula depend on the m and k selected. [5] has proved that the Obreshkov-based high-order integration method (2.27) can approximate the solution of the differential equations up

to the order $(k+m)$, that is

$$\boldsymbol{x}_{n+1}^{(0)} - \boldsymbol{x}(t_{n+1}) \approx O(h^{k+m+1}) \tag{2.30}$$

where $\boldsymbol{x}(t_{n+1})$ represents the exact value of the waveform in the vector $\boldsymbol{x}(t)$ at the time point $t = t_{n+1}$.

It was shown in [5] [9] that the method can be both A-stable and L-stable under the condition: $k-2 \leq m \leq k$. Therefore, this enables the circuit simulations to take larger step sizes without violating the stability, leading to faster simulation compared with the classical low-order methods.

2.5 Application in Circuit Simulation

2.5.1 Mathematical Representation of General Circuits

In circuit simulation, we use differential equations to describe a general linear or nonlinear circuit in the time domain by using Modified Nodal Analysis (MNA) [27]

$$\boldsymbol{G}\boldsymbol{x}(t) + \boldsymbol{C}\frac{d\boldsymbol{x}(t)}{dt} + \boldsymbol{f}(\boldsymbol{x}(t)) = \boldsymbol{b}(t) \tag{2.31}$$

In this general equation, the memoryless elements in the circuit are described in $\boldsymbol{G} \in \mathbb{R}^{N\times N}$ matrix and respectively $\boldsymbol{C} \in \mathbb{R}^{N\times N}$ matrix is used to describe the energy storage elements. Node voltages, independent voltage sources and currents in inductors are described in the vector $\boldsymbol{x}(t) \in \mathbb{R}^N$. And the vector $\boldsymbol{b}(t) \in \mathbb{R}^N$ represents independent stimuli in the circuit. For nonlinear circuit, nonlinear vector $\boldsymbol{f}(\boldsymbol{x}(t))$ represents the nonlinear elements, such as the nonlinear inductors and capacitors as well as the currents of nonlinear conductances.

As discussed above, the behavior of the circuit can be described by a formulation of a system of nonlinear differential equations. Since both Backward Euler and Trapezoidal Rule are A-stable, they represent the simple but great integration methods for implementation in time-domain simulation of general nonlinear circuits. In this section, we consider the applications of BE and TR as examples to show how classical integration methods can be applied in transient simulation of nonlinear circuits.

First, let us rearrange the MNA formulation (2.31) into a more convenient form:

$$\boldsymbol{C}\frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{G}\boldsymbol{x}(t) - \boldsymbol{f}(\boldsymbol{x}(t))\boldsymbol{b}(t) \tag{2.32}$$

Our goal here is to compute the vector of unknown waveforms $\boldsymbol{x}$ at different time instants by using the Backward Euler and Trapezoidal Rule integration methods.

2.5.2 Transient Analysis of General Nonlinear Circuits using Backward Euler Method

Let us consider the Backward Euler method first. According to the BE formula (2.12), we can approximate the vector of the values of the MNA waveforms $\boldsymbol{x}(t_1)$ at the time point t_1:

$$\boldsymbol{x_1} = \boldsymbol{x_0} + h\frac{d\boldsymbol{x}(t)}{dt}\bigg|_{t=t_1} \tag{2.33}$$

where

$$\boldsymbol{x_0} = \boldsymbol{x}(t_0) \tag{2.34}$$

To approximate $\boldsymbol{x_1}$, multiplying both sides of (2.33) by the the matrix $\boldsymbol{C}$, and we can get

$$\boldsymbol{C}\boldsymbol{x_1} = \boldsymbol{C}\boldsymbol{x_0} + h\boldsymbol{C}\frac{d\boldsymbol{x}(t)}{dt}\bigg|_{t=t_1} \tag{2.35}$$

To write the MNA formulation (2.31) at the time point $t = t_1$, we can get

$$\boldsymbol{C}\frac{d\boldsymbol{x}(t)}{dt}\bigg|_{t=t_1} = -\boldsymbol{G}\boldsymbol{x}(t_1) - \boldsymbol{f}(\boldsymbol{x}(t_1)) + \boldsymbol{b}(t_1) \tag{2.36}$$

Let us substitute (2.35) into (2.36). In other words, we use the Backward Euler integration formula to eliminate the time derivatives in the system of differential equations. After rearrangement we can get a system of nonlinear algebraic equations as follows

$$(\boldsymbol{C} + h\boldsymbol{G})\boldsymbol{x_1} + h\boldsymbol{f}(\boldsymbol{x_1}) = \boldsymbol{C}\boldsymbol{x_0} + h\boldsymbol{b}(t_1) \tag{2.37}$$

Now, the system of nonlinear equations in $\boldsymbol{x_1}$ can be described by the above system of equations. We assume that the right side is already known.

Further, for the generalization of the Backward Euler method, we can use the values of $\boldsymbol{x}(t)$ at the previous time point $t = t_{n-1}$ to compute the current values of $\boldsymbol{x}(t)$ at the current time point $t = t_n$. Thus , for the general case, we can get the corresponding systems of equations as follows

$$(\boldsymbol{C} + h_n\boldsymbol{G})\boldsymbol{x_n} + h_n\boldsymbol{f}(\boldsymbol{x_n}) = \boldsymbol{C}\boldsymbol{x_{n-1}} + h_n\boldsymbol{b}(t_n) \tag{2.38}$$

To calculate $\boldsymbol{x_n}$, we need to solve the system of nonlinear algebraic equations by using some iterative technique such as Newton-Raphson (N-R) mehtod. The Newton-Raphson method compute the $\boldsymbol{x}(t)$ at various time points with a linear set of equations based on a good initial guess, say $\boldsymbol{x_n}^{(0)}$. Then, the N-R method proceeds to modify the initial guess $\boldsymbol{x_n}^{(0)}$ based on the correction term $\Delta\boldsymbol{x}^{(0)}$ given by

$$\Delta\boldsymbol{x}^{(0)} = \left(\boldsymbol{J}|_{\boldsymbol{x}=\boldsymbol{x}_n^{(0)}}\right)^{-1}\left(-\Psi(\boldsymbol{x}_n)|_{\boldsymbol{x}_n=\boldsymbol{x}_n^{(0)}}\right) \tag{2.39}$$

where the Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{N\times N}$ is given by

$$\boldsymbol{J}|_{\boldsymbol{x}=\boldsymbol{x}_n^{(0)}} = \boldsymbol{C} + h_n\boldsymbol{G} + h_n\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\bigg|_{\boldsymbol{x}=\boldsymbol{x}_n^{(0)}} \tag{2.40}$$

and $\Psi(\boldsymbol{x}_n)$ is defined as

$$\Psi(\boldsymbol{x}_n) = (\boldsymbol{C} + h_n\boldsymbol{G})\boldsymbol{x}_n + h_n\boldsymbol{f}(\boldsymbol{x}_n) - \boldsymbol{C}\boldsymbol{x}_{n-1} - h_n\boldsymbol{b}(t_{n-1}) \tag{2.41}$$

The convergence of N-R iteration at each step is typically fast (few N-R iterations are required) if we take the initial guess from the previous time step (i.e., $\boldsymbol{x}_n^{(0)} = \boldsymbol{x}_{n-1}$). The Jacobian matrix of the MNA formulation is typically very sparse since its sparsity pattern is identical to the sparsity pattern of the circuit MNA formulation. Thus some sparse factorization techniques can be applied to factorize the Jacobian matrix.

2.5.3 Transient Analysis of General Nonlinear Circuits using Trapezoidal Rule Method

Consider the Trapezoidal Rule method (2.17), we can approximate the vector of the values of the MNA waveforms $\boldsymbol{x}(t_1)$ at the time point t_1

$$\boldsymbol{x_1} = \boldsymbol{x_0} + \frac{1}{2}h\left(\frac{d\boldsymbol{x}(t)}{dt}\bigg|_{t=t_0} + \frac{d\boldsymbol{x}(t)}{dt}\bigg|_{t=t_1}\right) \tag{2.42}$$

To approximate $\boldsymbol{x_1}$, we apply similar steps as we did above. Using the Trapezoidal Rule to transform the set of nonlinear differential equations, at different time points, into a set of nonlinear algebraic equations by eliminating the time derivatives terms. After the rearrangement we can get the follwing system of nonlinear algebraic equations

$$(2\boldsymbol{C} + h\boldsymbol{G})\boldsymbol{x_1} + h\boldsymbol{f}(\boldsymbol{x_1}) = (2\boldsymbol{C} - h\boldsymbol{G})\boldsymbol{x_0} - h\boldsymbol{f}(\boldsymbol{x}(t_0)) + h(\boldsymbol{b}(t_0) + \boldsymbol{b}(t_1)) \tag{2.43}$$

The system of nonlinear equations in $\boldsymbol{x_1}$ can be described by the above system of equations. We assume that the values of the right side are known.

Similarly, for the generalization of the Trapezoidal Rule, we have the corresponding systems of equations as follows

$$(2\boldsymbol{C} + h_n\boldsymbol{G})\,\boldsymbol{x}_n + h_n\boldsymbol{f}(\boldsymbol{x}_n) = (2\boldsymbol{C} - h_n\boldsymbol{G})\,\boldsymbol{x}_{n-1} - h_n\boldsymbol{f}(\boldsymbol{x}(t_{n-1})) + h_n(\boldsymbol{b}(t_{n-1}) + \boldsymbol{b}(t_n)) \tag{2.44}$$

Also the Newton-Raphson method can be used to solve the above system of nonlinear algebraic equations based on a good initial guess.

2.5.4 Transient Analysis of General Nonlinear Circuits using the Obreshkov-based High-Order Method

Reference [5] presented how to apply the Obreshkov-based high-order formula into the system (2.31) to approximate $\boldsymbol{x}(t_{n+1})$ given that the approximations to $\boldsymbol{x}(t_n)$ and its high order derivatives are already known. The implementation of the high-order method leads to the augmented system of equations as follows

$$\tilde{\boldsymbol{C}}\boldsymbol{\xi}_{n+1} + \tilde{\boldsymbol{G}}\boldsymbol{\xi}_{n+1} + \tilde{\boldsymbol{\rho}}_{n+1} = \tilde{\boldsymbol{b}}_{n+1} \tag{2.45}$$

where the vector $\boldsymbol{\xi}_{n+1}$ is constructed as follows

$$\boldsymbol{\xi}_{n+1} = \begin{bmatrix} \boldsymbol{x}_{n+1}^{(0)\top} & h\boldsymbol{x}_{n+1}^{(1)\top} & \cdots & h^k\boldsymbol{x}_{n+1}^{(k)\top} \end{bmatrix}^\top \tag{2.46}$$

where $\top$ denotes the transpose operator, and $\boldsymbol{x}_{n+1}^{(i)} = [x_{1,n+1}^{(i)} \quad x_{2,n+1}^{(i)} \ldots x_{N,n+1}^{(i)}]^\top$. Here, the vector $\boldsymbol{x}(t) \in \mathbb{R}^N$ consists of N individual components, (i.e., $x_a(t), a = 1, 2, ..., N$), which may be node voltages, independent voltage sources or currents in inductors and so forth.

The matrix $\tilde{\boldsymbol{G}} \in \mathbb{R}^{(k+1)N\times(k+1)N}$ can be represented from $\boldsymbol{G}$ as follows

$$\tilde{\boldsymbol{G}} = \begin{bmatrix} \boldsymbol{G} & \mathbf{0} & \dots & & \mathbf{0} \\ \mathbf{0} & \boldsymbol{G} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{G} & \mathbf{0} \\ \alpha_0\boldsymbol{I} & \alpha_1\boldsymbol{I} & & \dots & \alpha_k\boldsymbol{I} \end{bmatrix} \tag{2.47}$$

where $\boldsymbol{I} \in \mathbb{R}^{N\times N}$ is an identity matrix of size N.

The matrix $\tilde{\boldsymbol{C}} \in \mathbb{R}^{(k+1)N\times(k+1)N}$ can be represented from $\boldsymbol{C}$ as follows

$$\tilde{\boldsymbol{C}} = \frac{1}{h}\begin{bmatrix} \mathbf{0} & \boldsymbol{C} & \dots & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{C} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{C} \\ \mathbf{0} & \mathbf{0} & \dots & \dots & \mathbf{0} \end{bmatrix} \tag{2.48}$$

The vector $\tilde{\boldsymbol{\rho}}_{n+1}$ is given by

$$\tilde{\boldsymbol{\rho}}_{n+1} = \begin{bmatrix} {\boldsymbol{\rho}_{n+1}^{(0)}}^\top & {\boldsymbol{\rho}_{n+1}^{(1)}}^\top & \cdots & {\boldsymbol{\rho}_{n+1}^{(k-1)}}^\top & \mathbf{0}^\top \end{bmatrix}^\top \tag{2.49}$$

where $\boldsymbol{\rho}_{n+1}^{(i)} = h^i \frac{\mathrm{d}^i}{\mathrm{d}t^i}\boldsymbol{f}(\boldsymbol{x}(t))$ computed at the time instance t_{n+1} .

The source vector $\tilde{\boldsymbol{b}}_{n+1}$ is constructed from the h-scaled derivatives of the independent stimuli to the circuit and the previous state variables at the time point $t = t_n$ as follows

$$\tilde{\boldsymbol{b}}_{n+1} = \begin{bmatrix} {\boldsymbol{b}_{n+1}^{(0)}}^\top & h{\boldsymbol{b}_{n+1}^{(1)}}^\top & \cdots & h^{k-1}{\boldsymbol{b}_{n+1}^{(k-1)}}^\top & \sum_{j=0}^{m}\beta_j h^j {\boldsymbol{x}_n^{(j)}}^\top \end{bmatrix}^\top \tag{2.50}$$

where $\boldsymbol{b}_{n+1}^{(i)} = \frac{\mathrm{d}^i \boldsymbol{b}(t)}{\mathrm{d}t^i}$ computed at the time instance t_{n+1}.

The iterative techniques such as Newton-Raphson (N-R) mehtod can be used to solve the system of nonlinear algebraic equations (2.45). In the N-R method, the Jacobian matrix needs to be factorized at each iteration and the corresponding Jacobian matrix can be given by

$$\tilde{\boldsymbol{J}} = \overbrace{\begin{bmatrix} \boldsymbol{G}+\boldsymbol{J}_{0,0} & \frac{1}{h}\boldsymbol{C}+\boldsymbol{J}_{0,1} & \boldsymbol{J}_{0,2} & \dots & \boldsymbol{J}_{0,k} \\ \boldsymbol{J}_{1,0} & \boldsymbol{G}+\boldsymbol{J}_{1,1} & \frac{1}{h}\boldsymbol{C}+\boldsymbol{J}_{1,2} & \dots & \boldsymbol{J}_{1,k} \\ \vdots & & \ddots & & \vdots \\ \boldsymbol{J}_{k-1,0} & \boldsymbol{J}_{k-1,1} & \dots & & \frac{1}{h}\boldsymbol{C}+\boldsymbol{J}_{k-1,k} \\ \alpha_0\boldsymbol{I} & \alpha_1\boldsymbol{I} & \dots & & \alpha_k\boldsymbol{I} \end{bmatrix}}^{(k+1)\times(k+1)\quad blocks} \tag{2.51}$$

where $\boldsymbol{J}_{i,j} = \frac{\partial \boldsymbol{f}_{n+1}^{(i)}}{\partial \boldsymbol{x}_{n+1}^{(j)}} h^{i-j}$. It is clear that algebraic nonlinearities can be shown as follows

$$\boldsymbol{J}_{i,j} = 0 \quad \text{for} \quad i < j \tag{2.52}$$

which leads to the Jacobian matrix becoming a lower block Hessenberg matrix [28].

Comparing the structure of the Jacobian matrix of the Obreshkov-based high-order method shown in (2.51) with that of the classical low-order methods shown in (2.40), it is clear that the Jacobian matrix of the high-order method is $(k+1)$ times larger than that of the classical low-order methods such as the Backward Euler method. This indicates that computational cost at each time step in the high order method is higher than that in the classical low-order methods.

But the Obreshkov-based high order integration method enables the circuit simulation to take larger step size, which leads to considerable reduction in the number of time steps required by the simulation. Thus, this can offset the computational cost increased due

to the larger size of the Jacobian matrix and can lead to the overall speedup compared to the classical low-order methods.

Chapter 3

Derivation of the Warping Error in Classical Integration Methods

3.1 Introduction to Warping Error

The Local Truncation Error (LTE) is a possible way to estimate the error introduced in a numerical integration method and it is an additive type of error measure. Reference [1] defined a new type of error measure, denoted as the warping error, where the term of the error introduced by the method appears as a multiplication term, in the sense that the exact solution can be recovered by multiplying the approximation, generated by the method, by the derived error term.

The warping error term derived in this chapter is formulated as the exponential function of another error term. The objective of this formulation is to enable investigating the effect of the error term on the eigenvalues of the circuit modeled by the differential equations.

In doing so, one can visualize the approximation to the eigenvalues of the system of DEs representing the circuit to be simulated.

It was shown in [1] that the order of the accuracy of the linear multistep (LMS) integration methods is related to the warping effect and the higher order LMS integration methods bring smaller warping error for a fixed integration time step size.

The notion of warping error hinges upon the definition of an error function, $\zeta(t)$, such that

$$x(t) = x^{\text{method}}(t)e^{\zeta^{\text{method}}(t)} \tag{3.1}$$

where $x^{\text{method}}(t)$ is the approximation generated by the integration method to $x(t)$, the exact solution to the differential equation. The derivation of $\zeta(t)$ for various integration methods is the subject of this chapter. It should be noted that the error function defined above is a multiplicative term.

In the following sections, the warping error of several classical integration methods will be discussed and we will derive the warping error expressions for these formulas based on the following test equation

$$\frac{\mathrm{d}x}{\mathrm{d}t} + \lambda x(t) = 0, \quad \lambda \in \mathcal{C}; \quad \Re(\lambda) \in \mathbb{R}^+; \quad t \geq t_0 \tag{3.2}$$

where $\Re(\lambda)$ is the real part of the eigenvalue λ and the analytical solution to the test equation (3.2) is

$$x(t) = x_0 e^{-\lambda t} \tag{3.3}$$

where $x(t_0) = x_0 \in \mathbb{R}$ represents the initial condition at the time point t_0. According to (3.3), at the time instant t_n, we have

$$x(nh) = x_0 e^{-\lambda nh} \tag{3.4}$$

3.2 Warping Error of the Backward Euler Method

Let us first consider the Backward Euler (BE) formula

$$x_{n+1}^{\mathrm{BE}} = x_n^{\mathrm{BE}} + h \left.\frac{\mathrm{d}x^{\mathrm{BE}}}{\mathrm{d}t}\right|_{t=t_{n+1}} \tag{3.5}$$

where h is the fixed time step and $t_n = nh$.

Applying the BE formula into (3.2), we have

$$\begin{aligned} x_{n+1}^{\mathrm{BE}} &= x_n^{\mathrm{BE}} - \lambda h x_{n+1}^{\mathrm{BE}} \\ &\downarrow \\ x_n^{\mathrm{BE}} &= x_{n+1}^{\mathrm{BE}}(1+\lambda h) \end{aligned} \tag{3.6}$$

which can lead to the approximation for $x(t)$ generated by BE method at $t = t_n$ in terms of the initial condition x_0 as follows

$$x_n^{\mathrm{BE}} = \frac{1}{(1+\lambda h)^n} x_0. \tag{3.7}$$

According to the definition of error function in (3.1), the error function of the BE method $\zeta^{\mathrm{BE}}(t)$ can be defined as follows

$$x(t) = x^{\mathrm{BE}}(t) e^{\zeta^{\mathrm{BE}}(t)} \tag{3.8}$$

At the time instant t_n, (3.8) can be rewitten as

$$\ln(x(nh)) = \ln(x^{\mathrm{BE}}(nh)) + \ln(e^{\zeta^{\mathrm{BE}}(nh)}) \tag{3.9}$$

Substituting (3.4) and (3.7) into (3.9), we have

$$\zeta^{\mathrm{BE}}(nh) = -n[\lambda h - \ln(1+\lambda h)] \tag{3.10}$$

The Taylor expansion of the function $\ln(1+\alpha\beta)$ is known as follows

$$\ln(1+\alpha\beta) = \sum_{i=1}^{\infty} \frac{(-1)^{i-1}(\alpha\beta)^i}{i} \tag{3.11}$$

According to (3.11), (5.1) can be rewritten as

$$\begin{aligned} \zeta^{\mathrm{BE}}(t_n) &= -n\left(\lambda h - \sum_{i=1}^{\infty} \frac{(-1)^{i-1}(\lambda h)^i}{i}\right) \\ &= \overbrace{\sum_{i=2}^{\infty} \frac{(-1)^{i-1}\lambda^i h^{i-1}}{i}}^{\psi^{\mathrm{BE}}}(nh) = \psi^{\mathrm{BE}}(nh) \end{aligned} \tag{3.12}$$

where $\psi^{\mathrm{BE}} \in \mathbb{C}$ is defined as the warping error of the Backward Euler method. It is a function of λ and h and is constant in time. ψ^{BE} does not depend on n. Here, (3.12) describes a linear relation between the error function $\zeta^{\mathrm{BE}}(t_n)$ and the discrete time points $t_n = nh$.

The above analysis indicates that (3.6) actually solves the following differential equation

$$\frac{\mathrm{d}x}{\mathrm{d}t} + (\lambda + \psi^{\mathrm{BE}})x(t) = 0, \tag{3.13}$$

instead of the original system (3.2). We can see from (3.13) that ψ^{BE} is added to λ and it warps the original eigenvalue λ of the system (3.2).

According to (3.12), $|\psi^{\mathrm{BE}}|$ is monotonic increasing w.r.t. the step size h and vanishes as a linear function of h

$$|\psi^{\mathrm{BE}}| \sim \frac{1}{2}|\lambda^2 h|, \quad \text{as} \quad h \to 0. \tag{3.14}$$

3.3 Warping Error of the Trapezoidal Rule Method

Consider the Trapezoidal Rule (TR) formula as follows

$$x_{n+1}^{\mathrm{TR}} = x_n^{\mathrm{TR}} + \frac{h}{2}\left(\left.\frac{\mathrm{d}x^{\mathrm{TR}}}{\mathrm{d}t}\right|_{t=t_n} + \left.\frac{\mathrm{d}x^{\mathrm{TR}}}{\mathrm{d}t}\right|_{t=t_{n+1}}\right) \tag{3.15}$$

Applying the TR formula into (3.2), we have

$$\begin{gathered} x_{n+1}^{\mathrm{TR}} = x_n^{\mathrm{TR}} - \lambda\frac{h}{2}(x_{n+1}^{\mathrm{TR}} + x_n^{\mathrm{TR}}) \\ \downarrow \\ x_{n+1}^{\mathrm{TR}}\left(1+\lambda\frac{h}{2}\right) = x_n^{\mathrm{TR}}\left(1-\lambda\frac{h}{2}\right) \end{gathered} \tag{3.16}$$

which can lead to the TR approximation for $x(t)$ at $t = t_n$ in terms of the initial condition x_0 as follows

$$x_n^{\mathrm{TR}} = \left(\frac{1-\lambda\frac{h}{2}}{1+\lambda\frac{h}{2}}\right)^n x_0 \tag{3.17}$$

Then, we proceed in the similar steps as with the BE method to obtain the warping error expression for the Trapezoidal Rule Method.

First, rewriting the error function in (3.1) for the TR case leads to the error function $\zeta^{\mathrm{TR}}(t)$ as follows

$$x(t) = x^{\mathrm{TR}}(t)e^{\zeta^{\mathrm{TR}}(t)} \tag{3.18}$$

At the time instant t_n, (3.18) can be rewritten as

$$\ln(x(nh)) = \ln(x^{\mathrm{TR}}(nh)) + \ln(e^{\zeta^{\mathrm{TR}}(nh)}) \tag{3.19}$$

Substituting (3.4) and (3.17) into (3.19), we have

$$\zeta^{\mathrm{TR}}(nh) = n\left[\lambda h - \ln\left(1-\lambda\frac{h}{2}\right) + \ln\left(1+\lambda\frac{h}{2}\right)\right] \tag{3.20}$$

and then

$$\zeta^{\mathrm{TR}}(t_n) = n\left(\lambda h - \sum_{i=1}^{\infty}\frac{(-1)^{i-1}(-\lambda\frac{h}{2})^i}{i} + \sum_{i=1}^{\infty}\frac{(-1)^{i-1}(\lambda\frac{h}{2})^i}{i}\right) \tag{3.21}$$

and finally we have

$$\zeta^{\mathrm{TR}}(t_n) = \overbrace{\sum_{i=2}^{\infty}\left(\frac{\lambda}{2}\right)^{2i-1}\frac{2h^{2(i-1)}}{2i-1}}^{\psi^{\mathrm{TR}}}(nh) \tag{3.22}$$

From the above analysis, it can be seen that the warping error ψ^{TR} vanishes as a quadratic function of h

$$|\psi^{\mathrm{TR}}| \sim \frac{1}{12}|\lambda^3 h^2|, \quad \text{as} \quad h \to 0. \tag{3.23}$$

thus the Trapezoidal rule method introduces smaller warping error compared with that of the Backward Euler method shown in (3.14) [29].

It can be seen from the expression of the TR warping error (3.22) that the exponent term of the eigenvalue λ is always odd, which means that the TR method does not introduce real warping error parts in pure imaginary eigenvalues. By contrast, the warping error of BE has real parts due to the polynomial (3.12).

3.4 Warping Error of the General LMS Methods

After demonstrating the warping errors of the BE and TR methods, in the section, the warping error of the general LMS methods will be discussed, in particular we focus on the Gear's methods.

Reference [1] defined the warping error function of the LMS methods as $\zeta^{\mathrm{LMS}}(t)$ as

follows

$$x(t) = x^{\mathrm{LMS}}(t)e^{\zeta^{\mathrm{LMS}}(t)} \tag{3.24}$$

Let us consider the general form of the LMS methods [18], which can be described as follows

$$\sum_{i=0}^{k} \alpha_i x(t_n + (k-i)h)) - h\sum_{i=0}^{k} \beta_i \frac{dx}{dt}(t_n + (k-i)h)) = 0 \tag{3.25}$$

where h is the fixed time step size, k is the order of the LMS method and α_i and β_i are the coefficients that determine the method [16] [3]. Some coefficients may be zero in some LMS methods.

Applying the general LMS formula to (3.2) and the solution to (3.2) at the time point $t = t_n$ is supposed being affected by the warping error ψ^{LMS} as follows

$$x^{\mathrm{LMS}}(t) = x_0 e^{-(\lambda+\psi^{\mathrm{LMS}})nh} \tag{3.26}$$

The substitution of $x^{\mathrm{LMS}}(t)$ in (3.2) and the implementation of the general LMS methods in (3.2) lead to

$$x_0 e^{-(\lambda+\psi^{\mathrm{LMS}})(n+k)h} \sum_{i=0}^{k} (\alpha_i - h\lambda\beta_i) e^{(\lambda+\psi^{\mathrm{LMS}})ih} = 0 \tag{3.27}$$

where the multiplicative term $x_0 e^{-(\lambda+\psi^{\mathrm{LMS}})(n+k)h} \neq 0, \forall(\lambda+\psi^{\mathrm{LMS}}) \in \mathbb{C}$ can be eliminated, and then we have

$$\sum_{i=0}^{k} (\alpha_i - h\lambda\beta_i) e^{(\lambda+\psi^{\mathrm{LMS}})ih} = 0 \tag{3.28}$$

Introducing a new variable $z(h)$ as follows

$$z(h) = e^{(\lambda+\psi^{\mathrm{LMS}})h} \tag{3.29}$$

So we can simplify (3.28) to

$$\sum_{i=0}^{k} (\alpha_i - h\lambda\beta_i) z^i(h) = 0 \tag{3.30}$$

which is a polynomial equation of order k in terms of the new variable $z(h)$ and has k roots. It is clear that the warping error is related to the roots of the above equation. Reference [1] proved that the warping error for any LMS method is unique in spite of the k roots that the equation has.

Let us start the derivation of the warping error ψ^{LMS} of the LMS methods.

First considering the following equation

$$\begin{aligned} z &= e^{-(\lambda+\psi^{\text{LMS}})h} \\ &= e^{-\lambda h} e^{-\psi^{\text{LMS}} h} \end{aligned} \tag{3.31}$$

After Taylor expansion, we have

$$\begin{aligned} z =& \sum_{m=0}^{\infty} \frac{(-\lambda h)^m}{m!} \cdot \sum_{m=0}^{\infty} \varrho_m \frac{(-\lambda h)^m}{m!} \\ =& \left(1 - \lambda h + \frac{(\lambda h)^2}{2!} - \frac{(\lambda h)^3}{3!} + \cdots\right) \\ & \cdot \left(\varrho_0 - \varrho_1 \lambda h + \varrho_2 \frac{(\lambda h)^2}{2!} - \varrho_3 \frac{(\lambda h)^3}{3!} + \cdots\right) \\ =& \varrho_0 - (\varrho_0 + \varrho_1)\lambda h + \left(\frac{\varrho_0}{2!} + \varrho_1 + \frac{\varrho_2}{2!}\right)(\lambda h)^2 \\ & - \left(\frac{\varrho_0}{3!} + \frac{\varrho_1}{2!} + \frac{\varrho_2}{2!} + \frac{\varrho_3}{3!}\right)(\lambda h)^3 + \cdots \end{aligned} \tag{3.32}$$

where $\varrho_m \in \mathcal{C}$ represents any possible dependence of ψ^{LMS} on λh. Assuming $e^{-\psi^{\text{LMS}} h}$ is expandable in Taylor series. Reference [1] merged these two series in one by introducing a new variable ρ_m and by appropriately grouping some coefficients,

$$z = \sum_{m=0}^{\infty} (1 + \rho_m) \frac{(-\lambda h)^m}{m!} \tag{3.33}$$

where the ρ_m terms represent the combinations of the different ϱ_m terms. It can be seen from the above equation that $z(0) = 1$, which means $\rho_0 = 0$.

It has been proved that

$$\rho_m = \begin{cases} 0 & \text{for } 0 \leq m \leq k \\ -1-(k+1)\frac{\sum_{i=0}^{k}\beta_i i^k}{\sum_{i=0}^{k}\alpha_i i^{(k+1)}} & \text{for } m = k+1 \\ -1-(\rho_{m-1}+1)m\frac{\sum_{i=0}^{k}\beta_i i^{m-1}}{\sum_{i=0}^{k}\alpha_i i^m} & \text{for } m \geq k+2 \end{cases} \tag{3.34}$$

and that

$$\varrho_m = \begin{cases} 1 & \text{for } m = 0 \\ 0 & \text{for } m \in \{1, 2, ..., k\} \\ \rho_{k+1} & \text{for } m = k+1 \end{cases} \tag{3.35}$$

where k represents the order the LMS integration method and the above results indicates that ψ^{LMS} depends on the order of the method.

According to (3.34), examples of value ρ_{k+1} for some classical integration methods are as follows

$$\text{Backward Euler method}: \quad \rho_2^{BE} = -1 - 2\frac{0}{-1} = -1 \tag{3.36}$$

$$\text{Trapezoidal Rule method}: \quad \rho_2^{TR} = -1 - 2\frac{0+1}{0-2} = 0 \tag{3.37}$$

$$\text{Trapezoidal Rule method}: \quad \rho_3^{TR} = -1 - 3\frac{0+1}{0-2} = \frac{1}{2} \tag{3.38}$$

$$\text{Gear's method of order 2}: \quad \rho_3^{Gear2} = -1 - 3\frac{0+0}{0+2-4} = -1 \tag{3.39}$$

We derive the warping error expressions for Gear's methods based on the above warping error analysis of the LMS methods.

Let us consider several warping error examples of the Gear's method, and the general form of Gear's method of order k can be represented as follows

$$\sum_{i=0}^{k} \alpha_i x_{n+i} = h\beta_0 x'_{n+k} \tag{3.40}$$

where the coefficients α_i and β_0 in (3.25) are chosen such that the Gear's method achieves order k.

Gear's Method of Order 2 According to (3.31) and (3.32), we have

$$\begin{aligned} e^{-\psi^{\mathrm{LMS}}h} &= \sum_{m=0}^{\infty} \varrho_m \frac{(-\lambda h)^m}{m!} \\ &= \varrho_0 - \varrho_1 \lambda h + \varrho_2 \frac{(\lambda h)^2}{2!} - \varrho_3 \frac{(\lambda h)^3}{3!} + \cdots \end{aligned} \tag{3.41}$$

For Gear of order 2 case, (3.41) can be rewritten as

$$e^{-\psi^{\mathrm{GEAR2}}h} \approx \varrho_0 - \varrho_1 \lambda h + \varrho_2 \frac{(\lambda h)^2}{2!} - \varrho_3 \frac{(\lambda h)^3}{3!} + \varrho_4 \frac{(\lambda h)^4}{4!} - \varrho_5 \frac{(\lambda h)^5}{5!} \tag{3.42}$$

where the terms that are higher than order 5 are very small and can be ignored. Some calculations based on (3.34) and (3.35) can lead to

$$\begin{aligned} \varrho_0 &= 1 \\ \varrho_1 &= \varrho_2 = 0 \\ \varrho_3 &= -1 \\ \varrho_4 &= 3 \\ \varrho_5 &= -6 \end{aligned} \tag{3.43}$$

Substituting (3.43) into (3.42), we have

$$e^{-\psi^{\mathrm{GEAR2}}h} = 1 + \frac{(\lambda h)^3}{6} + \frac{(\lambda h)^4}{8} + \frac{(\lambda h)^5}{20} \tag{3.44}$$

then (3.44) can be rewritten as

$$\ln e^{-\psi^{\mathrm{GEAR2}}h} = -\psi^{\mathrm{GEAR2}}h = \ln\left(1 + \frac{(\lambda h)^3}{6} + \frac{(\lambda h)^4}{8} + \frac{(\lambda h)^5}{20}\right) \tag{3.45}$$

Proceeding in the similar steps as with BE in (5.1), (3.11) and (3.12) lead to the final warping error expression of the Gear's method of order 2 as follows

$$\psi^{\mathrm{GEAR2}} = \sum_{i=1}^{\infty} \frac{(-1)^i \left[\frac{(\lambda h)^3}{6} + \frac{(\lambda h)^4}{8} + \frac{(\lambda h)^5}{20}\right]^i}{ih} \tag{3.46}$$

Gear's Method of Order 3 Proceeding in the sme steps as with Gear's Method of Order 2 ends up with

$$\psi^{\mathrm{GEAR3}} = \sum_{i=1}^{\infty} \frac{(-1)^i \left[\frac{-1}{24}(\lambda h)^4 + \frac{-1}{30}(\lambda h)^5\right]^i}{ih} \tag{3.47}$$

Gear's Method of Order 4 Similarly as above, we have

$$\psi^{\mathrm{GEAR4}} = \sum_{i=1}^{\infty} \frac{(-1)^i \left[\frac{(\lambda h)^5}{5!} + \frac{5}{6!}(\lambda h)^6\right]^i}{ih} \tag{3.48}$$

3.5 Discussion

This chapter presented the derivation of the warping error in numerical integration methods utilized by SPICE-based simulators [30], such as the Backward Euler method, the Trapezoidal Rule method and the Gear's methods. This new type of error measure copes with the error accumulation at each integration time step in the numerical integration process and the warping error term is formulated as a multiplicative term in order to visualize the perturbation of the eigenvalues of the system of differential equations modelling the circuits to be simulated.

Chapter 4

Derivation of the Warping Error in Obreshkov-based High-Order Integration Methods

Compared with the classical low-order integration methods discussed in the previous chapter, such as the Backward Euler method, the Trapezoidal Rule method and the Gear's methods, the new A-stable and L-stable Obreshkov-based high-order integration method is constructed by including information from higher-order derivatives. This chapter presents the derivation of the warping error in the Obreshkov-based high-order integration methods.

Recall the Obreshkov-based high-order formula as follows

$$\sum_{i=0}^{k} \alpha_i h^i x_{n+1}^{(i)} = \sum_{i=0}^{m} \beta_i h^i x_n^{(i)} \tag{4.1}$$

where h is the time step size and $t_n = nh$, and

$$\alpha_i = (-1)^i \frac{(m+k-i)!}{(m+k)!} \frac{k!}{i!(k-i)!} \tag{4.2}$$

$$\beta_i = \frac{(m+k-i)!}{(m+k)!}\frac{m!}{i!(m-i)!} \tag{4.3}$$

Applying the high-order formula into the test equation $\mathrm{d}x/\mathrm{d}t + \lambda x(t) = 0$ in (3.2) leads to

$$x_{n+1}^{\text{High}} = \frac{\beta_0 + \beta_1(-\lambda h)^1 + \beta_2(-\lambda h)^2 + \cdots + \beta_m(-\lambda h)^m}{\alpha_0 + \alpha_1(-\lambda h)^1 + \alpha_2(-\lambda h)^2 + \cdots + \alpha_k(-\lambda h)^k} \cdot x_n^{\text{High}} \tag{4.4}$$

then the approximate solution to the scalar system (3.2) at $t = t_n$ in terms of the initial condition x_0 is as follows

$$x_n^{\text{High}} = \left(\frac{\beta_0 + \beta_1(-\lambda h)^1 + \beta_2(-\lambda h)^2 + \cdots + \beta_m(-\lambda h)^m}{\alpha_0 + \alpha_1(-\lambda h)^1 + \alpha_2(-\lambda h)^2 + \cdots + \alpha_k(-\lambda h)^k}\right)^n \cdot x_0 \tag{4.5}$$

Let us introduce two new variables A and B as follows

$$B = \beta_0 + \beta_1(-\lambda h)^1 + \beta_2(-\lambda h)^2 + \cdots + \beta_m(-\lambda h)^m \tag{4.6}$$

and,

$$A = \alpha_0 + \alpha_1(-\lambda h)^1 + \alpha_2(-\lambda h)^2 + \cdots + \alpha_k(-\lambda h)^k \tag{4.7}$$

So we can simplify (4.5) to

$$x_n^{\text{High}} = \left(\frac{B}{A}\right)^n x_0 \tag{4.8}$$

According to the definition of error function in (3.1), the error function of the modified Obreshkov high-order method $\zeta^{\text{High}}(t)$ can be defined as follows

$$x(t) = x^{\text{High}}(t)e^{\zeta^{\text{High}}(t)} \tag{4.9}$$

At the time instant t_n, by recalling that $x(nh) = x_0 e^{-\lambda nh}$, and the substitution of x_n^{High} in (4.9) lead to

$$\zeta^{\text{High}}(nh) = -n(\lambda h + \ln B - \ln A) \tag{4.10}$$

Expanding $\ln(A)$ and $\ln(B)$ in Taylor series similar to BE and TR discussed above, we can have

$$\zeta^{\text{High}}(nh) = -n\left\{\lambda h + \sum_{j=1}^{\infty} \frac{(-1)^{j-1}}{j}\left[(B-1)^j - (A-1)^j\right]\right\} \tag{4.11}$$

since $\alpha_0 = 1$ and $\beta = 1$, (4.11) can be written as

$$\zeta^{\text{High}}(nh) = -n\left\{\lambda h + \sum_{j=1}^{\infty} \frac{(-1)^{j-1}}{j}\left(\left[\sum_{i=1}^{m}\beta_i(-\lambda h)^i\right]^j - \left[\sum_{i=1}^{k}\alpha_i(-\lambda h)^i\right]^j\right)\right\} \tag{4.12}$$

Let us consider several examples of the Obreshkov-based high-order integration formulas.

4.1 High-Order Method of Order 2 ($k = 1, m = 1$)

The substitution of $k = 1 \quad \text{and} \quad m = 1$ in (4.7) and (4.6) respectively lead to

$$A = 1 + \frac{\lambda h}{2}, \quad B = 1 - \frac{\lambda h}{2} \tag{4.13}$$

Substituting (4.13) into (4.11) lead to

$$\zeta^{\text{High}}(nh) = -n\left(\lambda h + \sum_{j=1}^{\infty} \frac{(-1)^{j-1}(-\lambda\frac{h}{2})^j}{j} - \sum_{j=1}^{\infty} \frac{(-1)^{j-1}(\lambda\frac{h}{2})^j}{j}\right), \tag{4.14}$$

then we have

$$\zeta^{\text{High}}(nh) = \overbrace{\sum_{i=2}^{\infty}\left(\frac{\lambda}{2}\right)^{2i-1}\frac{2h^{2(i-1)}}{2i-1}}^{\psi^{\text{High}},\quad if\quad k=1,m=1}(nh) \tag{4.15}$$

which is the same as TR method in (3.22). Thus, the order of the high-order method is $(k + m) = 2$. The error function approximation of the high order method of order 2 (when $k = 1$ and $m = 1$) as follows

$$\zeta^{\text{High}}(nh) \approx -n\left\{\frac{1}{12}(-\lambda h)^3 + \frac{1}{80}(-\lambda h)^5 + \frac{1}{448}(-\lambda h)^7 + \frac{1}{2304}(-\lambda h)^9 + \cdots\right\} \tag{4.16}$$

where the higher order terms are very small and can be ignored.

4.2 High-Order Method of Order 3 ($k = 2, m = 1$)

Proceeding in the same steps as above, we can have

$$\begin{aligned} B &= \beta_0 + \beta_1(-\lambda h) = 1 + \frac{1}{3}(-\lambda h) \\ A &= \alpha_0 + \alpha_1(-\lambda h) + \alpha_2(-\lambda h)^2 = 1 + \frac{-2}{3}(-\lambda h) + \frac{1}{6}(-\lambda h)^2 \end{aligned} \tag{4.17}$$

then substituting (4.17) into (4.11) lead to

$$\zeta^{\text{High}}(nh) = -n\left(\lambda h + \sum_{j=1}^{\infty} \frac{(-1)^{j-1}}{j}\left([\frac{1}{3}(-\lambda h)]^j - [\frac{-2}{3}(-\lambda h) + \frac{1}{6}(-\lambda h)^2]^j\right)\right) \tag{4.18}$$

then we have the error function approximation of the high order method of order 3 (when $k = 2$ and $m = 1$) as follows

$$\zeta^{\text{High}}(nh) \approx -n\left\{-\frac{1}{72}(-\lambda h)^4 - \frac{1}{270}(-\lambda h)^5 - \frac{1}{648}(-\lambda h)^6 - \frac{1}{6804}(-\lambda h)^7 + \cdots\right\} \tag{4.19}$$

where the higher order terms are very small and can be ignored. It can be seen from (4.19) that the order of the high-order method is $(k + m) = 3$.

4.3 High-Order Method of Order 4 ($k = 2, m = 2$)

Proceeding in the same steps as above, we can have

$$\begin{aligned} B &= \beta_0 + \beta_1(-\lambda h) + \beta_2(-\lambda h)^2 = 1 + \frac{1}{2}(-\lambda h) + \frac{1}{12}(-\lambda h)^2 \\ A &= \alpha_0 + \alpha_1(-\lambda h) + \alpha_2(-\lambda h)^2 = 1 + \frac{-1}{2}(-\lambda h) + \frac{1}{12}(-\lambda h)^2 \end{aligned} \tag{4.20}$$

then substituting (4.20) into (4.11) lead to

$$\zeta^{\text{High}}(nh) = -n\left(\lambda h + \sum_{j=1}^{\infty} \frac{(-1)^{j-1}}{j}\left([\frac{1}{2}(-\lambda h) + \frac{1}{12}(-\lambda h)^2]^j - [\frac{-1}{2}(-\lambda h) + \frac{1}{12}(-\lambda h)^2]^j\right)\right) \tag{4.21}$$

then we have the error function approximation of the high order method of order 4 (when $k = 2$ and $m = 2$) as follows

$$\zeta^{\text{High}}(nh) \approx -n\left\{-\frac{1}{720}(-\lambda h)^5 - \frac{1}{12096}(-\lambda h)^7 + \frac{11}{2737152}(-\lambda h)^{11} + \frac{13}{38817792}(-\lambda h)^{13} + \cdots\right\} \tag{4.22}$$

where the higher order terms are very small and can be ignored. It can be seen from (4.22) that the order of the high-order method is $(k + m) = 4$.

4.4 High-Order Method of Order 5 ($k = 3, m = 2$)

Proceeding in the same steps as above, we can have

$$\begin{aligned} B - 1 &= \frac{2}{5}(-\lambda h) + \frac{1}{20}(-\lambda h)^2 \\ A - 1 &= \frac{-3}{5}(-\lambda h) + \frac{3}{20}(-\lambda h)^2 + \frac{-1}{60}(-\lambda h)^3 \end{aligned} \tag{4.23}$$

then substituting (4.23) into (4.11) lead to

$$\zeta^{\text{High}}(nh) = -n\left(\lambda h + \sum_{j=1}^{\infty} \frac{(-1)^{j-1}}{j}\left([\frac{2}{5}(-\lambda h) + \frac{(-\lambda h)^2}{20}]^j - [\frac{-3}{5}(-\lambda h) + \frac{3}{20}(-\lambda h)^2 + \frac{-(-\lambda h)^3}{60}]^j\right)\right) \tag{4.24}$$

and finally we can have the error function approximation of the high order method of order 5 (when $k = 3$ and $m = 2$) as follows

$$\zeta^{\text{High}}(nh) \approx -n\left\{\frac{1}{7200}(-\lambda h)^6 + \frac{1}{42000}(-\lambda h)^7 + \frac{1}{120000}(-\lambda h)^8 + \frac{1}{1012500}(-\lambda h)^9 + \cdots\right\} \tag{4.25}$$

where the higher order terms are very small and can be ignored. It can be seen from (4.25) that the order of the high-order method is $(k + m) = 5$.

4.5 High-Order Method of Order 6 ($k = 3, m = 3$)

Proceeding in the same steps as above, we can have

$$\begin{aligned} B - 1 &= \beta_1(-\lambda h) + \beta_2(-\lambda h)^2 + \beta_3(-\lambda h)^3 = \frac{1}{2}(-\lambda h) + \frac{1}{10}(-\lambda h)^2 + \frac{1}{120}(-\lambda h)^3 \\ A - 1 &= \alpha_1(-\lambda h) + \alpha_2(-\lambda h)^2 + \alpha_3(-\lambda h)^3 = \frac{-1}{2}(-\lambda h) + \frac{1}{10}(-\lambda h)^2 + \frac{-1}{120}(-\lambda h)^3 \end{aligned} \tag{4.26}$$

substituting (4.26) into (4.11) lead to

$$\begin{aligned} \zeta^{\text{High}}(nh) = -n\Bigg[\lambda h + \sum_{j=1}^{\infty} \frac{(-1)^{j-1}}{j}\Bigg(\Big[\frac{(-\lambda h)}{2} + \frac{(-\lambda h)^2}{10} + \frac{(-\lambda h)^3}{120}\Big]^j \\ - \Big[\frac{-(-\lambda h)}{2} + \frac{(-\lambda h)^2}{10} + \frac{-(-\lambda h)^3}{120}\Big]^j\Bigg)\Bigg] \end{aligned} \tag{4.27}$$

Then we can have the error function approximation of the high order method of order 6 (when $k = 3$ and $m = 3$) as follows

$$\zeta^{\text{High}}(nh) \approx -n\left\{\frac{(-\lambda h)^7}{100800} + \frac{(-\lambda h)^9}{2592000} + \frac{(-\lambda h)^{11}}{190080000} + \frac{(-\lambda h)^{13}}{6739200000} + \cdots\right\} \tag{4.28}$$

where the higher order terms are very small and can be ignored. It can be seen from (4.28) that the order of the high-order method is $(k + m) = 6$.

As mentioned earlier in chapter 3 and in the warping error expression of TR method (is the same as the high-order method of order 2 with $k = m = 1$) shown in (4.16) that the exponent term of the eigenvalue λ is always odd. This means that the high-order method of order 2 does not introduce real warping error parts in pure imaginary

eigenvalues. If looking into (4.22) and (4.28), it can be seen that the high-order method of order 4 (with $k = m = 2$) and the high-order method of order 6 (with $k = m = 3$) also have the same property. In contrast, the real parts of the warping error exist in the high-order method of order 3 (with $k = 2$ and $m = 1$) and in the high-order method of order 5 (with $k = 3$ and $m = 2$), and this can be shown in (4.19) and (4.25).

4.6 Discussion

This chapter presented the derivation of the warping error in the Obreshkov-based high-order integration methods. The selection of k and m in the high-order integration formula determines the order and the stability characteristics of the method itself. Reference [5] proved that this new high-order integration method can approximate the solution of the differential equations up to the order $(k + m)$ and that the numerical stability condition of the high-order integration method is $k - 2 \leq m \leq k$. All the cases discussed above are under this numerical stability condition. The numerical results will be presented to compare the warping errors of these high-order methods and the warping errors of the classical low-order methods in the next chapter, such as the Backward Euler, the Trapezoidal Rule and the Gear's methods.

Chapter 5

Derivation of Shooting Method Based on the High Order Integration Methods

We have derived the warping error expressions for the classical low-order integration methods, such as BE, TR and the Gear's methods, and the A-stable and L-stable Obreshkov-based high-order integration methods in Chapter 3 and Chapter 4, respectively. This chapter compares the warping error of the high-order methods against the warping error of the classical low-order methods. It will be shown that the warping error of the high-order methods is smaller than that of the low-order methods. In order to inspect how the smaller warping error in the high-order methods can affect the general circuit simulation tasks, say steady state analysis using the shooting method, we will derive the shooting method based on the high-order integration methods later in this chapter.

5.1 Numerical Comparison between Warping Error in High-Order and Low-Order Methods

The main objective of this section is to compare the values of the warping error in different integration methods such as BE, TR, Gear methods of order 2, 3, 4 and high-order methods of order 2, 3, 4, 5, 6, respectively. Note that high-order method of order 2 is actually equivalent to the TR.

As discussed in the previous chapters, $\psi^{\text{Method}} \in \mathbb{C}$ is the so-called the warping error of an integration method and it is a function of the system eigenvalue λ and time step size h. ψ^{Method} is constant in time. The linear test funciton $\mathrm{d}x/\mathrm{d}t + \lambda x(t) = 0$ in (3.2) has been used to derive the warping error expressions ψ^{Method} for the classical low order methods in Chapter 3, such as BE, TR and the Gear's methods and has been used to derive the warping error function expressions ζ^{Method} for the Obreshkov-based high-order methods, say high-order method of order 2, 3, 4, 5 and 6. It should be noted that the following equation describes a simple linear relation between the warping error ψ^{Method} and the warping error function $\zeta^{\text{Method}}(t_n)$

$$\zeta^{\text{Method}}(t_n) = \psi^{\text{Method}} \times (nh) \tag{5.1}$$

where the discrete time points $t_n = nh$. Therefore, the values of the warping error in the high-order methods can be easily obtained based on the relation above.

As mentioned above, the warping error $\psi^{\text{Method}} \in \mathbb{C}$ in any integration method is a function of the system eigenvalue λ and time step size h. Let us first consider a system with a purely imaginary eigenvalue case, say $\lambda = -j$. The imaginary and the real parts of the warping error results in different integration methods are shown in Figure 5.1 and Figure 5.2, respectively. The values of the warping error are computed repeatedly for $0.001 \leq h \leq 0.1$.

In both figures, the dashed line represents the BE method, the solid lines represent the Gear's methods of order 2, 3, 4 (GEAR2, GEAR3, GEAR4, respectively), and the dotted lines represnet the Obreshkov-based high-order methods of order 2 (same as TR), 3, 4, 5, 6, respectively.

It can be seen in Figure 5.1 that the Obreshkov-based high-order methods of order 3, 4, 5, 6 outperform the Gear's methods of order 3, 4 and that the Obreshkov-based high-order methods of order 2 (same as TR) outperforms the Gear's method of order 2. Here outperform is to be understood as producing much lower warping error. In Gear's methods, the higher order methods introduce smaller warping error. Similarly, in the Obreshkov-based high-order methods, higher order methods always perform better. It shows that the BE method introduces the largest warping error compared with all the other methods.

Figure 5.2 shows that the high-order method of order 2, 4, 6, respectively, do not introduce real warping-error components for the pure imaginary eigenvalue, i.e., $\lambda = -j$. This is because the exponent of the λ term of the warping error expressions in these methods are always odd, which can be seen in (4.16), (4.22) and (4.28). By contrast, Figure 5.2 shows that the real parts of the warping error exist in the high-order method of order 3 and 5. Because the warping error expressions in these two methods have both odd and even power terms, which are shown in(4.19) and (4.25).

It should be noted that all these methods, except the Gear's methods of order 3 and 4, are all A-stable.

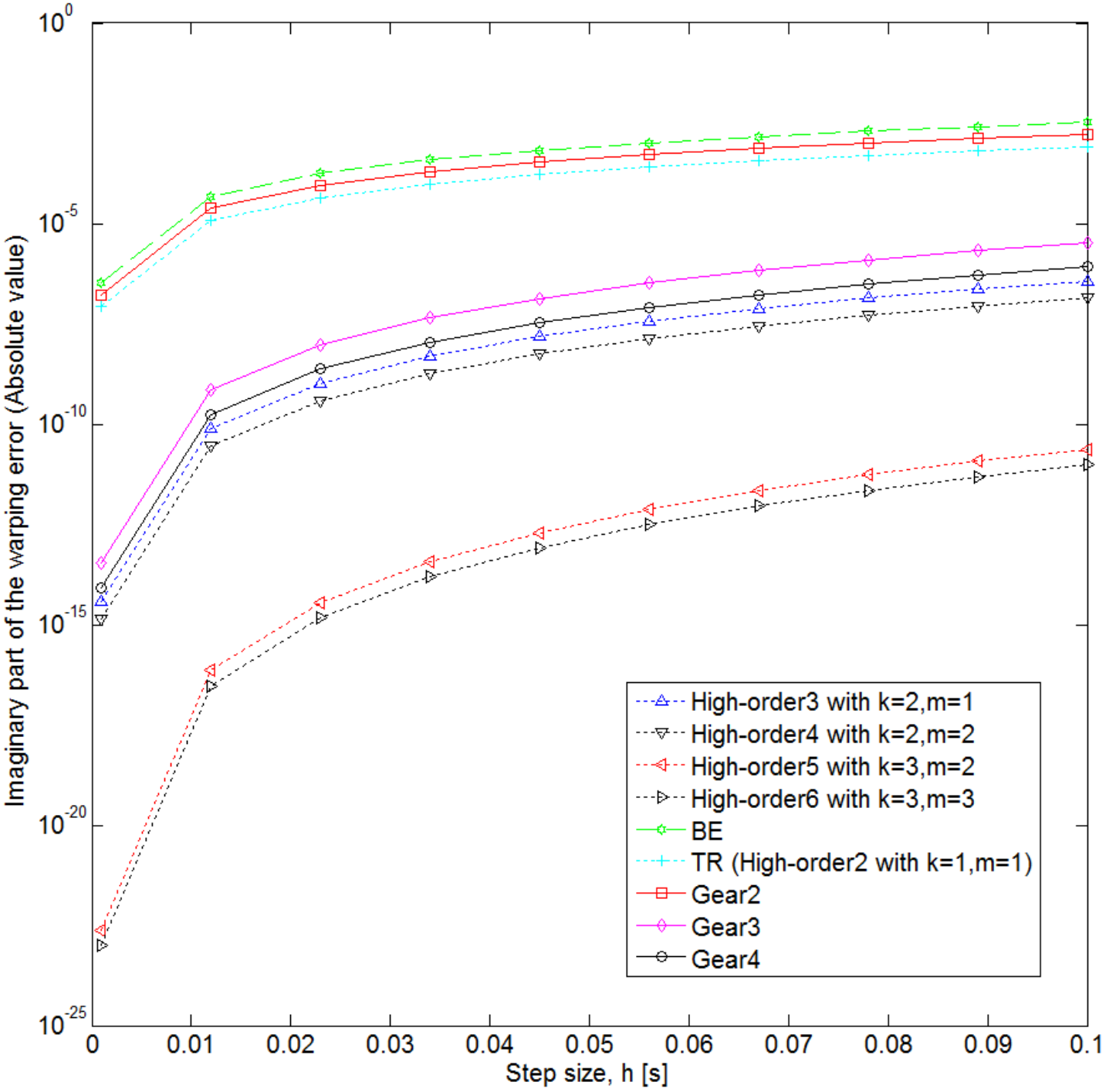

Figure 5.1: Imaginary part of warped eigenvalue -j

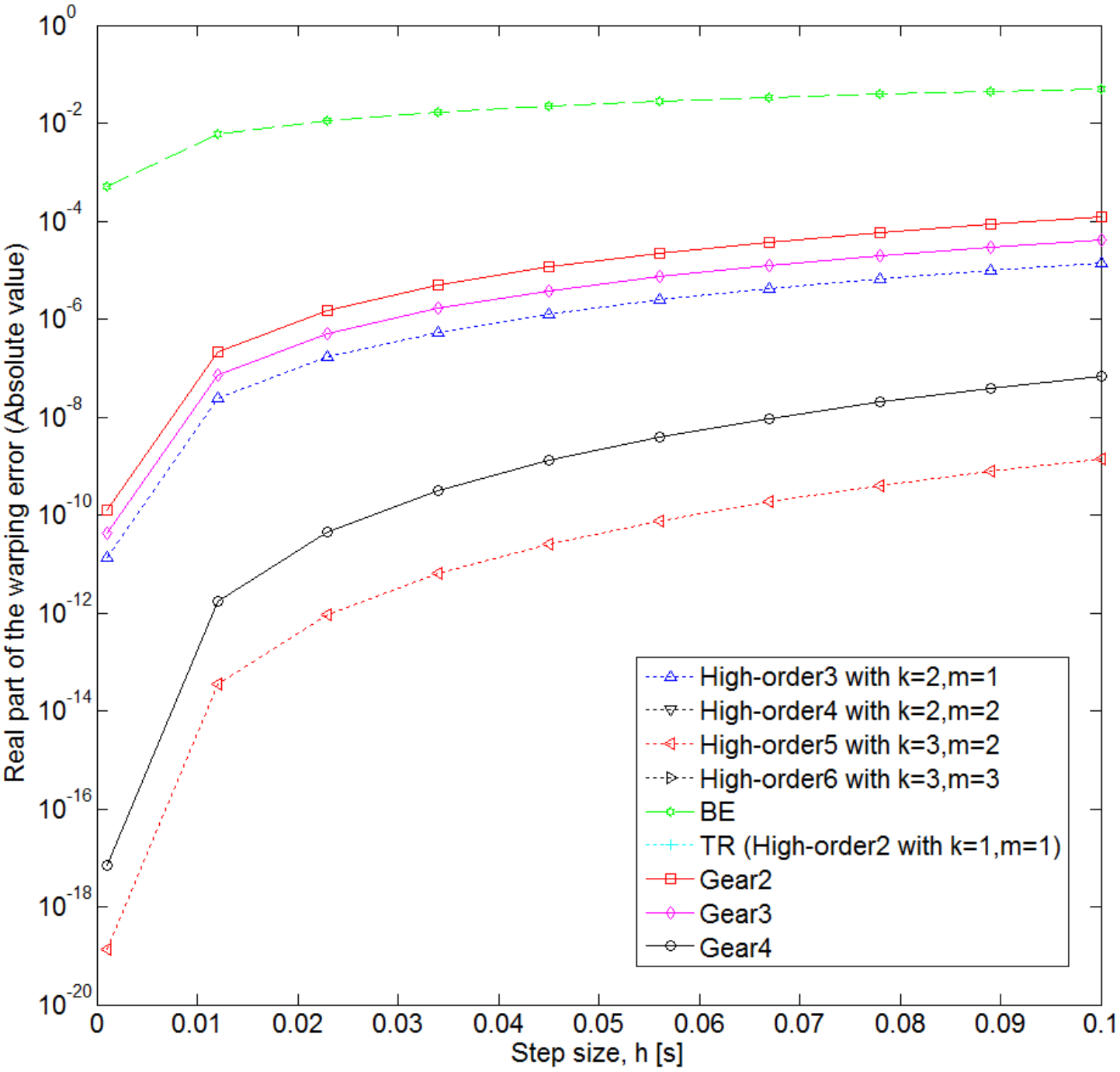

Figure 5.2: Real part of warped eigenvalue -j

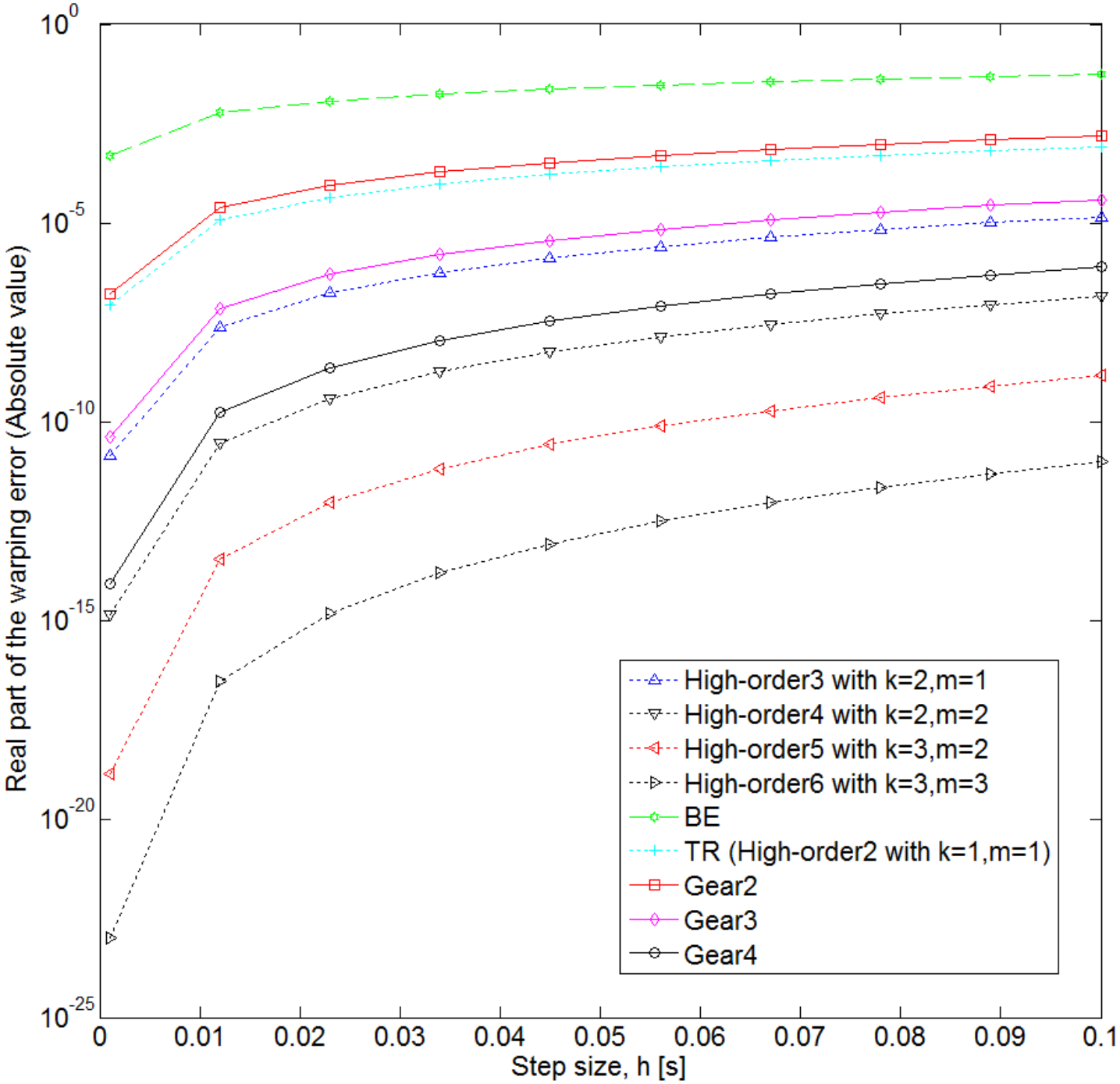

Figure 5.3: Real part of warped eigenvalue -1

We also consider a real eigenvalue $\lambda = -1$ case to compare the values of the warping error in different integration methods as shown in Figure 5.3 above. In this case, it is obvious that the values of the warping error are real for all the integration methods.

Figure 5.3 shows that the Obreshkov-based high-order methods of order 4, 5 and 6 outperform the Gear's methods of order 4 and that the Obreshkov-based high-order method of order 3 outperforms the Gear's method of order 3 as well as that the Obreshkov-based high-order method of order 2 (same as TR) outperforms the Gear's method of order 2. Within both the Gear's methods and the Obreshkov-based high-order methods, the higher order methods always introduce less warping error. Again, the BE method is showed to introduce the largest warping error compared with all the other higher order integration methods.

5.2 Implementation of Shooting Method Based on the High-Order Integration Methods

It can be seen from the previous section that the warping error in the Obreshkov-based high-order integration methods is much smaller than the warping error in the classical low-order methods, such as the Backward Euler, the Trapezoidal Rule and the Gear's methods. In order to take advantage that the warping error in the high-order methods is smaller, this section presents a new way to implement the shooting method by using the Obreshkov-based high-order methods (high-order shooting method).

5.2.1 Introduction to the Shooting Method

The main objective in the shooting method [31] [32] is to search for a suitable set of initial conditions that can bypass the long transient phase in finding the steady-state response in circuit simulation. Reconsider the following system that describes a general

nonlinear linear circuit in the time-domain using the MNA formulation

$$\boldsymbol{G}\boldsymbol{x}(t) + \boldsymbol{C}\frac{d\boldsymbol{x}(t)}{dt} + \boldsymbol{f}(\boldsymbol{x}(t)) = \boldsymbol{b}(t) \tag{5.2}$$

Assume that the vector $\boldsymbol{x}_0^*$ is the optimum set of initial conditions, which when used as the initial conditions for $\boldsymbol{x}(t)$ at $t = t_0$, there will be no transient phase and the circuit can directly enter into the periodic steady-state. In other words, if transient analysis is carried out with the initial conditions of $\boldsymbol{x}(0) = \boldsymbol{x}_0^*$, and continued for a period of time up to $t = T$, where T represents the common period of the circuit periodic excitation sources, then we can get

$$\boldsymbol{x}(T) = \boldsymbol{x}(0) = \boldsymbol{x}_0^* \tag{5.3}$$

Introducing the concept of State-Transition Function (STF), around which the main idea of shooting method is built. The STF is defined for a system described by differential equations as a function that takes as input the values of the system state at certain time point, such as $t = t_0$, i.e., $\boldsymbol{x}(t_0)$, and a period of time ΔT and yields its output, the system state at time point $t = t_0 + \Delta T$, i.e., $\boldsymbol{x}(t_0 + \Delta T)$. Here, the system state represents the MNA variables.

Figure 5.4 illustrates the concept of the STF.

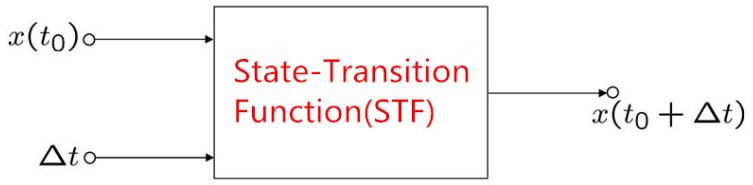

Figure 5.4: Graphical description of the notion of STF

The STF can be described by the notation $\boldsymbol{\phi}(\boldsymbol{x}_0, t_0, \Delta T)$, where $\boldsymbol{x}_0 = \boldsymbol{x}_0(t_0)$. Based on

the definition of STF, we have

$$\boldsymbol{x}(t_0 + \Delta T) = \boldsymbol{\phi}(\boldsymbol{x}_0, t_0, \Delta T) \tag{5.4}$$

Employing the definition of STF in (5.4), (5.3) can be rewritten as

$$\boldsymbol{\phi}(\boldsymbol{x}_0^*, t_0, \Delta T) = \boldsymbol{x}_0^* \tag{5.5}$$

Therefore, the optimum initial condition $\boldsymbol{x}_0^*$ can be regarded as the solution vector for $\boldsymbol{x}_0$, which satisfies the following nonlinear system of equations,

$$\boldsymbol{\psi}(\boldsymbol{x}_0) \equiv \boldsymbol{\phi}(\boldsymbol{x}_0, t_0, T) - \boldsymbol{x}_0 = 0 \tag{5.6}$$

Solving the above system of nonlinear equations (5.6) can be achieved through using some iterative techniques, such as the Newton-Raphson (N-R) method, and it is called the Newton shooting method.

Then we presents how the new A-stable and L-stable Obreshkov-based high-order integration method can be employed inside the shooting method and inspect whether better accuracy can be achieved by adopting the high-order methods, compared with the low-order shooting methods (the shooting method based on the classical low-order integration methods).

5.2.2 Shooting Method Based on the High-Order Methods

The main goal in the shooting method is to compute the vector $\boldsymbol{x}_0$ that satisfies the nonlinear system of equations (5.6).

The staring point in the shooting method is to come up with a suitably chosen initial trial vector, denoted by $\boldsymbol{x}_0^{(0)}$, for the solution vector $\boldsymbol{x}_0$. Typically, it is not likely that this

initial trial vector $\boldsymbol{x}_0^{(0)}$ is the exact solution, so it is not expected to satisfy the nonlinear system in (5.6).

Using the N-R method to solve (5.6) requires the iterative computation of the error function $\psi\left(\boldsymbol{x}_0^{(j)}\right)$ and its Jacobian matrix at successive trial vector $\boldsymbol{x}_0^{(j)}$. We can obtain the i^{th} trial vector $\boldsymbol{x}_0^{(i)}$ from the $(i-1)^{th}$ vector $\boldsymbol{x}_0^{(i-1)}$ through the Newton-Raphson update rule,

$$\boldsymbol{x}_0^{(i)} = \boldsymbol{x}_0^{(i-1)} - \Delta\boldsymbol{x}_0^{(i-1)} \tag{5.7}$$

where $\boldsymbol{x}_0^{(i-1)}$ is calculated from the previous N-R iteration $(i-1)$ and $\Delta\boldsymbol{x}_0^{(i-1)}$ is the N-R correction term as follows

$$\Delta\boldsymbol{x}_0^{(i-1)} = \left(\boldsymbol{\mathcal{J}}(\boldsymbol{x}_0^{(i-1)})\right)^{-1}\boldsymbol{\psi}\left(\boldsymbol{x}_0^{(i-1)}\right) \tag{5.8}$$

where $\boldsymbol{\mathcal{J}}(\boldsymbol{x}_0^{(i-1)})$ is the Jacobian matrix of $\boldsymbol{\psi}$ w.r.t $\boldsymbol{x}_0$ computed at $\boldsymbol{x}_0 = \boldsymbol{x}_0^{(i-1)}$, i.e.,

$$\begin{aligned}\boldsymbol{\mathcal{J}}(\boldsymbol{x}_0^{(i-1)}) &= \left.\frac{\partial\boldsymbol{\psi}}{\partial\boldsymbol{x}_0}\right|_{\boldsymbol{x}_0=\boldsymbol{x}_0^{(i-1)}} \\ &= \left.\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}_0}\right|_{\boldsymbol{x}_0=\boldsymbol{x}_0^{(i-1)}} - \boldsymbol{I}\end{aligned} \tag{5.9}$$

and where $\left(\cdot\right)^{-1}$ in (5.8) denotes the matrix inversion operation and $\boldsymbol{\psi}\left(\boldsymbol{x}_0^{(i-1)}\right)$ is given by

$$\boldsymbol{\psi}(\boldsymbol{x}_0^{(i-1)}) = \boldsymbol{\phi}(\boldsymbol{x}_0^{(i-1)}, t_0, T) - \boldsymbol{x}_0^{(i-1)} \tag{5.10}$$

by substituting $\boldsymbol{x}_0 = \boldsymbol{x}_0^{(i-1)}$ in (5.6).

It is usually not possible to obtain an analytical expression for the STF, i.e., $\boldsymbol{\phi}(\boldsymbol{x}_0, t_0, T)$. Thus, we can only use some numerical techniques to calculate the STF and the corresponding Jacobian matrix of the STF, i.e., $\boldsymbol{\mathcal{J}}(\boldsymbol{x}_0^{(i-1)})$. The two key issues of the implementation of the Newton Shooting method include both evaluating the STF, $\boldsymbol{\phi}(\boldsymbol{x}_0, t_0, T)$,

and the Jacobian matrix $\boldsymbol{\mathcal{J}}(\boldsymbol{x}_0)$ at any given trial vector $\boldsymbol{x}_0^{(i)}$. The following part shows how to deal with these two central issues while adopting the new Obreshkov-based high-order integration methods.

Computation of the STF, $\boldsymbol{\phi}(\boldsymbol{x}_0^{(i)}, t_0, T)$

Let us assume that $\boldsymbol{x}(0) = \boldsymbol{x}_0^{(i)}$ is the system initial state at $t = 0$. According to the definition of the STF, $\boldsymbol{\phi}(\boldsymbol{x}_0^{(i)}, t_0, T)$ is the numerical value of the system state, i.e., the MNA variables $\boldsymbol{x}(t)$, after a period of time T. In other words,

$$\boldsymbol{\phi}(\boldsymbol{x}_0^{(i)}, t_0, T) = \boldsymbol{x}(T)|_{\boldsymbol{x}(0)=\boldsymbol{x}_0^{(i)}} \tag{5.11}$$

It can be seen from (5.11) that the calculation of $\boldsymbol{\phi}(\boldsymbol{x}_0^{(i)}, t_0, T)$ is equivalent to the calculation of $\boldsymbol{x}(T)$ using the initial conditions $\boldsymbol{x}(0) = \boldsymbol{x}_0^{(i)}$, which is an example of a classical initial value problem [33]. Thus, the classical low-order integration methods such as BE and TR discussed in Chapter 2 can be used for this objective. Instead, however, we employ the new Obreshkov-based high-order formula to compute $\boldsymbol{x}(T)$ using the initial conditions $\boldsymbol{x}(0) = \boldsymbol{x}_0^{(i)}$.

Applying the Obreshkov-based high-order formula in (2.27) into the system (5.2) leads to the following augmented system of equations

$$\tilde{\boldsymbol{C}}\boldsymbol{\xi}_{n+1} + \tilde{\boldsymbol{G}}\boldsymbol{\xi}_{n+1} + \tilde{\boldsymbol{\rho}}_{n+1} = \tilde{\boldsymbol{b}}_{n+1} \tag{5.12}$$

whose components has been discussed in Chapter 2 and the vector $\boldsymbol{\xi}_{n+1}$ is constructed from the h-scaled derivatives of the current state variables at the current time point $t = t_{n+1}$ as follows

$$\boldsymbol{\xi}_{n+1} = \begin{bmatrix} \boldsymbol{x}_{n+1}^{(0)\top} & h\boldsymbol{x}_{n+1}^{(1)\top} & \cdots & h^k\boldsymbol{x}_{n+1}^{(k)\top} \end{bmatrix}^\top \tag{5.13}$$

where $\top$ denotes the transpose operator. $\boldsymbol{x}_{n+1}^{(0)} \approx \boldsymbol{x}(t_{n+1})$ is the approximation for the actual value of the waveform in $\boldsymbol{x}(t)$ at the time point $t = t_{n+1}$. Computing $\boldsymbol{\xi}_{n+1}$ can be achieved by solving the system of nonlinear equations (5.12) given that the approximations to $\boldsymbol{x}(t_n)$ and its high order derivatives at the previous time point $t = t_n$ are already known, from which the vector $\boldsymbol{\xi}_n$ at the previous time point $t = t_n$ is constructed

$$\boldsymbol{\xi}_n = \begin{bmatrix} \boldsymbol{x}_n^{(0)^\top} & h\boldsymbol{x}_n^{(1)^\top} & \cdots & h^m \boldsymbol{x}_n^{(m)^\top} \end{bmatrix}^\top \tag{5.14}$$

Our goal is to approximate $\boldsymbol{x}(t)$ at the time point $t = T$, and that is $\boldsymbol{x}^{(0)}|_{t=T} \approx \boldsymbol{x}(T)$, which can be extracted from the vector $\boldsymbol{\xi}$ at the time point $t = T$ as follows

$$\boldsymbol{\xi}|_{t=T} = \begin{bmatrix} \boldsymbol{x}_T^{(0)^\top} & h\boldsymbol{x}_T^{(1)^\top} & \cdots & h^k \boldsymbol{x}_T^{(k)^\top} \end{bmatrix}^\top \tag{5.15}$$

Computing the vector $\boldsymbol{\xi}|_{t=T}$ starts with the initial condition vector $\boldsymbol{\xi}_0$ based on the system initial state at $t = 0$, i.e., $\boldsymbol{x}(0) = \boldsymbol{x}_0^{(i)}$, and proceeds forward in time for a period of time T. Thus, the value of $\boldsymbol{\phi}(\boldsymbol{x}_0^{(i)}, t_0, T)$ can be obtained from the vector $\boldsymbol{\xi}(t)$ reached at $t = T$ in (5.15).

The N-R mehtod can be used to solve the system of nonlinear algebraic equations (5.12) and the Jacobian matrix needs to be factorized at each iteration, which can be

given by

$$\tilde{\boldsymbol{J}} = \tilde{\boldsymbol{C}} + \tilde{\boldsymbol{G}} + \frac{\partial \tilde{\boldsymbol{\rho}}_{n+1}}{\partial \boldsymbol{\xi}_{n+1}}$$

$$= \overbrace{\begin{bmatrix} \boldsymbol{G} + \boldsymbol{J}_{0,0} & \frac{1}{h}\boldsymbol{C} + \boldsymbol{J}_{0,1} & \boldsymbol{J}_{0,2} & \dots & \boldsymbol{J}_{0,k} \\ \boldsymbol{J}_{1,0} & \boldsymbol{G} + \boldsymbol{J}_{1,1} & \frac{1}{h}\boldsymbol{C} + \boldsymbol{J}_{1,2} & \dots & \boldsymbol{J}_{1,k} \\ \vdots & & \ddots & & \vdots \\ \boldsymbol{J}_{k-1,0} & \boldsymbol{J}_{k-1,1} & \dots & & \frac{1}{h}\boldsymbol{C} + \boldsymbol{J}_{k-1,k} \\ \alpha_0 \boldsymbol{I} & \alpha_1 \boldsymbol{I} & \dots & & \alpha_k \boldsymbol{I} \end{bmatrix}}^{(k+1)\times(k+1) \quad blocks} \tag{5.16}$$

where $\boldsymbol{J}_{i,j} = \frac{\partial \boldsymbol{f}_{n+1}^{(i)}}{\partial \boldsymbol{x}_{n+1}^{(j)}} h^{i-j}$ and the Jacobian matrix is a lower block Hessenberg matrix [34] because of $\boldsymbol{J}_{i,j} = 0$ when $i < j$ as discussed in Chapter 2.

Computation of the Jacobian matrix of the STF, $\partial\boldsymbol{\phi}/\partial\boldsymbol{x}_0$

According to the discussion above and the definition of the STF, we have

$$\left.\frac{\partial \boldsymbol{\phi}(\boldsymbol{x}_0, 0, T)}{\partial \boldsymbol{x}_0}\right|_{\boldsymbol{x}_0 = \boldsymbol{x}_0^{(i)}} = \left.\left(\frac{\partial \boldsymbol{x}(T)}{\partial \boldsymbol{x}_0}\right)\right|_{\boldsymbol{x}_0 = \boldsymbol{x}_0^{(i)}} \tag{5.17}$$

The matrix $\partial\boldsymbol{x}(T)/\partial\boldsymbol{x}_0 \in \mathbb{R}^{N\times N}$ is the so called sensitivity matrix of the final system state $\boldsymbol{x}(T)$ w.r.t the initial system state $\boldsymbol{x}_0$ and it can be easily obtained from the matrix $\partial\boldsymbol{\xi}(T)/\partial\boldsymbol{x}_0 \in \mathbb{R}^{(k+1)N\times N}$ for the first $N \times N$ block by neglecting the rest of the high order derivative blocks with the matrix size of $kN \times N$. Thus, calculating $\partial\boldsymbol{\xi}(T)/\partial\boldsymbol{x}_0$ becomes the main goal to calculate the Jacobian matrix of the STF and will be discussed next.

Computation of $\partial\boldsymbol{\xi}(T)/\partial\boldsymbol{x}_0$ Typically it is difficult to find the connection between the augmented final system state vector $\boldsymbol{\xi}(T)$ at $t = T$ and the initial system state $\boldsymbol{x}_0$

at $t = 0$. However it is always possible to find the relation relating the augmented final system state vector $\boldsymbol{\xi}(T)$ to the augmented system state vector $\boldsymbol{\xi}(T-h)$ at an earlier state at $t = T - h$, for a adequately small time step h. As discussed above, the Obreshkov-based high-order formula has been used to reach $\boldsymbol{\xi}(T)$ in the initial value problem phase [35], so we have

$$\tilde{\boldsymbol{C}}\boldsymbol{\xi}(T) + \tilde{\boldsymbol{G}}\boldsymbol{\xi}(T) + \tilde{\boldsymbol{\rho}}(T) = \tilde{\boldsymbol{b}}(T) \tag{5.18}$$

Note that the vector $\tilde{\boldsymbol{b}}(T) \in \mathbb{R}^{(k+1)N\times N}$ contains the previous state variables at $t = T-h$ and h-scaled derivatives of the independent stimuli to the circuit.

According to (5.18), the matrix of partial derivatives of $\boldsymbol{\xi}(T)$ w.r.t. $\boldsymbol{\xi}(T-h)$ can be obtained as the solution to the system as follows

$$\left(\tilde{\boldsymbol{C}} + \tilde{\boldsymbol{G}} + \frac{\partial\tilde{\boldsymbol{\rho}}(T)}{\partial\boldsymbol{\xi}(T)}\right)\frac{\partial\boldsymbol{\xi}(T)}{\partial\boldsymbol{\xi}(T-h)} = \frac{\partial\tilde{\boldsymbol{b}}(T)}{\partial\boldsymbol{\xi}(T-h)} \tag{5.19}$$

where the size of the matrix $\left(\tilde{\boldsymbol{C}} + \tilde{\boldsymbol{G}} + \partial\tilde{\boldsymbol{\rho}}(T)/\partial\boldsymbol{\xi}(T)\right)$ is $(k+1)N \times (k+1)N$, the size of the matrix $\partial\boldsymbol{\xi}(T)/\partial\boldsymbol{\xi}(T-h)$ is $(k+1)N \times (m+1)N$ and the size of the matrix $\partial\tilde{\boldsymbol{b}}(T)/\partial\boldsymbol{\xi}(T-h)$ is $(k+1)N \times (m+1)N$.

The Jacobian matrix of the augmented system (5.12) in the time-domain can be represented by the term between brackets in (5.19). It should be noted that the LU factors of the Jacobian matrix have been generated during the transient simulation to reach $\boldsymbol{\xi}(T)$ and those factors can be stored and reused to solve the linear system (5.19).

To use the matrix $\partial\boldsymbol{\xi}(T)/\partial\boldsymbol{\xi}(T-h)$ towards calculating $\partial\boldsymbol{\xi}(T)/\partial\boldsymbol{x}_0$, we apply the chain rule of differentiation to obtain

$$\frac{\partial\boldsymbol{\xi}(T)}{\partial\boldsymbol{x}_0} = \left(\frac{\partial\boldsymbol{\xi}(T)}{\partial\boldsymbol{\xi}(T-h)}\right)\left(\frac{\partial\boldsymbol{\xi}(T-h)}{\partial\boldsymbol{x}_0}\right) \tag{5.20}$$

where $\frac{\partial\boldsymbol{\xi}(T)}{\partial\boldsymbol{x}_0} \in \mathbb{R}^{(k+1)N\times N}$, $\frac{\partial\boldsymbol{\xi}(T)}{\partial\boldsymbol{\xi}(T-h)} \in \mathbb{R}^{(k+1)N\times(m+1)N}$ and $\frac{\partial\boldsymbol{\xi}(T-h)}{\partial\boldsymbol{x}_0} \in \mathbb{R}^{(m+1)N\times N}$.

Expanding $\partial\boldsymbol{\xi}(T)/\partial\boldsymbol{x}_0$ by repeatedly applying the chain rule of differentiation leads to

$$\frac{\partial\boldsymbol{\xi}(T)}{\partial\boldsymbol{x}_0} = \left(\frac{\partial\boldsymbol{\xi}(T)}{\partial\boldsymbol{\xi}(t_s)}\right)\left(\frac{\partial\boldsymbol{\xi}(t_s)}{\partial\boldsymbol{\xi}(t_{s-1})}\right)\cdots\left(\frac{\partial\boldsymbol{\xi}(t_1)}{\partial\boldsymbol{x}_0}\right)\left(\frac{\partial\boldsymbol{x}_0}{\partial\boldsymbol{x}_0}\right) \tag{5.21}$$

assuming that $t_1, t_2, \cdots, t_s$ is the sequence of the time steps used to reach $\boldsymbol{\xi}(T)$ starting from $\boldsymbol{\xi}(0)$. It is worth noting that the last matrix on the right side of (5.21) is equivalent to the identity matrix, so the matrix $\partial\boldsymbol{\xi}(T)/\partial\boldsymbol{x}_0$ can be reduced to

$$\frac{\partial\boldsymbol{\xi}(T)}{\partial\boldsymbol{x}_0} = \left(\frac{\partial\boldsymbol{\xi}(T)}{\partial\boldsymbol{\xi}(t_s)}\right)\left(\frac{\partial\boldsymbol{\xi}(t_s)}{\partial\boldsymbol{\xi}(t_{s-1})}\right)\cdots\left(\frac{\partial\boldsymbol{\xi}(t_1)}{\partial\boldsymbol{x}_0}\right) \tag{5.22}$$

which is a sequence of s matrix multiplications where the q^{th} matrix, $\partial\boldsymbol{\xi}(t_{q+1})/\partial\boldsymbol{\xi}(t_q)$, is given by the matrix-valued solution to the linear system of equations as follows,

$$\left(\tilde{\boldsymbol{C}} + \tilde{\boldsymbol{G}} + \frac{\partial\tilde{\boldsymbol{\rho}}(t_{q+1})}{\partial\boldsymbol{\xi}(t_{q+1})}\right)\frac{\partial\boldsymbol{\xi}(t_{q+1})}{\partial\boldsymbol{\xi}(t_q)} = \frac{\partial\tilde{\boldsymbol{b}}(t_{q+1})}{\partial\boldsymbol{\xi}(t_q)} \tag{5.23}$$

since the Obreshkov-based high order integration formula has been used to calculate $\boldsymbol{\xi}(T)$ in the initial value problem phase and $\partial\tilde{\boldsymbol{b}}(t_{q+1})/\partial\boldsymbol{\xi}(t_q)$ in (5.23) is given by

$$\frac{\partial\tilde{\boldsymbol{b}}(t_{q+1})}{\partial\boldsymbol{\xi}(t_q)} = \overbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \beta_0\boldsymbol{I} & \beta_1\boldsymbol{I} & \beta_2\boldsymbol{I} & \dots & \beta_m\boldsymbol{I} \end{bmatrix}}^{(k+1)\times(m+1)\quad blocks} \tag{5.24}$$

5.3 Summary

Algorithm 1 and Algorithm 2 illustrate the procedure to implement the shooting method based on the Obreshkov-base high-order integration methods. Algorithm 1 is the main body of the high-order shooting method and Algorithm 2 illustrates the two central issues in high-order shooting method including the computation of STF $\boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T)$ and its Jacobian matrix $\frac{\partial \boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T)}{\partial \boldsymbol{x}_0^{(j)}}$.

Algorithm 1: Shooting method based on Obreshkov-based high-order methods (Main body)

Input: Nonlinear MNA equation: $\boldsymbol{G}\boldsymbol{x}(t) + \boldsymbol{C}\frac{d\boldsymbol{x}(t)}{dt} + \boldsymbol{f}(\boldsymbol{x}(t)) = \boldsymbol{b}(t)$

Output: A solution vector $\boldsymbol{x}_0^*$ satisfying $\boldsymbol{\psi}(\boldsymbol{x}_0) = \boldsymbol{\phi}(\boldsymbol{x}_0, t_0, T) - \boldsymbol{x}_0 = 0$

1 Initial guess trial vector: $\boldsymbol{x}_0^{(0)} \qquad (j = 0)$;

2 $\epsilon \leftarrow$ some small positive number ; `// an acceptable error threshold`

3 Form the nonlinear error function: $\boldsymbol{\psi}(\boldsymbol{x}_0) = \boldsymbol{\phi}(\boldsymbol{x}_0, t_0, T) - \boldsymbol{x}_0 = 0$;

4 Form Jacobian of $\boldsymbol{\psi}(\boldsymbol{x}_0)$: $\boldsymbol{J}(\boldsymbol{x}_0) = \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{x}_0} = \frac{\partial \boldsymbol{\phi}(\boldsymbol{x}_0, t_0, T)}{\partial \boldsymbol{x}_0} - \boldsymbol{I}$;

5 **while** $||\boldsymbol{\psi}(\boldsymbol{x}_0^{(j)})|| > \epsilon$ **do**

  `/* Detailed calculation of` $\boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T)$ `and` $\frac{\partial \boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T)}{\partial \boldsymbol{x}_0^{(j)}}$ `are shown in`

  Algorithm 2 `*/`

6 Compute $\boldsymbol{\psi}(\boldsymbol{x}_0^{(j)}) = \boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T) - \boldsymbol{x}_0^{(j)}$;

7 Compute $\boldsymbol{J}(\boldsymbol{x}_0^{(j)}) = \frac{\partial \boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T)}{\partial \boldsymbol{x}_0^{(j)}} - \boldsymbol{I}$;

8 Solve $\boldsymbol{J}(\boldsymbol{x}_0^{(j)})\Delta\boldsymbol{x}_0^{(j)} = -\boldsymbol{\psi}(\boldsymbol{x}_0^{(j)})$;

9 Find $\boldsymbol{x}_0^{(j+1)} = \boldsymbol{x}_0^{(j)} + \Delta\boldsymbol{x}_0^{(j)}$;

10 $j = j + 1$;

11 $\boldsymbol{x}_0^* = \boldsymbol{x}_0^{(j)}$;

12 **return** $\boldsymbol{x}_0^*$;

Algorithm 2: Procedure in calculating $\boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T)$ and $\frac{\partial \boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T)}{\partial \boldsymbol{x}_0^{(j)}}$ using Obreshkov-based high-order methods

Input: The augmented system of DEs: $\tilde{\boldsymbol{C}}\boldsymbol{\xi}_{n+1} + \tilde{\boldsymbol{G}}\boldsymbol{\xi}_{n+1} + \tilde{\boldsymbol{\rho}}_{n+1} - \tilde{\boldsymbol{b}}_{n+1} = 0$, after applying high order formula

Output: STF $\boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T)$ and Jacobian matrix of STF $\frac{\partial \boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T)}{\partial \boldsymbol{x}_0^{(j)}}$

/* This shows qth time step only and $\boldsymbol{\xi}_{q-1}$ is the result from previous time step. */

for *time-step* $= q$ **do**

  /* Solve a nonlinear Eqs $\tilde{\boldsymbol{C}}\boldsymbol{\xi}_q + \tilde{\boldsymbol{G}}\boldsymbol{\xi}_q + \tilde{\boldsymbol{\rho}}_q - \tilde{\boldsymbol{b}}_q = 0$ using NR method to obtain $\boldsymbol{\xi}_q$. */

  /* This shows ith NR iteration only and $\boldsymbol{\xi}_q^{(i-1)}$ is the result from previous NR iteration */

  while $||\tilde{\boldsymbol{\psi}}(\boldsymbol{\xi}_q^{(i)})|| > \epsilon$ **do**

    Compute $\tilde{\boldsymbol{\psi}}(\boldsymbol{\xi}_q^{(i-1)}) = \tilde{\boldsymbol{C}}\boldsymbol{\xi}_q^{(i-1)} + \tilde{\boldsymbol{G}}\boldsymbol{\xi}_q^{(i-1)} + \tilde{\boldsymbol{\rho}}_q^{(i-1)} - \tilde{\boldsymbol{b}}_q^{(i-1)}$;

    Compute $\tilde{\boldsymbol{J}}(\boldsymbol{\xi}_q^{(i-1)}) = \tilde{\boldsymbol{C}} + \tilde{\boldsymbol{G}} + \frac{\partial \tilde{\boldsymbol{\rho}}_q^{(i-1)}}{\partial \boldsymbol{\xi}_q^{(i-1)}}$;

    Solve $\tilde{\boldsymbol{J}}(\boldsymbol{\xi}_q^{(i-1)})\Delta\boldsymbol{\xi}_q^{(i-1)} = -\tilde{\boldsymbol{\psi}}(\boldsymbol{\xi}_q^{(i-1)})$;

    Find $\boldsymbol{\xi}_q^{(i)} = \boldsymbol{\xi}_q^{(i-1)} + \Delta\boldsymbol{\xi}_q^{(i-1)}$;

    $i = i + 1$;

  /* At the end of NR iterations, we have $\boldsymbol{\xi}(t_q) = \boldsymbol{\xi}_q$, which is the augmented system state vector at time t_{q+1}. */

  Solve $\left(\tilde{\boldsymbol{C}} + \tilde{\boldsymbol{G}} + \frac{\partial \tilde{\boldsymbol{\rho}}(t_q)}{\partial \boldsymbol{\xi}(t_q)}\right)\frac{\partial \boldsymbol{\xi}(t_q)}{\partial \boldsymbol{\xi}(t_{q-1})} = \frac{\partial \tilde{\boldsymbol{b}}(t_q)}{\partial \boldsymbol{\xi}(t_{q-1})}$ /* $\frac{\partial \boldsymbol{\xi}(t_q)}{\partial \boldsymbol{\xi}(t_{q-1})}$ is stored and used as the qth multiplicative term in calculating $\frac{\partial \boldsymbol{\xi}(T)^{(j)}}{\partial \boldsymbol{x}_0^{(j)}}$, which is a sequence of s matrix multiplications. */

  $q = q + 1$; // Continue to the next time step

/* Assume that $t_1, t_2, \cdots, t_s$ is the sequence of the time steps used to reach $\boldsymbol{\xi}(T)$ starting from $\boldsymbol{\xi}(0)$ and h is the fixed time step size. */

$\boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T) = \boldsymbol{x}_T^{(0)}$ can be obtained from $\boldsymbol{\xi}(T)$ after reaching $t = T$; // See (5.15)

Compute $\frac{\partial \boldsymbol{\xi}(T)}{\partial \boldsymbol{x}_0} = \left(\frac{\partial \boldsymbol{\xi}(T)}{\partial \boldsymbol{\xi}(t_s)}\right)\left(\frac{\partial \boldsymbol{\xi}(t_s)}{\partial \boldsymbol{\xi}(t_{s-1})}\right)\cdots\left(\frac{\partial \boldsymbol{\xi}(t_1)}{\partial \boldsymbol{x}_0}\right)$; // Multiplicative terms have been obtained already

$\frac{\partial \boldsymbol{\phi}(\boldsymbol{x}_0^{(j)}, t_0, T)}{\partial \boldsymbol{x}_0^{(j)}}$ can be obtained from $\frac{\partial \boldsymbol{\xi}(T)}{\partial \boldsymbol{x}_0}$.

5.4 Discussion

This chapter compared the warping error of the high-order methods with the warping error of the classical low-order methods and it showed that the warping error of the high-order methods is much smaller. To inspect the impact of the smaller warping error in the high-order methods on the steady state analysis using the shooting method, the high-order shooting method has been derived in this chapter.

The high-order method is constructed by including information from higher-order derivatives compared with the low-order methods for transient analysis of circuits and allows circuit simulation to take larger step sizes without violating stability. This leads to considerable savings in the number of time point solutions for waveform evaluations and therefore accelerate time-domain simulation. Some high-order shooting method numerical results based on certain circuits will be presented in the next chapter.

Chapter 6

Numerical results

In the previous chapter, we have developed the shooting method using the high-order methods to inspect the difference in the performance between the high-order methods and the low-order methods adopted in the shooting method (SM). Some numerical examples are presented in the this chapter. Results were computed by employing the high-order shooting method (developed in the previous chapter) and the low-order shooting method with different number of time steps. Next, we compare the results of these two methods with the results computed by Harmonic Balance (HB) method [36] [37] [38], since HB does not introduce warping errors. However, it should be noted that HB performs well only if the circuits are mildly nonlinear with multi-tone excitations. In general, if circuits are strongly nonlinear, then time-domain approaches, such as shooting method, can perform better than HB. For highly nonlinear circuits, HB analysis would require a huge number of harmonics to account for the effects of nonlinearities, which may even cause convergence problems. The circuits used in this chapter are mildly nonlinear circuits. Our purpose, by comparing the results of the above three methods, is to demonstrate the basic advantages of the high-order methods by showing that they introduce much lower warping error compared to the low-order methods.

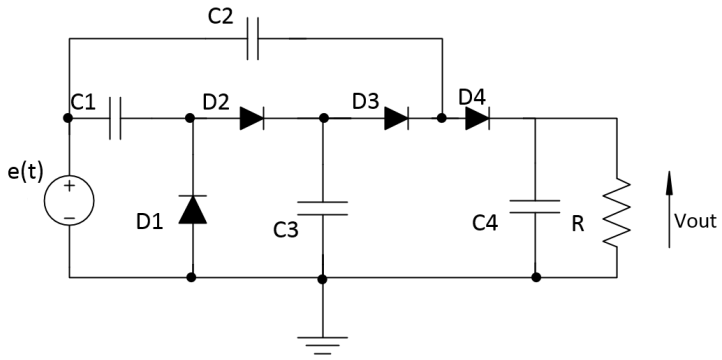

Figure 6.1: Voltage multiplier circuit diagram: $e(t) = 10sin(100\pi t)V$, $R = 470k\Omega$, $C1 = C2 = C3 = 1\mu F$, $C4 = 4.7\mu F$; Diode model parameters : $rs = 10m$, $is = 10f$, $tt = 100u$, $cjo = 10n$.

6.1 Example 1: Voltage Multiplier

The first circuit considered is the voltage multiplier shown in Figure 6.1 [39] [40] and the values of its parameters are also reported in Figure 6.1. We have calculated the behavior of the voltage multiplier at steady state by using the three methods mentioned above. In this case, the Trapezoidal Rule (TR) and high order method of order 6 (with k=m=3) were adopted in the shooting method. The Vout output voltage waveforms at steady state, computed by three methods, are shown in Figure 6.2 and Figure 6.3.

It can be seen from Figure 6.2 that the Vout voltage waveforms computed by high order (order 6) shooting method using 60 time steps are in very good agreement with the results from HB, but the waveforms computed by TR using the same number (60) of time steps shows an erroneous behavior of the voltage multiplier.

Figure 6.3 shows that after increasing the number of time steps up to 400, the results computed by low-order (TR) shooting method show good agreement with the results of HB. On the other hand, the high order shooting methods only requires 60 time steps for the same accuracy.

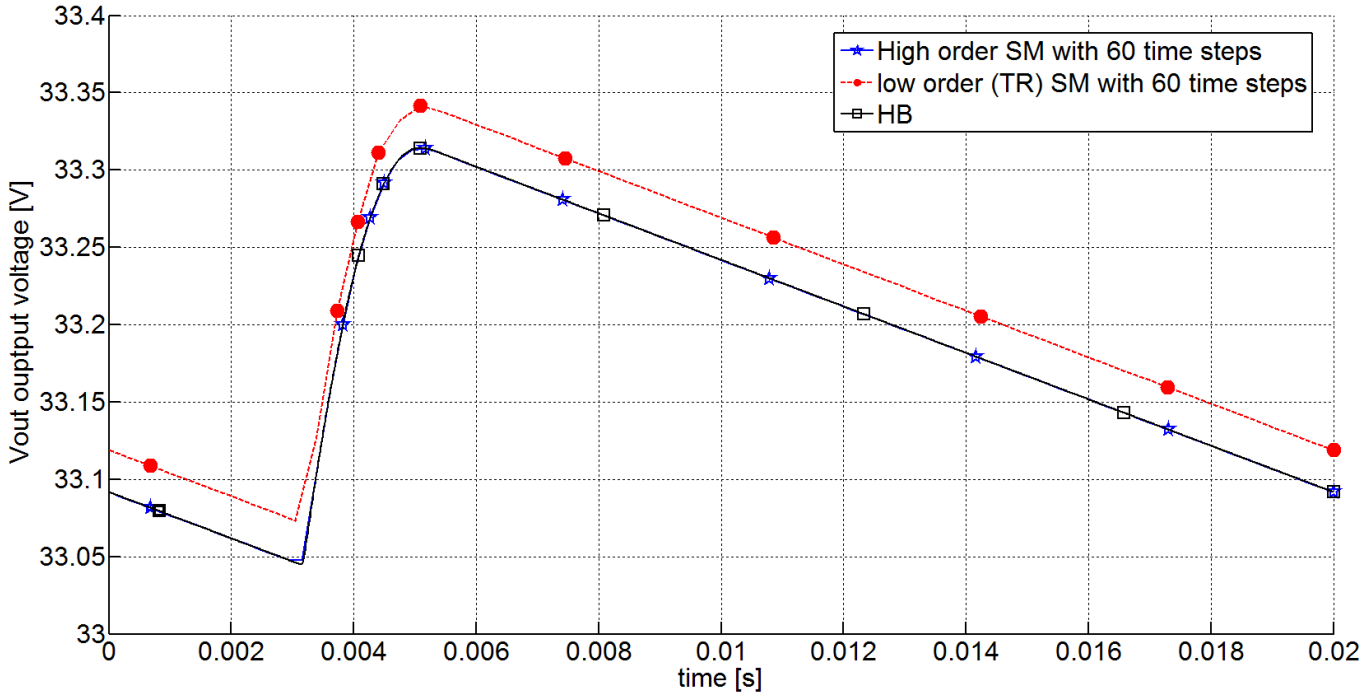

Figure 6.2: Vout voltage waveforms at steady state calculated by high-order SM, low-order SM and HB

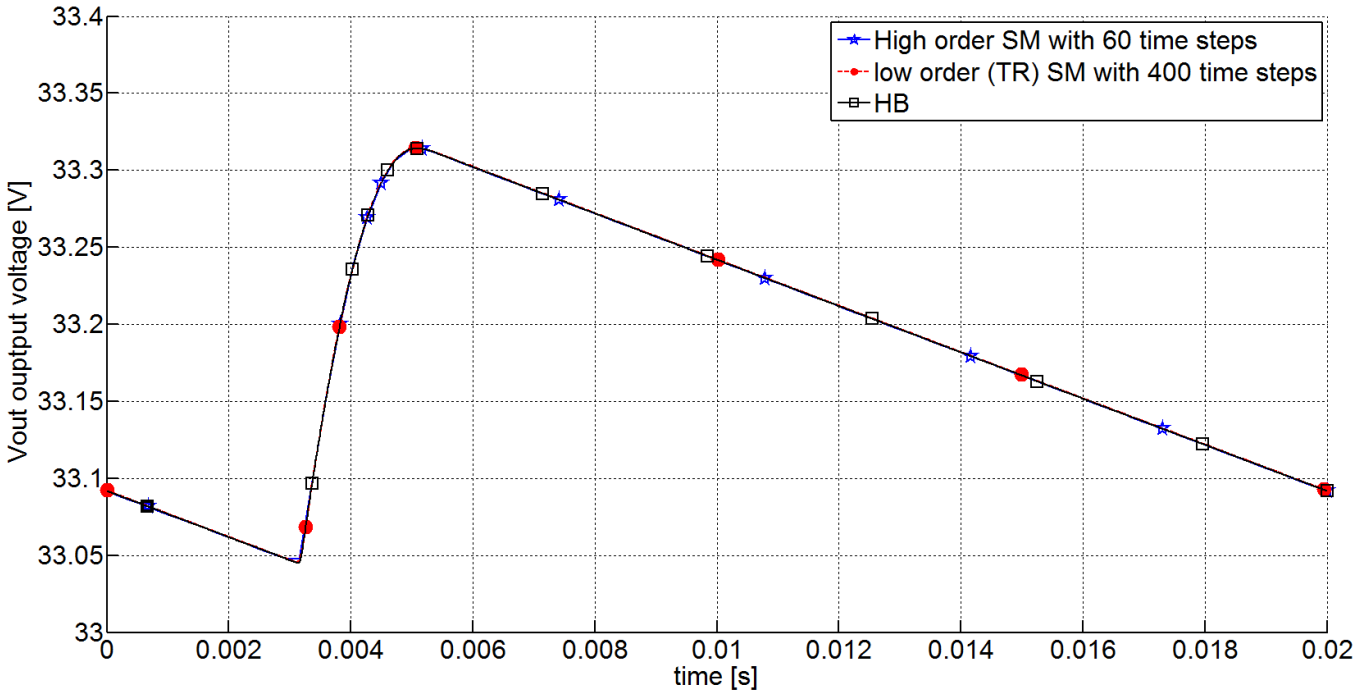

Figure 6.3: Vout voltage waveforms at steady state calculated by high-order SM, low-order SM and HB

6.2 Example 2: Tuned Amplifier

Another example considered is the tuned amplifier circuit shown in Figure 6.4 and the values of its parameters are also reported in Figure 6.4. The model of the crystal in the tuned amplifier and the values of its parameters are shown in Figure 6.5.

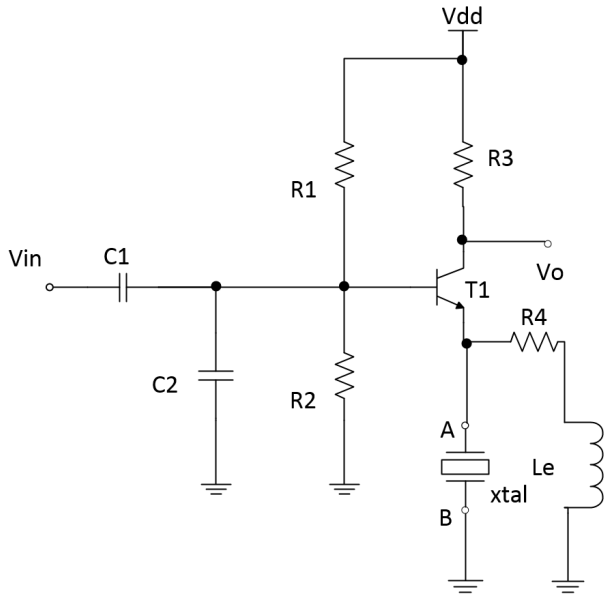

Figure 6.4: Tuned amplifier circuit diagram: $C1 = 33nF$, $C2 = 3.3nF$, $R1 = 8.2K\Omega$, $R2 = 4.7K\Omega$, $R3 = R4 = 750\Omega$, $Le = 220\mu H$, $Vdd = 20V$, $v_i = 100mV$, $f_o = 5MHz$, $Vin = v_i sin(2\pi f_o t)$.

In this example both TR and high-order method of order 5 (with $k = 3$, $m = 2$) were employed in the shooting method to simulate the tuned amplifier circuit as well as the HB as the third method, whose results are accurate. The Vo output voltage waveforms computed at steady state by the mentioned three methods are shown in Figure 6.6, Figure 6.7, Figure 6.8, Figure 6.9 and Figure 6.10, respectively. In each of these figures, the same number of time steps (step size h fixed) were used in both low-order and high-order shooting methods to simulate the circuit and then comparing the results of the

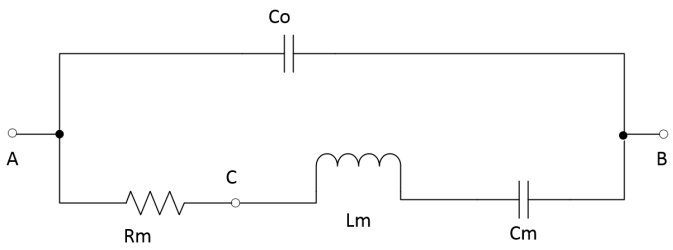

Figure 6.5: The equivalent model of the crystal in the tuned amplifier circuit: $Rm = 120\Omega$, $Cm = 0.165786399F$, $Co = 5pF$, $Lm = 6.11154981H$

low-order and high-order shooting methods with the HB results to inspect the difference in the performance between the low-order shooting methods and the high-order shooting methods.

It can be seen in Figure 6.6 that the Vo voltage waveforms computed by the low-order (TR) shooting method using 90 time steps shows an erroneous behavior of the tuned amplifier circuit, on the other hand, the results of the high-order shooting method using the same number (90) of time steps are closer to the HB results.

Then, increasing the number of time steps up to 250 in Figure 6.7, the results of HB and the high-order shooting method are in very good agreement, however, the results computed by the low-order shooting method still shows an erroneous behavior of the tuned amplifier.

We increased the number of time steps up to 1000 in Figure 6.8, the results of the low-order shooting method shows a slightly change compared with the previous figure but it is still erroneous. In contrast, the results of HB and the high-order shooting method are in very good agreement again.

After increasing the number of time steps up to 5000 in Figure 6.9, the waveforms computed by the low-order shooting method began to become closer to the results of HB and again the results of the high-order shooting method are in good agreement with the

HB results.

Finally, after we increased the number of time steps up to 20,000, the Vo voltage waveforms at steady state computed by all three methods are in good agreement, which are shown in Figure 6.10.

It can be seen from all these figures that to obtain the results that are in good agreement with HB results, the low order (TR) shooting methods needs 20,000 time steps, whereas the high order shooting methods only requires 250 time steps for the same accuracy.

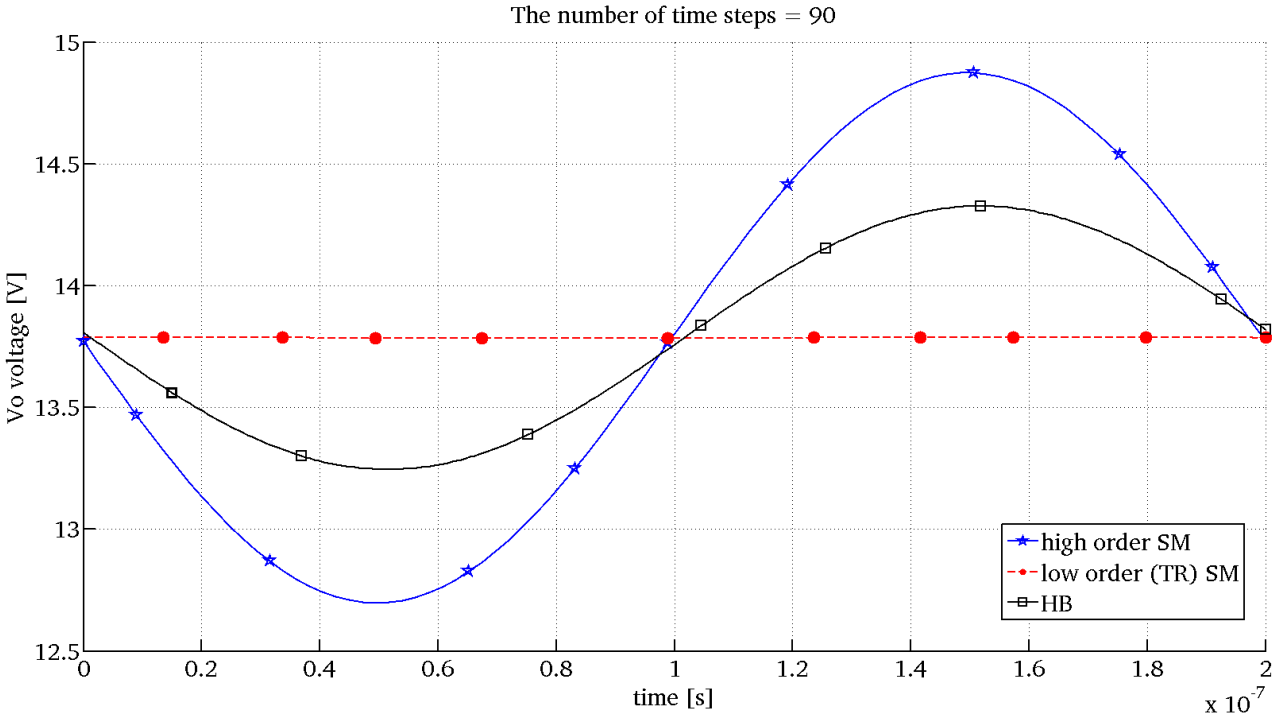

Figure 6.6: Vo voltage waveforms at steady state computed by high-order SM, low-order SM and HB

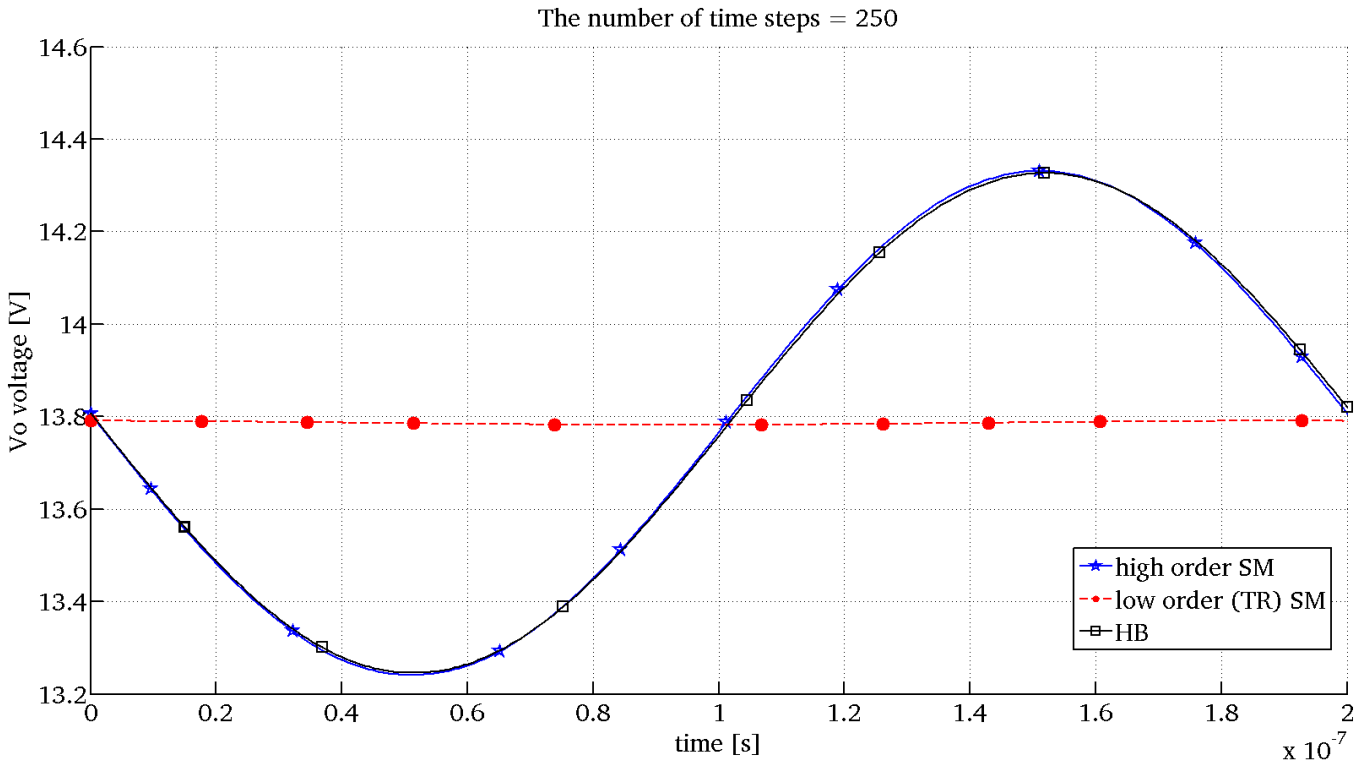

Figure 6.7: Vo voltage waveforms at steady state computed by high-order SM, low-order SM and HB

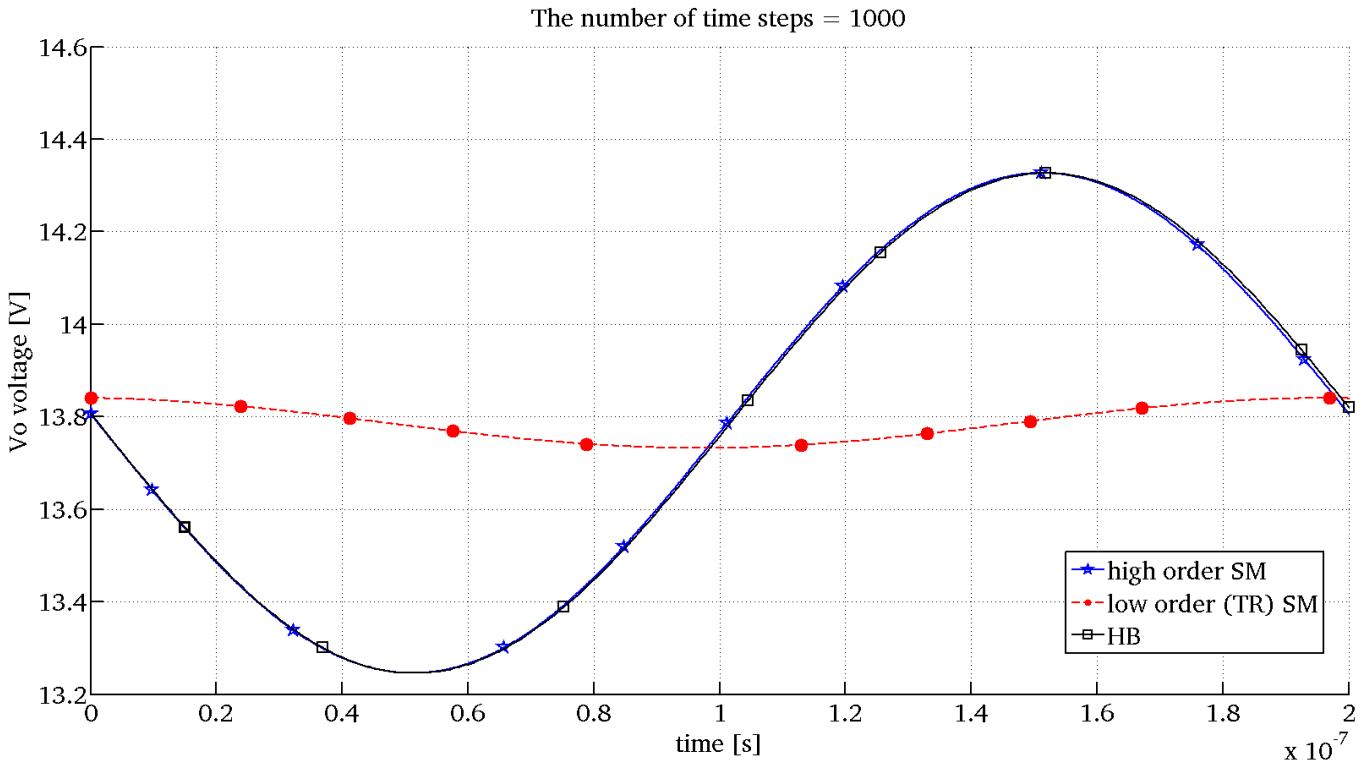

Figure 6.8: Vo voltage waveforms at steady state computed by high-order SM, low-order SM and HB

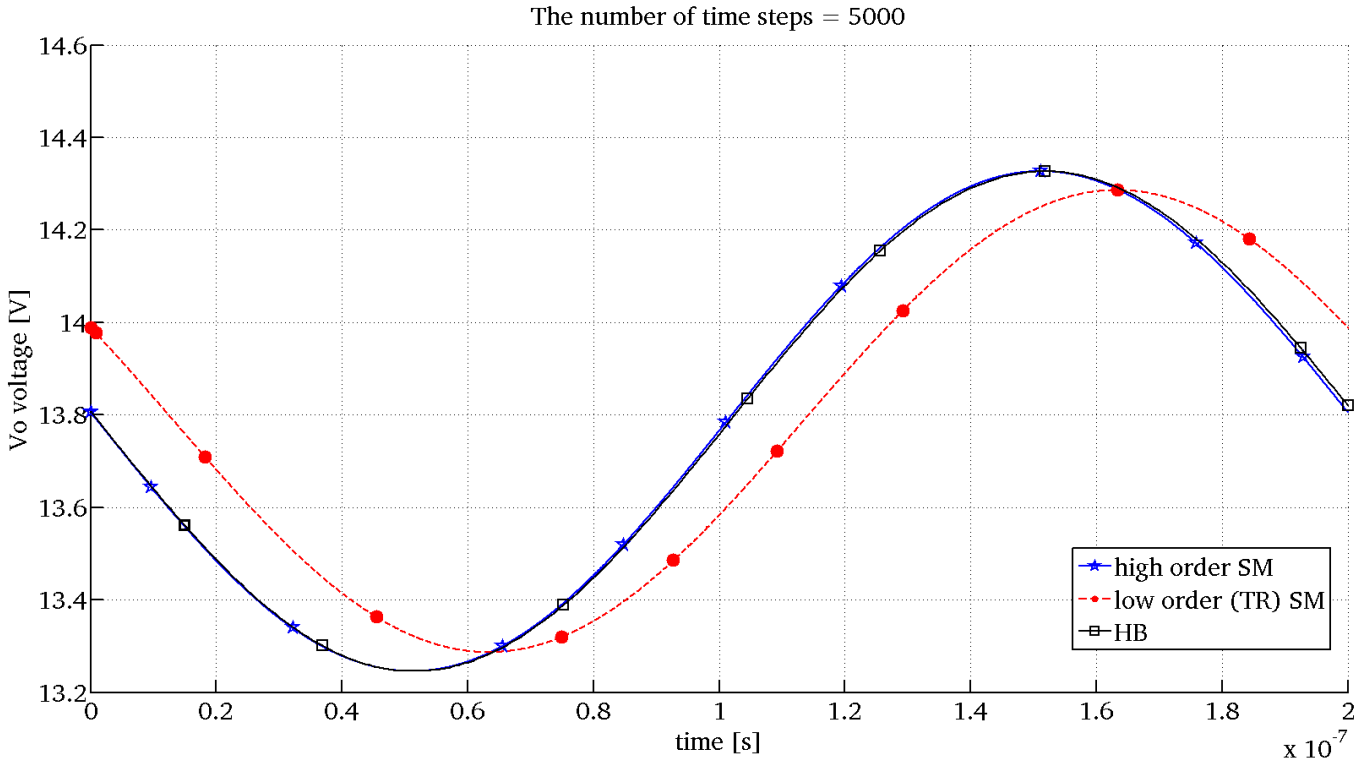

Figure 6.9: Vo voltage waveforms at steady state computed by high-order SM, low-order SM and HB

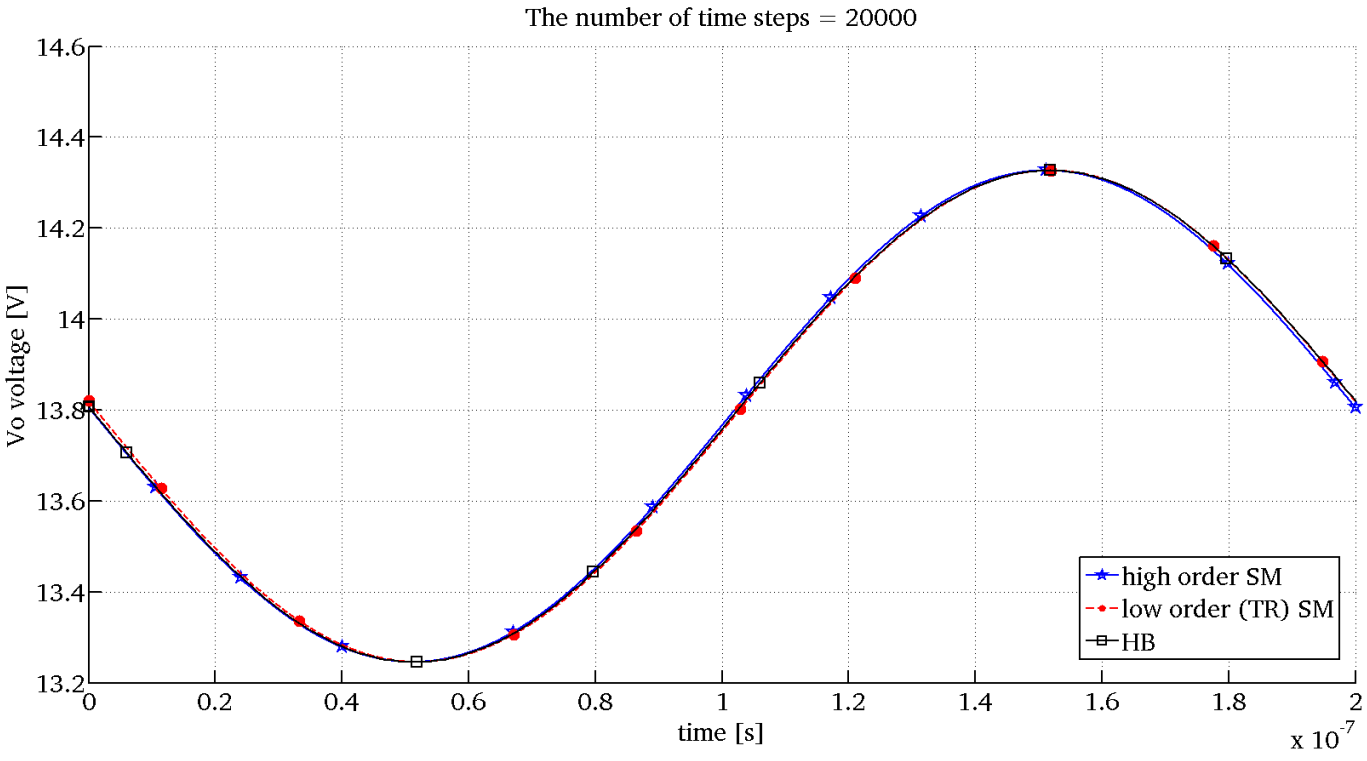

Figure 6.10: Vo voltage waveforms at steady state computed by high-order SM, low-order SM and HB

6.3 Summary

As can be observed above, the shooting method based on the high-order methods can achieve better accuracy while requiring shorter execution time when compared with the low-order shooting methods (Trapezoidal Rule in our case). This is because the Obreshkov-based high-order methods introduce much smaller warping error compared with the classical low-order methods discussed in Chapter 5. The Obreshkov-based high-order integration methods constructed with higher order derivatives allow circuit simulation to take larger step size without violating the stability, under the condition: $k - 2 \leq m \leq k$. This leads to considerable reduction in the number of time steps and therefore provides an overall speedup compared with the low-order methods.

Chapter 7

Summary and Future Work

7.1 Summary

This thesis first develops the derivation of the warping error of the Obreshkov-based high-order methods and then shows that the warping error introduced by high-order methods is much lower than the classical low-order methods adopted in SPICE engine, that is BE, TR and the Gear's methods. Next, in order to take the advantage that the high-order methods introduce much less warping errors compared to the classical low-order methods, this thesis develops a shooting method based on the high-order methods to compute the steady-state behaviour of nonlinear circuits excited by periodical sources. This new developed high-order shooting method shows that the steady-state response of nonlinear circuits can be constructed with a much smaller number of time steps compared to the number of time steps required by the low-order shooting method.

7.2 Future Work

The shooting method based on high-order integration methods developed in this thesis can compute the steady-state response of nonlinear circuits excited by periodical sources

with a much smaller number of time steps than the shooting method adopting the classical low-order methods. There is possibly some room to improve the implementation of shooting method. The shooting method considered in this thesis is single shooting method. One possible way to achieve improvement is combining the multiple shooting method [40] [41] with the Obreshkov-based high-order methods. We expect to develop the multiple shooting method based on high-order integration methods to inspect whether this can achieve better performance compared with the high-order single shooting method.

Bibliography

[1] B. Angelo and S. Giancarlo. Frequency warping in time-domain circuit simulation. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 50(7):904–913, 2003.

[2] G. Dahlquist. A special stability problem for linear multistep methods. *BIT Numerical Mathematics*, 3(1):27–43, 1963.

[3] C. Gear. *Numerical initial value problems in ordinary differential equations*, volume 59. Prentice-Hall Englewood Cliffs, NJ, 1971.

[4] T. L. Quarles. *The SPICE3 implementation guide*. Electronics Research Laboratory, College of Engineering, University of California, 1989.

[5] E. Gad, M. Nakhla, R. Achar, and Y. Zhou. A-stable and L-stable high-order integration methods for solving stiff differential equations. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(9):1359–1372, 2009.

[6] N. Obreshkov. Sur le quadrature mecaniques. *Spisanie Bulgar. Akad. Nauk.(Journal of the Bulgarian Academy of Sciences)*, 65:191–289, 1942.

[7] M. A. Farhan, E. Gad, M. Nakhla, and R. Achar. Fast simulation of microwave circuits with nonlinear terminations using high-order stable methods. 61(1):360–371, 2013.

[8] M. A. Farhan, E. Gad, M. Nakhla, and R. Achar. New method for fast transient simulation of large linear circuits using high-order stable methods. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 3(4):661–669, 2013.

[9] Y. Zhou, E. Gad, M. Nakhla, and R. Achar. Structural characterization and efficient implementation techniques for-stable high-order integration methods. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(1):101–108, 2012.

[10] Y. Lin and E. Gad. Formulation of the Obreshkov-based transient circuit simulator in the presence of nonlinear memory elements. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(1):86–94, 2015.

[11] B. Bradie. *A friendly introduction to numerical analysis*. Pearson Prentice Hall Upper Saddle River, NJ, 2006.

[12] J. C. Butcher. Numerical methods for ordinary differential systems, 2003.

[13] G. A. Baker and P. Graves-Morris. Padé approximants. 1996. *Encyclopedia of Mathematics and its Applications*, 1996.

[14] A. Iserles. *A first course in the numerical analysis of differential equations*, volume 44. Cambridge University Press, 2009.

[15] U. M. Ascher and L. R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*, volume 61. Siam, 1998.

[16] C. Gear. Simultaneous numerical solution of differential-algebraic equations. 18(1):89–95, 1971.

[17] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*, volume 12. Springer Science & Business Media, 2002.

[18] J. Vlach and K. Singhal. *Computer methods for circuit analysis and design*. Springer Science & Business Media, 1983.

[19] G. Wanner and E. Hairer. *Solving ordinary differential equations II*, volume 1. Springer-Verlag, Berlin, 1991.

[20] J. C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley, 2008.

[21] E. Hairer and G. Wanner. Solving ordinary differential equations ii. stiff and differential-algebraic problems, 2005.

[22] J. D. Lambert. *Computational Methods in Ordinary Differential Equations. Introductory Mathematics for Scientists and Engineers*. Wiley, 1973.

[23] G. Birkhoff and R. S. Varga. *Discretization errors for well-set Cauchy problems. I*. Defense Technical Information Center, 1964.

[24] G. Dahlquist. Convergence and stability in the numerical integration of ordinary differential equations. *Math. Scand*, 4(1):33–53, 1956.

[25] G. Wanner, E. Hairer, and S.P. Nørsett. Order stars and stability theorems. *BIT Numerical Mathematics*, 18(4):475–489, 1978.

[26] E. Gad, M. Nakhla, R. Achar, and Y. Zhou. An absolutely-stable arbitrarily high-order implicit numerical integration method and its application to the time-domain simulation of interconnect circuits. In *2007 IEEE Workshop on Signal Propagation on Interconnects*, pages 186–187, 2007.

[27] C. W. Ho, A. E. Ruehli, and P. A. Brennan. The modified nodal approach to network analysis. *Circuits and Systems, IEEE Transactions on*, 22(6):504–509, 1975.

[28] P. Favati, G. Lotti, and O. Menchi. Non-recursive solution of sparse block Hessenberg systems. *Numerical Linear Algebra with Applications*, 11(4):391–409, 2004.

[29] K. J. Aström and B. Wittenmark. *Computer-controlled systems: theory and design*. Courier Corporation, 2011.

[30] K. Kundert. *The designer's guide to SPICE and Spectre*. Kluwer Academic Publishers, 1995.

[31] D. Sheen. Introduction to numerical analysis. 1980.

[32] K. S. Kundert, J. K. White, and A. Sangiovanni-Vincentelli. Steady-state methods for simulating analog and microwave circuits. 1990.

[33] J. D. Lambert. The initial value problem for ordinary differential equations. *The state of the art in numerical analysis*, pages 451–500, 1977.

[34] G. W. Stewart. On the solution of block Hessenberg systems. *Numerical linear algebra with applications*, 2(3):287–296, 1995.

[35] C. Gear. *Numerical initial value problems in ordinary differential equations*, volume 59. Prentice-Hall Englewood Cliffs, NJ, 1971.

[36] M. Nakhla and J. Vlach. A piecewise harmonic balance technique for determination of periodic response of nonlinear systems. *Circuits and Systems, IEEE Transactions on*, 23(2):85–91, 1976.

[37] R. J. Gilmore and M. B. Steer. Nonlinear circuit analysis using the method of harmonic balanceŮa review of the art. part i. introductory concepts. *International Journal of Microwave and Millimeter-Wave Computer-Aided Engineering*, 1(1):22–37, 1991.

[38] K. S. Kundert and A. Sangiovanni-Vincentelli. Simulation of nonlinear circuits in the frequency domain. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 5(4):521–535, 1986.

[39] D. Bedrosian and J. Vlach. Time-domain analysis of networks with internally controlled switches. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 39(3):199–212, 1992.

[40] B. Angelo and M. Paolo. Envelope-following method to compute steady-state solutions of electrical circuits. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 50(3):407–417, 2003.

[41] D. Frey. On the equivalence of various methods for finding the periodic steady state solution of nonlinear networks. In *Circuits and Systems, 1999. ISCAS'99. Proceedings of the 1999 IEEE International Symposium on*, volume 5, pages 322–326. IEEE, 1999.