High-order and Stable Time-domain Simulation of Nonlinear Circuits

by

Bardia Bandali

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Doctor of Philosophy
in
Electrical and Computer Engineering

uOttawa

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

Examining Committee Members

The following peoples served on the Examining Committee for this thesis.

Supervisor:	Emad Gad Professor, School of Electrical Engineering & Computer Science University of Ottawa
Co-Supervisor:	Michel Nakhla Professor, Department of Electronics Carleton University
Internal Member:	Derek McNamara Professor, School of Electrical Engineering & Computer Science University of Ottawa
Internal Member:	Mustapha Yagoub Professor, School of Electrical Engineering & Computer Science University of Ottawa
Carleton Member:	Ramachandra Achar Professor, Department of Electronics Carleton University
External Member:	José Schutt Ainé Professor, Department of Electrical and Computer Engineering University of Illinois at Urbana-Champaigne

Abstract

This thesis presents a novel and efficient time-domain simulation methodology for nonlinear electronic circuits, addressing the computational challenges associated with traditional approaches. The core innovation lies in the development of a high-order, stable time-stepping scheme predicated on the Numerical Inversion of Laplace Transform (NILT). This method facilitates the accurate simulation of general circuits by employing a partitioned approach, wherein the circuit is decomposed into linear and nonlinear sub-networks. A key contribution is the matching of time-domain derivatives at the interface between these sub-networks, achieved through the use of the NILT and Rooted Trees (RT) methods.

Furthermore, this research utilizes an advanced formulation of NILT, designated as NILTn, which significantly enhances the accuracy per time step compared to conventional NILT (referred to as NILT0). By leveraging higher-order derivatives, NILTn greatly reduces truncation errors, thereby allowing simulation with larger step sizes enhancing overall performance.

The effectiveness of the proposed methodology is demonstrated through comprehensive numerical simulations, showcasing substantial speed improvements and enhanced accuracy compared to traditional time-domain simulation methods. This advancement enables the efficient and accurate simulation of nonlinear circuits, facilitating advancements in areas such as power electronics, Radio Frequency (RF) design, and mixed-signal systems.

Acknowledgments

It is almost impossible to appreciate properly of gentle people who are helpful during a journey, specifically using a non-native language.

My deep appreciation is presented to Professor Emad Gad, my supervisor, for his comprehensive and continuous support during my graduate studies.

I am also sincerely grateful to to my co-supervisor, Professor Michel Nakhla, for his guidance, which was crucial in many stages of this degree and the thesis. Most of all I wish to thank him for his motivation and encouragements.

Also, my wife whose endless help and support has been the greatest motive to complete this work.

Dedication

To my uncle, Professor ***Taimour Langaee***, whose outstanding years of research in the Pharmacogenomics field have always been the greatest motivation and inspiration.

I also dedicate this to my wonderful wife Roya, and our sons, Amin and Hossein for their endless love, support, and encouragements.

Statement of Originality

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at its point of use.

Table of Contents

List of Tables

List of Figures

List of Symbols

Notation	Description
$\mathbb{I}_\kappa$	An identity matrix of size κ
$\mathbf{e}_i$	The i^{th} column in an identity matrix
$\mathbb{N}$	The field of natural numbers
$\mathbb{R}$	The field of real numbers
$\mathbb{C}$	The field of complex numbers
$\Lambda_{\mathcal{P}}$	A set of multi-indices
$\mathcal{D}\left[f(t);h\right]$	A continues difference operator on a time function $f(t)$
$m_{[n]}$	Pochhammer symbol defined by $(m+n-1)!/(m-1)!$
$\mathbf{u}(t) \in \mathbb{R}^n$	Auxiliary voltage sources
$\mathbf{i}(t) \in \mathbb{R}^n$	Currents in auxiliary sources in the linear partition
$\mathbf{j}(t) \in \mathbb{R}^n$	Currents in auxiliary sources in the nonlinear partition
$\hat{t}$, $(\hat{s})$	Scaled and shifted t, (the Laplace counterpart)
$\mathscr{L}^{-1}\{\cdot\}(t)$	Inverse Laplace transform from s to t domain
$\mathscr{L}^{-1}_{\text{NILT}}\{\cdot\}(t)$	Inverse Laplace transform from s to t domain using NILT
$\hat{\mathbf{i}}_l^{(m)} \in \mathbb{R}^n$	NILT-based computation of $h^m \frac{\mathrm{d}^m \mathbf{i}(t)}{\mathrm{d}t^m}$ at $t = lh$
$\hat{\mathbf{j}}_l^{(m)} \in \mathbb{R}^n$	Rooted trees computation of $h^m \frac{\mathrm{d}^m \mathbf{j}(t)}{\mathrm{d}t^m}$ at $t = lh$

Notation	Description
$\widehat{\mathscr{L}}^{-1}_{\text{NILT}}\{\cdot\}(\hat{t}\,)$	Inverse Laplace from $\hat{s}$ to $\hat{t}$ domain using the NILT method
$\xi_{N,M}(z)$	The N/M rational Padé approximation of the exponential function e^z
$A \otimes B$	Kronecker product of matrices A and B, defined as: $\begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}$

Abbreviations

Notation	Description
ANTLR	ANother Tool for Language Recognition
BDF	Backward Differentiation Formulae
BE	Backward Euler
BJT	Bipolar Junction Transistor
CAD	Computer-Aided Design
CMOS	Complementary Metal-Oxide Semiconductor
CPU	Central Processing Unit
DAE	Differential Algebraic Equations
DC	Direct Current
FE	Forward Euler
HB	Harmonic Balance

Notation	Description
IVP	Initial Value Problem
KVL	Kirchhoff's Voltage Law
LMS	Linear Multi-Step
LTE	Local Truncation Error
LU	Lower–Upper
MNA	Modified Nodal Analysis
MOSFET	Metal-Oxide Semiconductor Field-Effect Transistor
NILT	Numerical Inversion of Laplace Transform
NILTn	Modified and Improved NILT
NILT0	Conventional NILT
NILT1	NILT1
NILT2	NILT2
NR	Newton-Raphson
ODE	Ordinary Differential Equation
OOP	Object-Oriented Programming
PCB	Printed Circuit Board
PEEC	Partial Element Equivalent Circuit

Notation	Description
RF	Radio Frequency
RK	Runge-Kutta
RLGC	Resistance, Inductance, Conductance and Capacitance
RT	Rooted Trees
TL	Transmission Line
TR	Trapezoidal Rule

Chapter 1

Introduction

Computer-Aided Design (CAD) tools have become a central component in the design of modern electronic systems. At any stage of the design process, the deployment of CAD software tools is almost universal. The role that a certain CAD tool plays depends mainly on the target application. For example, utilizing a CAD tool in the context of microwave filter design is oftentimes aimed at characterizing the scattering parameters of a given microwave device. Such a task starts by discretizing the surface and volume of the device structural elements into triangular or tetrahedral elements of small but finite size [1–3]. Subsequently, electric and magnetic fields that satisfy the Maxwell's equations inside and at the boundaries of the computational domain are computed. The mathematical problem formulated from this task is a repeated solution of linear systems of equations whose matrix is very sparse. In other tasks involving circuits with devices whose behavior is modeled by nonlinear equations, the mathematical problem takes the form of a system of nonlinear differential equations whose solution is sought in characterizing the transient behavior of the circuit variables

(voltage and currents) [4–6]. If the task at hand is the characterization of the signal distortion or noise figures in a circuit used at the front-end of a communication system, the problem is transformed into a nonlinear algebraic problem whose solution using the Harmonic Balance (HB) technique, yields the said parameters [7–9].

The principal CAD problem considered in this thesis falls under the category of "the numerical solution of system of nonlinear differential equations", with the objective of obtaining the behavior of circuit variables in the transient phase of the circuit response in the time-domain. The quest of solving differential equations, numerically, predates the emergence of the CAD tools as an essential component in electronic design by decades, going back to the middle of the past century. Despite the considerable maturity of this inquiry, scholarly interest has not diminished; indeed, ongoing endeavors in this domain continue to be evidenced within the broader mathematical body of literature [10–14].

The motivation that spurred the pursuit of this subject in the thesis stemmed from two observations. First, the methodologies ultimately incorporated into commercial tools constitute merely a fraction of the extensive mathematical literature devoted to this topic. Indeed, the differential equations solvers employed in time-domain circuit simulation belong to a class of methods known as the Linear Multi-Step (LMS), with the Gear's method, Trapezoidal Rule (TR) and Backward Euler (BE) being among the most commonly used in the context of circuit simulations [15], mainly due to the ease with which they can handle the differential equations that model large integrated circuits. Notwithstanding the popularity of these methods, it has been a well-established fact that the LMS family inherently suffers from a conflict between the approximation order and the numerical stability [16]. This conflict essentially forces the simulator to use the low-order, and less efficient version of the LMS method, in order to preserve

the numerical stability in the time marching process [17].

The second observation representing the motivation for the work in this thesis is one approach that has been shown to be free from this conflict. That approach is based on the idea of NILT proposed by [18, Ch. 10]. NILT is essentially a time-domain approach that is grounded in the frequency-domain. As a result of this feature, it offers the rare advantage in being more suitable to handle those circuits whose physics are captured more easily in the frequency-domain than in the time-domain. For example, NILT is more natural in dealing with multi-conductor transmission lines since they are concisely captured in the frequency-domain using the exponential matrix formulation [19, 20]. In addition to this feature, the efficiency of NILT compared to conventional transient time-domain simulators, e.g., those based on the Backward Differentiation Formulae (BDF) such as the Gear's method [21], or TR, stems from several factors. First, it enables approximating the circuit waveforms with high-degree polynomials, a fact that allows the step taken in marching through time to be larger than the step afforded by conventional methods, which relies on low-order approximating polynomials to maintain the numerical stability. NILT, on the other hand, is structurally stable regardless of the degree of the underlying approximating polynomial. The stability in this context, mathematically recognized as the L-stability, is another factor that adds to the NILT robustness. L-stability is typically stronger than A-stability [22] and is highly desirable in circuit simulations because it guarantees numerical stability for any type of circuits including stiff circuits and circuits modelled by Differential Algebraic Equations (DAE) [23].

Although NILT was first introduced several decades ago, the recent developments in model-order reduction techniques [24, 25] and the complex full-wave circuit modeling approaches developed using Partial Element Equivalent Circuit (PEEC) [26, 27] have

shed new lights on its efficiency and usefulness, and renewed interest in employing it as an engine in general circuit simulation. In fact, for PEEC circuits, the NILT approach proved to be more reliable and numerically stable compared to the time-stepping approach which is frequently affected by instability [28].

1.1 Objective

The above introduction made it obvious that there are two key advantages of NILT. The first is that NILT is suitable for problems whose modeling can afford a frequency-domain representation that is more compact than its time-domain representation, by virtue of the fact that it necessitates a frequency-domain solution at specific points in the complex domain. The second advantage, on the other hand, is that NILT is provably not bound by the order/stability dichotomy that are found in the traditional LMS solvers. Such a dichotomy impedes boosting the efficiency of these methods in circuit simulation.

Nevertheless, NILT by its very construction and methodology, is an approach that has been mainly confined to linear circuits. This is a fact that stems from its reliance on computing the response of the circuit in the complex or the Laplace s-domain, thus, assuming that the circuit can be formulated in the Laplace-domain, which is only feasible if the circuit does not have any nonlinear devices. The objective of the research conducted in this thesis is to bridge this gap.

It is important to highlight earlier attempts that sought to bridge this gap. For example, one can find works in the literature that succeeded in bridging the gap between the application of NILT and the simulation of nonlinear circuits, e.g. [29–31]. However, those approaches do not take advantage of the full potentials of NILT in terms of

its high-order and stability characteristics which make it superior to the traditional differential equation solvers. Thus, the objective in this work has been to not only link NILT to nonlinear circuit simulations, but to also demonstrate, and prove wherever possible, that the high-order and stability characteristics can be retained.

1.2 Contributions of the Thesis

The first component of the contribution in this thesis is the development of a complete framework that has seen the NILT approach generalized from linear to nonlinear time-domain circuit simulation approach. The pursuit of this work was made while ensuring that the developed approach inherits all the characteristics of the NILT technique that have been demonstrated in its application to linear circuits. This contribution was published in [32, 33].

The pursuit of the research in the above contribution happened to be taking place in concurrence with a new development in the basic NILT framework for linear circuits. It demonstrated that it is possible to increase the accuracy of the conventional NILT by simple reformulation and with roughly the same computational efforts [34]. The new approach for simulation of linear circuits was dubbed NILTn to distinguish it from the conventional NILT which was labeled NILT0. To clarify and emphasize, NILT0 is the foundational NILT variant used in the aforementioned contribution of this thesis.

The maturity and eventual success of NILTn in improving the efficiency of the time-domain simulation was, however, still limited to linear circuit domain. This success established a new direction for the thesis to make new contribution, whose objective is to generalize the NILTn from linear to nonlinear transient circuit simulation in the time-domain. The work on this contribution was published in [35, 36].

1.3 Organization of the Thesis

The subsequent chapters of this thesis are organized as follows,

- Chapter 2 introduces a concise overview of the foundational theories and methods essential for time-domain analysis. This chapter serves as a cornerstone for the development of the proposed novel method by presenting the relevant background information.

- Chapter 3 delves into the methodological details of the proposed approach. It begins by establishing the necessary notation and illustrating the core concept through a simplified example. Subsequently, the chapter expands upon this foundation, deriving the comprehensive mathematical equations required for the analysis of general circuit topologies.

- Chapter 4 investigates the stability of the proposed method through classical test equation while carefully considering the inherent characteristics of the NILT. The chapter leverages relevant lemmas and theorems to rigorously prove the stability properties of the employed numerical method in the complex-domain over the course of simulation time.

- Chapter 5 determines the order and approximation error of the proposed method. This chapter establishes the mathematical relationship that governs the high-order and accuracy of the proposed approach as a numerical integration solver.

- Chapter 6 explores a recent advancement in utilizing traditional NILT for linear circuit simulation. The chapter expands upon the theoretical underpinnings of

the NILTn method, adapting it to address the challenges posed by general nonlinear circuits within the proposed framework. This adaptation leads to a significant improvement in both the accuracy and efficiency of the proposed method.

- Chapter 7 presents several illustrative examples to demonstrate the precision and efficiency of the proposed method as well as its stability and achieved speedup comparing to conventional methods.

- Chapter 8 summarizes the major contributions of this thesis and provides potential directions for future research.

Chapter 2

Background

This chapter offers a concise overview of the theoretical and methodological topics central to the research presented in this thesis. It starts in Section 2.1 by providing the general approach used to formulate nonlinear circuits in the time-domain. It then follows with a brief discussion of the methods used in time-domain circuit simulation, in Section 2.2. A particular emphasis is placed on the technique known as NILT in Section 2.2.3. The NILT method is treated in greater depth since it forms the foundation that is employed in the research ahead. Finally, Section 2.3 presents the motivations behind the research conducted in the thesis through demonstrating a gap in the current methodology.

2.1 Mathematical Formulation of Circuit Equations

The first step in the computer simulation of a general circuit is to assemble a set of equations that model the behaviour of the circuit. This set of equations are formed

from the models of its individual elements. Elements in the circuit are in turn modelled using their constitutive equations. The constitutive equations of the individual circuit elements capture the relation between the terminal voltages, branch currents, charge or flux. The standard approach used to automatically formulate a set of equations for a general circuit is based on the Modified Nodal Analysis (MNA) approach [37]. A brief of this approach is described in next section.

2.1.1 Modified Nodal Analysis: Time-Domain Formulation

The MNA method starts by writing the summation of currents, i, for all circuit's nodes,

$$\sum_{k=1}^{j_n} i_{k,n} = 0 \quad , \quad n = 1, \ldots, N \tag{2.1}$$

where j_n is the number of branches at node n, and N is the total number of circuits nodes. For simple elements such as resistors, capacitors, and inductors their constitutive relations like Ohm's Law and current-voltage relationship formulas are used to express the currents in terms of node voltages and their derivatives. Next, using Kirchhoff's Voltage Law (KVL), all branch voltages are replaced by node voltages. Through this process, each type of circuit element is associated with so-called "stamp" which identifies its contribution to the system equations (2.1). The MNA system of equations representing the circuit is then built up by accumulating the stamps from all circuit elements. Upon completing this process, the system of MNA equations takes the form of a set of K nonlinear DAEs, typically represented by,

$$\mathbf{G}\mathbf{x}(t) + \mathbf{C}\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} + \mathbf{f}\left(\mathbf{x}(t)\right) = \mathbf{b}(t) \tag{2.2}$$

where

- $\mathbf{C}, \mathbf{G} \in \mathbb{R}^{K \times K}$ are sparse matrices that describe the memory and memoryless elements in the circuit,
- $\mathbf{x}(t) \in \mathbb{R}^K$ is a vector of circuit nodes voltages and currents in inductors and voltage sources,
- $\mathbf{f}(\mathbf{x}(t))$ is a vector that captures the elements whose constitutive equations are nonlinear,
- $\mathbf{b}(t) \in \mathbb{R}^K$ is a vector representing the independent voltages and currents stimuli, and
- K is the size of the MNA formulation.

The key advantages of using MNA in time-domain analysis is its ability to handle complex circuits with multiple energy storage elements, such as capacitors and inductors. By including the time-dependent behavior of these elements, MNA provides a comprehensive framework for analyzing transient responses and steady-state conditions. This makes it particularly useful for studying the dynamic performance of circuits, such as in signal processing, power electronics, and communication systems.

2.1.2 Modified Nodal Analysis: Laplace Domain Formulation

The Laplace transform is defined by the following integral:

$$F(s) = \mathcal{L}\{f\}(s) = \int_0^\infty f(t)e^{-st}\mathrm{d}t \tag{2.3}$$

where $s = \sigma + j\omega$ is a complex value with σ and ω being real numbers.

The Laplace transform, being a linear integral transformation operation, converts linear operations in the time-domain into linear operations in the Laplace domain (also known as the s-domain). On the other hand, nonlinear operations in the time-domain do not have a well-defined corresponding representation in the s-domain that can be described analytically in closed-form. This rather inconvenient fact makes it difficult to have a Laplace-based representation of the MNA formulation of a general circuit with arbitrarily defined nonlinear elements. Thus, to apply the Laplace transform on the MNA circuit formulation, the circuit should not have elements with nonlinear constitutive equations, or in other words $\mathbf{f}(\mathbf{x}(t))$ should be null. Applying the Laplace transform on (2.2) (assuming that $\mathbf{f}(\mathbf{x}(t)) = \mathbf{0}$), results in

$$\mathbf{G}\mathbf{X}(s) + s\mathbf{C}\mathbf{X}(s) = \mathbf{B}(s) + \mathbf{C}\mathbf{x}(0) \tag{2.4}$$

where $\mathbf{X}(s) \in \mathbb{C}^K = \mathscr{L}\{\mathbf{x}(t)\}$ is the vector of unknowns, $\mathbf{B}(s) \in \mathbb{C}^K = \mathscr{L}\{\mathbf{b}(t)\}$ is the input vector, $\mathbf{x}(0)$ is the initial conditions for $\mathbf{x}(t)$ at $t = 0$.

The fact that nonlinear elements do not allow the application of the Laplace-transform on the MNA equations is the key motivation that instigated the contributions presented in this thesis.

2.1.3 Modified Nodal Analysis: A Circuit Example

The notion of element's stamp that is used to build up the MNA formulation is further illustrated in this section through a specific example. Table 2.1 lists a sample of the stamps of components that are frequently used in circuit modeling. For example, the

admittance of a resistor R connected between two nodes i and j is added to the elements i, j of matrix $\mathbf{G}$ where $i = j$, and subtracted from the elements i, j of matrix $\mathbf{G}$ where $i \neq j$. A complete discussion about stamps and related tables can be found in [18, Ch. 4].

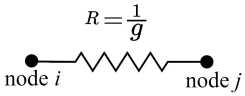
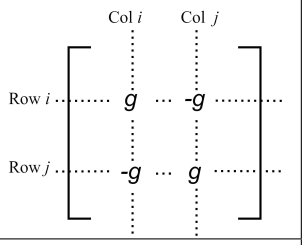
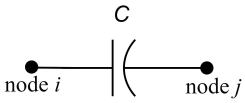
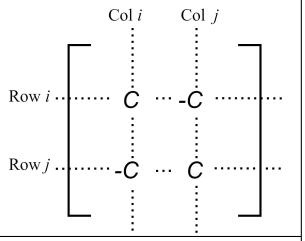
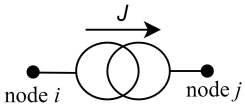
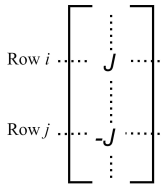
Element	Schematic symbol	Stamp into:
Resistor		**G matrix:** 
Capacitor		**C matrix:** 
Current source		**b matrix:** 

Table 2.1: Component stamps for the example of Figure 2.1

The circuit shown in Figure 2.1 is an example to illustrate the use of the elements stamps in constructing the MNA formulation. For this circuit, let $R_1 = \frac{1}{g_1}$, $R_2 = \frac{1}{g_2}$, and a capacitor given as C_1. This circuit is excited by the current source J. Using component stamps, the circuit equations in the matrix form in the Laplace domain are

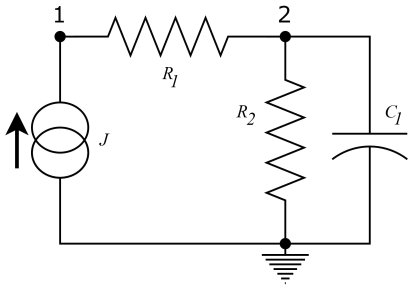


Figure 2.1: Simple RC circuit

written as,

$$\left(\underbrace{\begin{bmatrix} g_1 & -g_1 \\ -g_1 & g_1+g_2 \end{bmatrix}}_{\mathbf{G}} + s \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & C_1 \end{bmatrix}}_{\mathbf{C}} \right) \underbrace{\begin{bmatrix} V_1(s) \\ V_2(s) \end{bmatrix}}_{\mathbf{X}(s)} = \begin{bmatrix} J(s) \\ 0 \end{bmatrix} \tag{2.5}$$

Figure 2.2 shows a different circuit example with a nonlinearity represented by the presence of a diode element with the constitutive current-voltage equation of $i_D(t) = I_s(e^{v_2(t)/V_T} - 1)$, where I_s is the diode saturation current and V_T is the voltage equivalent of the thermal energy at the junction of a diode.

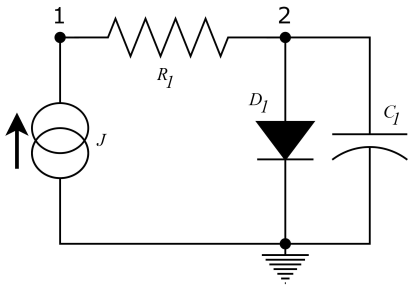


Figure 2.2: A circuit with diode

As mentioned above, the nonlinearity allows only time-domain circuit equations

which in this case becomes,

$$\begin{bmatrix} g_1 & -g_1 \\ -g_1 & g_1+g_2 \end{bmatrix} \begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & C_1 \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}v_1(t)}{\mathrm{d}t} \\ \frac{\mathrm{d}v_2(t)}{\mathrm{d}t} \end{bmatrix} + \begin{bmatrix} 0 \\ I_s(e^{v_2(t)/V_T}-1) \end{bmatrix} = \begin{bmatrix} J \\ 0 \end{bmatrix} \quad (2.6)$$

Clearly, the Laplace transform cannot directly handle the exponential relationship in the above equation because it is not a linear operation.

2.2 Time Domain Simulation

Time-domain circuit simulation involves examining how electrical circuits react to various inputs over time. The primary focus is transient analysis which is essential for understanding the dynamic behavior of circuits, especially when dealing with transient phenomena like switching events, signal propagation, startup sequences, and responses to time-varying signals and other dynamic conditions that cannot be adequately studied using frequency-domain methods. Another significant advantage of time-domain simulation is the ability to model nonlinear components and their complex interactions within a circuit. Many real-world circuits include elements like diodes, transistors, and operational amplifiers which exhibit nonlinear behavior. Time-domain simulation can accurately represent these nonlinearities and their impact on circuit performance. This is particularly important in the design of analog and mixed-signal circuits, where precise modeling of component behavior is crucial for achieving desired performance specifications.

2.2.1 Basic Concepts

Time-domain simulation is the process of solving the MNA system of DAEs that models the circuit behaviour. Unlike Ordinary Differential Equation (ODE) systems, there is no theorem that guarantees availability and uniqueness of a solution for DAE systems [15]. Therefore, numerical methods are needed to approximate the solution at discrete time points.

This section presents a wide overview of various numerical methods that have been used to solve a system of differential equations numerically. The presentation of these methods shall be illustrated through their application in approximating the solution to a general nonlinear problem represented by

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t), t) \tag{2.7}$$

where $f(t)\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a general nonlinear function.

A generic multi-step method that is used to approximate the solution $x(t)$ at discrete time points t_n; $n = 1, 2, 3, \cdots$ may be expressed in the form

$$\begin{aligned} x_{n+1} = \phi \Big(& x^{(0)}_{n+1}, x^{(0)}_{n}, \cdots, x^{(0)}_{n+1-k_0}, \\ & x^{(1)}_{n+1}, x^{(1)}_{n}, \cdots, x^{(1)}_{n+1-k_1}, \\ & x^{(2)}_{n+1}, x^{(2)}_{n}, \cdots, x^{(2)}_{n+1-k_2}, \\ & \vdots \\ & x^{(p)}_{n+1}, x^{(p)}_{n}, \cdots, x^{(p)}_{n+1-k_p}, t_n; h_n \Big) \end{aligned} \tag{2.8}$$

where

- ϕ is a general function,
- $x_m^{(0)}$ is an approximation to $x(t)$ at $t = t_m$,
- $x_m^{(q)}$ is an approximation to $\frac{\mathrm{d}^q x(t)}{\mathrm{d}t^q}$ at $t = t_m$ with $q \geq 1$, and
- $h_n = t_{n+1} - t_n$.

The method is said to be a linear method if the function ϕ is a linear function of its arguments. It is also labeled "**explicit**" if ϕ *does not depend* on $x_{n+1}^{(m)}, m > 0$. Otherwise, it is labeled "**implicit**". The label that is used in case that $k_0 = k_1 = k_2 = \cdots = 1$ is a "**single-step**" and "**multi-step**" if at least one $k_i > 1$. The general class of methods expressed is known as an Initial Value Problem (IVP) if $x_n, \cdots, x_{n-k}$ are available via some starting (initialization) procedure. For example, $x_0 = x(t_0)$ is obtained as the quiescent point where $\frac{\mathrm{d}x(t)}{\mathrm{d}t} = 0$. Finally, an implementation of the method can be carried out by fixing h_n for all discrete time points, or adapting the length of step according to the conditions and rate of variation observed from computing x_{n+1}. The former case is known as "**fixed step size**", while the latter is known as "**variable step size**" implementation.

Two key characteristics are essential for a method to be suitable for circuit simulation. Those are **accuracy** and **stability**.

In essence, the **accuracy** of a method focuses on how closely the approximation x_n approximates the exact value $x(t_n)$ and how it is influenced by the time step size h_n.

Stability analysis, on the other hand, and broadly speaking, examines how errors introduced at a given step affect the errors at subsequent steps.

In the context of circuit simulation, single-step methods are the easiest to start, since they require only the value of $x(t)$ at $t = 0$. They are also the simplest to

implement within variable step length scheme. Explicit methods are the least complex in the sense of computational efforts associated with their implementation. However, they are generally weaker than implicit methods in terms of stability performance. More precise characterization of stability and accuracy will be provided in reviewing specific examples of methods used in practice.

2.2.1.1 Accuracy and Order of the Method

To simplify the notation used in this chapter, we will employ $x_{[\mathbf{k}_p]}$ to denote the set of x arguments in the function ϕ,

$$\mathbf{k}_p = [k_0, k_1, \cdots, k_p],$$

where k_i are non-negative integers. We will use $\bar{x}_{[\mathbf{k}_p]}$ when there is a need to refer to the corresponding exact values, that is, to $x(t_n)^{(q)}$ instead of $x_n^{(q)}$.

The accuracy of a method is characterized locally in the sense that approximations from the past time points $t \leq t_n$ are assumed to be exact. Thus, it is assumed that

$$x_{n+1-k_i}^{(q)} = x(t_{n+1-k_i})^{(q)} \tag{2.9}$$

with $k_i \geq 1$. Subsequently, those exact values are substituted in ϕ, i.e., $\phi\left(\bar{x}_{[\mathbf{k}_p]}; h\right)$, to obtain the approximation x_{n+1}, which is subsequently compared to the exact $x(t_{n+1})$ to determine the accuracy of the method. The residual error found in $x(t_{n+1}) - \phi\left(\bar{x}_{[\mathbf{k}_p]}\right)$ would then serve as the measure for the accuracy. This notion is formalized by defining a continuous operator $\mathcal{D}\left[\bar{x}_{[\mathbf{k}]}; h\right]$ that operates on a continuous function of t, and on delayed versions from it and its derivatives, to produce another continuous function of

time. The operator in this case is given by

$$\mathcal{D}\left[\bar{x}_{[\mathbf{k}]}; h\right] = \phi\left(\bar{x}_{[\mathbf{k}_p]}; h\right) \tag{2.10}$$

where it is assumed that the method is implemented with fixed step size h. The accuracy of the method is then examined by writing the operator using a Taylor-like series in h

$$\mathcal{D}\left[\bar{x}_{[\mathbf{k}]}; h\right] = C_0 x(t_{n+1}) + C_1 h x^{(1)}(t_{n+1}) + C_2 h^2 x^{(2)}(t_{n+1}) + \cdots \tag{2.11}$$

It should be obvious that the method is exact if all C_i coefficients vanish. Alternatively, the method is more accurate if more of the first coefficients, $C_0, C_1, C_2, \cdots$ are identically zero. This notion motivated the definition of the **order** of a given method as the integer p such that $C_i = 0$ for $i \leq p$ whereas $C_{p+1} \neq 0$.

Due to truncation of the Taylor series expression, an error may be introduced at every step. This is called Local Truncation Error (LTE), also known as one-step error. The LTE quantifies the discrepancy between the true solution of a differential equation and the solution generated by a single iteration of a numerical method, under the idealized condition that the prior step's value is exact. Assuming moving from t_n to t_{n+1} of a numerical method of order p, then LTE is defined as,

$$\text{LTE} = x(t_{n+1}) - \tilde{x}_{n+1} \tag{2.12}$$

where $\tilde{x}_{n+1}$ is the value computed by the numerical method and $x(t_{n+1})$ is the exact solution at t_{n+1}.

2.2.1.2 Concepts of Stability

The stability of the method is typically studied by characterizing its behaviour in approximating the solution of a given test problem. The test problem is usually selected to be a simple differential equation, whose exact solution is often known beforehand. The test problem often used is given by

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \lambda x(t), \quad x(0) = x_0 \tag{2.13}$$

where λ is a complex scalar. The similar test equation sometimes used in the literature is formed from a system of ordinary differential equations of the form,

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{A}\mathbf{x}(t), \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{2.14}$$

where $\mathbf{A}$ is a matrix of a given size. Obviously, when the matrix $\mathbf{A}$ is diagonalizable, then (2.14) can be reduced to a decoupled equations of the form (2.13).

The exact solution to the above test equation is given by $x(t) = \mathrm{e}^{\lambda t}$, with the property that $x(t) \to 0$ if $\Re(\lambda) < 0$. The stability of a given method is then viewed from the angle of how the successive approximations generated by a given method to that test problem, or x_n, align with the exact solution.

Applying a numerical method to the test problem results in a recurrence relation. For example, in the case of a single-step method, the recurrence takes the form

$$x_{n+1} = R(h\lambda)x_n, \tag{2.15}$$

where $R(z)$ is known as the **stability function** of the method.

The numerical solution remains bounded, and thus in some sense reflects the decaying nature of the exact solution, if

$$|R(h\lambda)| \leq 1. \tag{2.16}$$

The set of values $z = h\lambda$ for which $|R(z)| \leq 1$ defines the **region of absolute stability** of the method.

A method is called **A-stable** if its absolute stability region includes the entire left-half complex plane $\Re(z) \leq 0$. This means the method remains stable for all negative real eigenvalues, which is crucial for stiff ODEs [1].

An **L-stable** method is a stronger form of an A-stable method. In addition to being A-stable, an L-stable method satisfies:

$$\lim_{z \to -\infty} R(z) = 0. \tag{2.17}$$

This means that for very large negative values of z (which correspond to very stiff problems), the numerical solution damps out rapidly, preventing spurious oscillations or slow decay [38].

Another concept of stability is known as $A(\alpha)$-stability and is defined when a

[1] For linear systems, a system of differential equations is termed stiff if the ratio between the largest and the smallest eigenvalues of the system matrix is large. Stiff systems have some poles close to the origin of coordinates and some poles very far away from them, all in the left half plane. They can cause numerical difficulties in simulations due to their wide range of time scales. In the context of circuit simulations, stiffness often arises in systems with components that have vastly different time constants, such as circuits with both very fast and very slow dynamics. For example, a circuit might include both high-frequency switching elements and low-frequency filters.

method has a region of absolute-stability given by

$$\{\mu\colon\ |\arg(-\mu)| < \alpha,\ \mu \neq 0\}\,,$$

for $0 < \alpha < \pi/2$ [23]. $A(\alpha)$-stability is a weaker condition than A-stability. It recognizes that sometimes, full A-stability is too restrictive. A numerical method is $A(\alpha)$-stable if its region of absolute stability contains a wedge-shaped region in the left half of the complex plane, defined by the angle α.

2.2.2 Examples of Numerical Methods

Having elaborated on key concepts related to accuracy and stability of methods used in numerical solution of differential equations, we now list a few of the main methods used in circuit simulations and analyze them in the light of the above-mentioned concepts.

2.2.2.1 Linear Multi-Step Methods

LMS methods use a linear form of the function ϕ, with $k_0 = k_1 = k$ and $k_i = 0, i > 1$ thereby casting the method equation (2.8) as

$$\sum_{j=-1}^{k-1} \alpha_j x_{n-j} = h \sum_{j=-1}^{k-1} \beta_j x_{n-j}^{(1)} \quad ; \quad |\alpha_{k-1}| + |\beta_{k-1}| \neq 0 \tag{2.18}$$

where α_i, β_i are coefficients that are determined as shown below.

To find the coefficients, α_i, β_i, the operator $\mathcal{D}(\cdot)$ defined in (2.10) is first customized

for the method (2.18). In this method, the operator reads,

$$\mathcal{D}[y(t);h] = \sum_{j=-1}^{k-1} \alpha_j x(t-jh) - h \sum_{j=-1}^{k-1} \beta_j x'(t-jh) \tag{2.19}$$

Expanding $x(t-jh)$ and $x'(t-jh)$ in Taylor series at $t=0$ and rearranging the terms will result in,

$$\mathcal{D}[y(t);h] = C_0 y(t) + C_1 h y'(t) + \cdots + C_q h^q y^{(q)}(t) + \ldots \tag{2.20}$$

It is straightforward but a tedious task to express the coefficient C_i in terms of the coefficients α_i, β_i. Doing so allows computing the coefficients α_i, β_i and leads to building different LMS method with different orders.

The most popular methods used in circuit simulation, are TR and the BDF methods. The remainder of this section lists some of the well-known variants of the LMS methods discussing their stability criteria.

2.2.2.2 Single Step LMS

The following famous single step methods can be derived from (2.18) with $k=1$ and setting all coefficients α_j, β_j to zero except the ones that are listed for each method.

- **Forward Euler (FE)**,

  The FE method is obtained by setting $\alpha_0 = -1$, $\alpha_{-1} = 1$, $\beta_0 = 1$, and all other α_i and β_i set to 0, resulting in the integration formula

$$x_{n+1} = x_n + h x_n^{(1)} \tag{2.21}$$

It can be shown that the FE results in having the Taylor coefficients of the operator $\mathcal{D}[y(t);h]$ being $C_0 = C_1 = 0$ while $C_2 = 0.5$ with $C_i \neq 0$ for $i > 2$ indicating that the method is of order 1 and its LTE is of order 2.

To study the stability of the FE method, we use it to approximate the solution to the linear test problem in (2.13). Applying FE method to the stability test equation will yield

$$x_{n+1} = R(z)x_n$$

where the stability function $R(z)$ is given by

$$R(z) = (1 + z)$$

with $z = h\lambda$. For the method to be stable, we require that

$$|R(z)| = |1 + z| \leq 1. \tag{2.22}$$

making stability region confined to values of $z = h\lambda$ such that:

$$-2 < h\lambda < 0. \tag{2.23}$$

Therefore the region of stability is shown in Figure 2.3. The FE method is an explicit method.

- **Backward Euler (BE)**.

  In the BE method, $\alpha_{-1} = 1, \alpha_0 = -1, \beta_{-1} = 1$, with all other α_i and β_j set to 0. Similar to the FE method, $C_0 = C_1 = 0$ and $C_2 = -0.5$ indicating the method is of order 2, with LTE equal to (in magnitude) 0.5. However, and unlike the FE

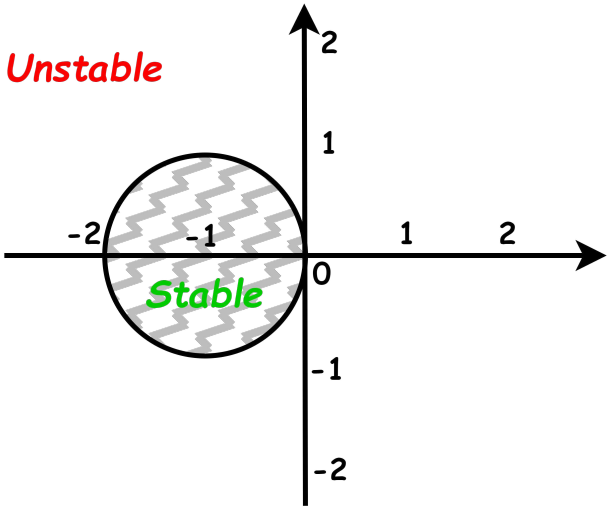


Figure 2.3: FE's stability region in $h\lambda$ complex plane

method, it is an implicit method due to the existence of a derivative at the next time step.

$$x_{n+1} = x_n + hx^{(1)}_{n+1} \tag{2.24}$$

By applying BE method to the stability test equation, we will have a stability function given by

$$R(z) = \frac{1}{1-z}$$

with $|R(z)| = \left|\frac{1}{1-z}\right| \leq 1$ which holds for all $\Re(z) > 0$, meaning BE is A-stable.

Also, BE is an L-stable method because if $h\lambda \to \infty$ then $|R(z)| \to 0$. Figure 2.4 shows the stability domain for the BE method.

- **Trapezoidal Rule (TR)**

  The TR method is derived from the LMS methods by setting $\alpha_{-1} = 1, \alpha_0 =$

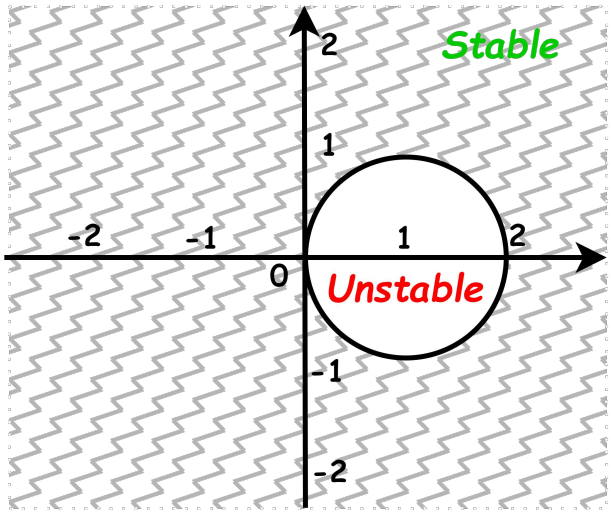


Figure 2.4: BE's stability region in $h\lambda$ complex plane

$-1, \beta_{-1} = 0.5, \beta_0 = 0.5$ with all α_i, β_i set to 0, leading to the integration formula,

$$x_{n+1} = x_n + \frac{h}{2}(x^{(1)}_{n+1} + x^{(1)}_n) \tag{2.25}$$

It can be shown that for TR, we have $C_0 = C_1 = C_2 = 0$ while $C_3 = \frac{-1}{12}$, making the method of order 2 with an LTE of order 3.

Applying it to the test equation yields a stability function given by

$$R(z) = \frac{1 + \frac{z}{2}}{1 - \frac{z}{2}}, \quad \text{where } z = h\lambda.$$

Since $R(z)$ above is a Möbius transformation [39], it satisfies the property that it maps the left-half complex plane onto the unit disk, meaning the TR is A-stable. This makes it a useful method for solving stiff differential equations while maintaining second-order accuracy. However, the TR is not L-stable since

$\lim_{z\to\infty}(|R(z)|) \to 1$.

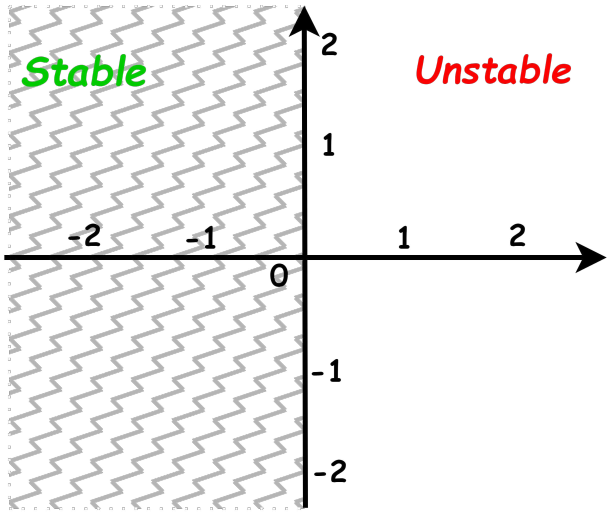


Figure 2.5: TR's stability region in $h\lambda$ complex plane

In general, since both BE and TR are implicit methods, i.e. including derivative of the next time step, applying them to DAEs would result nonlinear equations which then must be solved using iterative methods at each time step.

2.2.2.3 Backward Differentiation Formulas (BDF)

A class of implicit integration method could be derived from (2.18) by setting $k = 1, \ldots, 6$, $\beta_{-1} \neq 0$, all other $\beta_j = 0$, and following the same steps as above to compute the non-zero coefficients.

For $k = 1$, the result is the BE method and for $k = 2$ it yields the following second-order two steps method,

$$x_{n+1} = \frac{-1}{3}x_{n-1} + \frac{4}{3}x_n + \frac{2h}{3}x'_{n+1} \tag{2.26}$$

This equation is called BDF2 and is also famous as 2^{nd}-order Gear method. Higher-order BDF methods are more accurate, although need more complicated time-stepping algorithms.

For a linear multi-step method like BDF, we analyze stability using the test equation (2.13). Applying a BDF method to this ODE transforms the difference equation into one that depends on both the method's coefficients and the term $h\lambda$, that is

$$(\alpha_{-1} - h\lambda\beta_{-1})x_n + \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \cdots + \alpha_{k-1} y_{n-k+1} = 0. \tag{2.27}$$

This is a k-th order linear difference equation where the coefficients depend on $h\lambda$. Assume a solution $x_n = r^n$:

$$(\alpha_{-1} - hb\lambda)r^n + \alpha_1 r^{n-1} + a_2 r^{n-2} + \cdots + \alpha_{k-1} r^{n-k} = 0. \tag{2.28}$$

Factoring out r^{n-k}:

$$r^{n-k}\left[(a_0 - hb\lambda)r^k + a_1 r^{k-1} + a_2 r^{k-2} + \cdots + a_k\right] = 0. \tag{2.29}$$

The characteristic polynomial becomes:

$$p(r) = (a_0 - hb\lambda)r^k + a_1 r^{k-1} + a_2 r^{k-2} + \cdots + a_k = 0. \tag{2.30}$$

For the method to be stable, all roots r_i of this polynomial must satisfy $|r_i| < 1$ (inside the unit circle) for all $h\lambda$ in the left-half plane. The stability region is the set of $h\lambda$ values in the complex plane where this holds.

Figure 2.6 depicts the stability region of all BDF methods where the stability region

for each BDF method is outside of its closed shape (similar to BE in Figure 2.4). As can be seen, BDF3 to BDF6 are not A-stable, however similar to BE, the BDF2 method is L-stable.

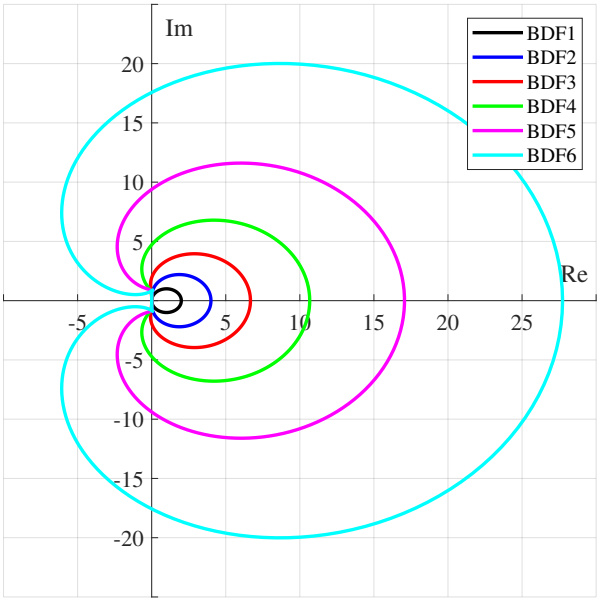


Figure 2.6: Stability regions of 6 BDF methods in $h\lambda$ complex plane

2.2.2.4 Single-step Multi-Stage Methods

The idea behind designing these methods is to improve accuracy and stability by using multiple intermediate stages within a single time-step that do not rely on previous time steps which makes them different from multi-step methods. Within each time step, these methods perform several intermediate calculations to improve the accuracy of the final result. Each stage involves evaluating the differential equation at different points within the time step. The final solution at the next time step is obtained by

taking a weighted average of these intermediate values which helps to achieve higher accuracy.

The most famous single-step multi-stage methods are the Runge-Kutta (RK) methods which include both explicit and implicit methods. The general form of an s-stage RK method is,

$$x_{n+1} = x_n + h \sum_{i=1}^{s} b_i k_i \tag{2.31}$$

where k_i are the stage values to be computed from,

$$k_i = f\left(t_n + c_i h, x_n + h \sum_{j=1}^{m} a_{ij} k_j\right) \tag{2.32}$$

where h is the time step. If $m = i - 1$, (2.32) provides an explicit method and if $m = s$ it provides an implicit method. Coefficients a_{ij}, b_i and c_i are usually arranged in a *Butcher tableau* [40] and define the specific RK method,

$$\begin{array}{c|ccccc} 0 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \dots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \dots & b_{s-1} & b_s \end{array} \tag{2.33}$$

Common condition for b_i is $\sum_{i=1}^{s} b_i = 1$ and for c_i is $\sum_{j=1}^{i-1} a_{ij} = c_i$, $i = 2, \dots, s$. Equations (2.31) and (2.32) with the tableau $\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$ will generate FE the method as the simplest

explicit RK method, and with the tableau $\begin{array}{c|c} 0 & 1 \\ \hline & 1 \end{array}$ will generate the BE method as the simplest implicit RK method.

Another famous example is the classical 4^{th}-order Runge-Kutta method (RK4) that uses four stages with LTE of order 5. The RK4 is a popular explicit method due to its balance between accuracy and computational effort. It involves the following four stages,

1. $k_1 = f(t_n, x_n)$
2. $k_2 = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1)$
3. $k_3 = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2)$
4. $k_4 = f(t_n + h, x_n + hk_3)$

The solution at the next time step is then given by:

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.34}$$

Stability analysis involves applying the RK method to the test equation and analyzing the resulting difference equation. The stability region of the first four RK methods are depicted in Figure 2.7 which are limited to the inside of closed areas.

Implicit RK methods are preferred for stiff problems because of their large stability regions. They allow for larger time steps without sacrificing stability. It should be noted that not all implicit RK methods are A-stable. Ehle [41] has shown that s-stage fully implicit RK methods of order $2s$ developed in [42] are all A-stable. However, there are implicit methods that are L-stable too like Radau-IIA [43]. The main drawback

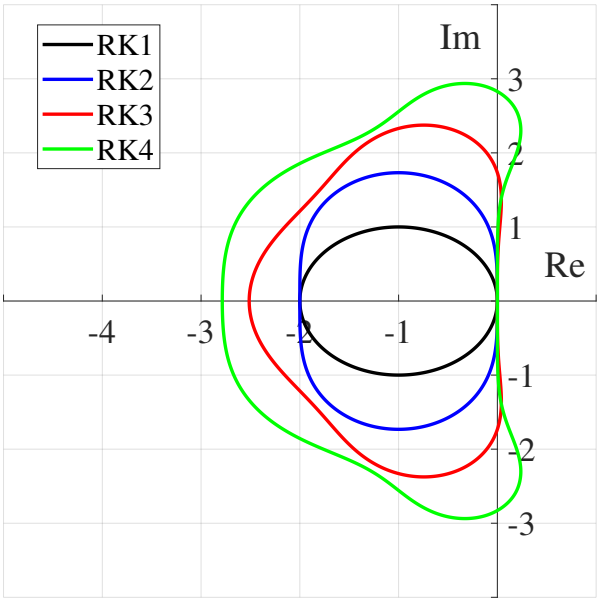


Figure 2.7: Stability regions of explicit RK methods in $h\lambda$ complex plane

of these implicit methods is the need to solve nonlinear equations at each time-step, which can be computationally expensive. However, their superior stability properties often justify this cost for stiff problems.

2.2.2.5 Single-Step Multi-Derivative Method

The basic idea of this method was introduced by Nikola Obreshkov. It is based on using a generalized single-step integration method that includes higher-order derivatives [44],

$$\sum_{i=0}^{j}(-1)^i\alpha_{k,j,i}h^i x_{n+1}^{(i)} = \sum_{i=0}^{k}\alpha_{k,j,i}h^i x_n^{(i)} \tag{2.35}$$

The coefficients $\alpha_{k,j,i}$ in (2.35) are computed by substituting for x_n with $x\left(t_n\right)$ and x_{n+1} with $x(t_{n+1})$ in the operator (2.10), expanding in Taylor series around $t = t_n$, and

equating similar powers of h to obtain [21],

$$\alpha_{k,j,i} = \frac{(k+j-i)!k!}{(k+j)!i!(k-i)!} \tag{2.36}$$

Obreshkov showed that x_n obtained from (2.35) approximates $x(t_n)$ to order $O(h^{i+j})$, i.e., $x_{n+1} = x\left(t_{n+1}\right) + O(h^{i+j+1})$.

In [45], a modified Obreshkov formula was introduced to improve its stability characteristics as,

$$\sum_{i=0}^{m} \alpha_{i,l,m}(-1)^i h^i x_n^{(i)} = \sum_{i=0}^{l} \alpha_{i,l,m} h^i x_{n-1}^{(i)} \tag{2.37}$$

where l and m are integers to define stability and order of a specific integration equation. In contrast to RK methods which require complex computation to find the method coefficients, $\alpha_{i,l,m}$ are predetermined by the following equation,

$$\alpha_{i,l,m} = \frac{(m+l-i)!}{(m+l)!}\binom{m}{i} \tag{2.38}$$

The modified Obreshkov method has been proven to be A-stable and L-stable for $m-2 \leq l \leq m$ with the error constant being defined by [45],

$$C_{l,m} = (-1)^m \frac{l!\,m!}{(l+m)!(l+m+1)!} \tag{2.39}$$

and it can achieve a high order of convergence, which means that the error decreases rapidly as the step size is reduced. This is particularly beneficial for problems requiring high precision [46]. The LTE error constants of the modified Obreshkov method and BDF of orders 2 to 6 are listed in Table 2.2 for comparison.

As will be shown in Chapter 4, the Obreshkov formula will play a central role in

Table 2.2: Local truncation error constants of BDF and modified Obreshkov

Method	Order				
	2	3	4	5	6
BDF	0.222	0.136	0.096	0.073	0.058
Obreshkov	0.083	0.014	1.40e-3	1.39e-4	9.92e-6

the stability of the NILT-based method proposed in this thesis.

2.2.3 Numerical Inversion of the Laplace Transform (NILT)

One approach that has shown a great potential in handling time-domain simulation of complex circuits is based on the idea of NILT [18, Ch. 10]. NILT is essentially a time-domain approach that is fundamentally grounded in the frequency-domain. As a result of this feature, it offers the rare advantage in being more suitable to handle those circuits whose physics are captured more easily in the frequency-domain than in the time-domain.

2.2.3.1 Principles of NILT

Like other integration methods, the goal of using NILT in time-domain simulation is to approximate $x(t)$ at specific time intervals $t = lh, l = 0, 1, 2, \ldots$ where h is step size. The NILT framework achieves this by leveraging concepts from the inversion of the Laplace transform, along with the Cauchy residue theorem, as described next.

NILT starts by putting MNA formulation in the Laplace domain as in (2.4). The *exact* solution $\mathbf{x}(t)$ is recovered using the inverse Laplace domain operation given by

the infinite line integral

$$\mathbf{x}(t) = \mathscr{L}^{-1}\{\mathbf{X}(s)\} = \frac{1}{2\pi\jmath}\int_{c-\jmath\infty}^{c+\jmath\infty}\mathbf{X}(s)\mathrm{e}^{st}\mathrm{d}s \tag{2.40}$$

where $\jmath = \sqrt{-1}$ and c is a constant chosen such that all singularities in $\mathbf{X}(s)$ lie on the left of the vertical line $\Re(s) = c$, with $\Re(\cdot)$ denoting the real part of a complex number. The integral in (2.40) is a line integration performed in the complex domain along infinite vertical line $\Re(s) = c$.

The NILT approach proceeds by replacing st with z, e^z with its $[N/M]$ Padé approximant $\xi_{N,M}(z)$ [47, 48] and then, using the Cauchy theorem of residues, exchanges the line integral with a finite summation of the residues of $\mathbf{X}\left(\frac{z}{t}\right)\xi_{N,M}(z)$ at the poles of $\xi_{N,M}(z)$. Those steps, combined, and denoted as $\mathscr{L}^{-1}_{\mathrm{NILT}}\{\mathbf{X}(s)\}(t)$, provide an approximation, $\tilde{\mathbf{x}}(t)$, to $\mathbf{x}(t)$ defined through

$$\tilde{\mathbf{x}}(t) = \mathscr{L}^{-1}_{\mathrm{NILT}}\{\mathbf{X}(s)\}(t) \coloneqq -\frac{1}{t}\sum_{i=1}^{M/2} 2\Re\left(k_i\mathbf{X}\left(\frac{z_i}{t}\right)\right) \tag{2.41}$$

for even M, with k_i and z_i, $i = 1, \cdots, M$ being, respectively, the residues and poles of the partial fraction expression of $\xi_{N,M}(z)$. Note that for even values of M, poles and residues appear in complex-conjugate pairs, and for odd M, only one residue-pole duo (k_0, z_0) is real, with the rest being in complex-conjugate pairs. In the rest of this thesis, M will be assumed even, knowing that the case of odd M can be treated analogously. The theory underlying the formulation of (2.41) is detailed in [18, Ch. 10].

Figure 2.8 depicts important steps and elements of NILT method. The process involves solving the Laplace domain equations at *limited number of points* in the s-domain and then combining the solutions to generate the time-domain waveform. It is

worth emphasizing that:

- The Laplace domain points are predetermined and do not depend on the problem. The number of points is typically between 2 to 6 depending on the required order for the NILT.
- The combination coefficients k_i are also predetermined, which means that they are independent of the circuit being simulated or the system of equations representing them.

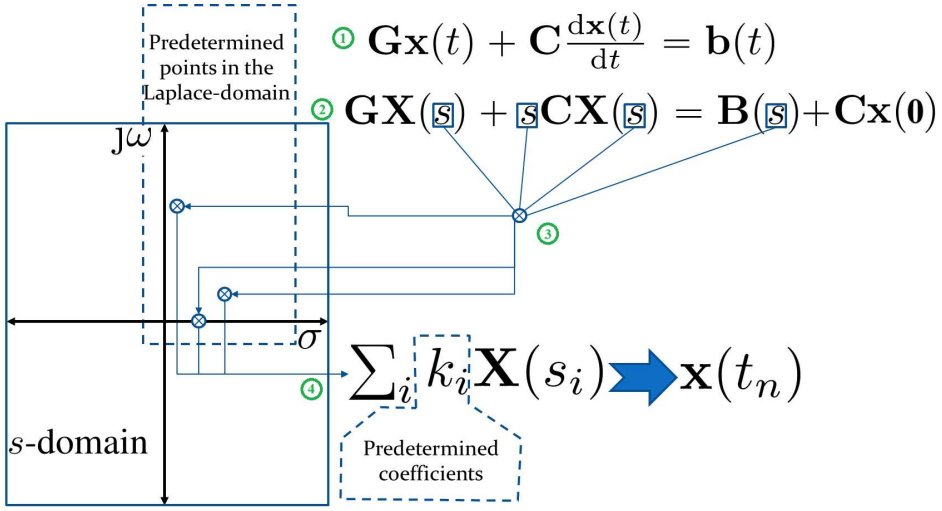


Figure 2.8: NILT block diagram steps

The NILT method can be implemented and deployed in two different modes that are explained in the following sections.

2.2.3.1.1 NILT in Linear Circuits: Initial-time mode

Implementation of NILT in (2.41) is known as the *initial time mode*, where the initial condition of the circuit at $t = 0$, $\mathbf{x}(0)$, is used in computing $\mathbf{X}(s)$ from (2.4) at $s = \frac{z_i}{t}$,

and then determining the state of the circuit at arbitrary time t. More specifically, $\mathbf{x}(0)$ is used in (2.4) to compute $\mathbf{X}(\frac{z_i}{t})$ in order to obtain the approximation $\tilde{\mathbf{x}}(t)$. This mode of implementation cannot be used for arbitrarily large values of t since the approximation of the exponential function e^z through the Padé rational function can be valid for a finite value of t.

2.2.3.1.2 NILT in Linear Circuits: Re-initialized mode

The NILT approach can also be used in a *marching* mode, by allowing the value computed for $\tilde{\mathbf{x}}(t)$ at certain time, say $t = lh$, to be used *as the initial condition* in advancing to a new time point with an appropriate step size h, with $l = 1, 2, \cdots$. This is done by shifting the time origin to $(l-1)h$, and scaling the time t by dividing by h leading to the definition of the new time variable $\hat{t}$ given by,

$$\hat{t} = \frac{t}{h} - (l-1) \tag{2.42}$$

If the Laplace transform is applied with respect to the $\hat{t}$ variable, then the Laplace operator and its inverse will be denoted by $\widehat{\mathscr{L}}\{\cdot\}(\hat{s})$ and $\widehat{\mathscr{L}}^{-1}\{\cdot\}(\hat{t})$, respectively, where $\hat{s} = hs$. Likewise, the NILT-based realization to the latter will be given the notation $\widehat{\mathscr{L}}^{-1}_{\text{NILT}}\{\cdot\}(\hat{t})$ and the value obtained from it will be denoted $\hat{\mathbf{x}}(\hat{t})$. Hence, by this notation we have

$$\tilde{\mathbf{x}}(t = (l-1)h) \equiv \hat{\mathbf{x}}(\hat{t} = 0), \quad \tilde{\mathbf{x}}(t = lh) \equiv \hat{\mathbf{x}}(\hat{t} = 1) \tag{2.43}$$

$$\hat{\mathbf{x}}(1) = \widehat{\mathscr{L}}^{-1}_{\text{NILT}}\left\{\mathbf{X}(\hat{s})\right\}(\hat{t})\Big|_{\hat{t}=1} = -\sum_{i=1}^{M/2} 2\Re\left(k_i \hat{\mathbf{X}}(z_i)\right) \tag{2.44}$$

where $\hat{\mathbf{X}}(z_i)$ is obtained from

$$\hat{\mathbf{X}}(z_i) = \left(\frac{z_i}{h}\mathbf{C} + \mathbf{G}\right)^{-1}\left(\hat{\mathbf{B}}(z_i) + \frac{1}{h}\mathbf{C}\hat{\mathbf{x}}(0)\right) \tag{2.45}$$

and $\hat{\mathbf{B}}(\hat{s})$ is the Laplace transform of $\mathbf{b}\left(h\hat{t} + (l-1)h\right)$ taken with respect to $\hat{t}$ and computed at $\hat{s} = z_i$, while $\tilde{\mathbf{x}}_{l-1}$ is short for $\tilde{\mathbf{x}}(t)$ at $t = (l-1)h$.

The time marching process of computing $\hat{\mathbf{x}}(1)$ from $\hat{\mathbf{x}}(0)$, (respectively, $\tilde{\mathbf{x}}_l$ from $\tilde{\mathbf{x}}_{l-1}$) using (2.44)-(2.45), is also known as re-initializing NILT. In the re-initialization framework, the solution from the previous step serves as the initial condition for the next step.

2.2.3.2 Characteristics of NILT

The unique features of NILT method, which are discussed shortly, are due to its main characteristics: high-order, small truncation error and great stability.

2.2.3.2.1 NILT order

Due to the usage of Padé approximant $\xi_{N,M}(z)$, it is clear that NILT represents an approximation of order $N + M$ for the circuit variables. This combined order $N + M$ is typically much higher than the order achieved by conventional SPICE solvers. Consequently, it allows for an increase in the time step h and significantly reduces the overall computational cost. There are two criteria to consider choosing integers N and M. The first criterion is to ensure that the $\mathbf{X}(s)\xi_{N,M}(s)$ tends to 0 as $s \to \infty$ to allow for the use of the Cauchy theorem of residues. The second criterion is related to its stability which is explained next.

2.2.3.2.2 NILT Truncation Error

The truncation error of the NILT is the result of approximating $\tilde{\mathbf{x}}(t)$ to the exact $\mathbf{x}(t)$ and is defined through the following equation [49, 50],

$$\tilde{\mathbf{x}}(t) - \mathbf{x}(t) = \Psi_{N,M} \left. \frac{d^{N+M+1}}{dt^{N+M+1}} \mathbf{x}(t) \right|_{t=0} t^{N+M+1} + \mathcal{O}\left(t^{N+M+2}\right) \tag{2.46}$$

where,

$$\Psi_{N,M} = \frac{(-1)^M M! N!}{(M+N)!(N+M+1)!} \tag{2.47}$$

Therefore its truncation error is a function of t^{N+M+1} and is directly related to Padé approximation.

2.2.3.2.3 Stability Characteristics of NILT

NILT's stability can be studied by applying Laplace transform to the test equation of (2.13) resulting in,

$$X = \frac{x_0}{s - \lambda} \tag{2.48}$$

using the NILT Laplace inversion formula (2.41) with $t = h$,

$$x_1 = \frac{-1}{h} \sum_{i=1}^{M} k_i \frac{x_0}{\frac{z_i}{h} - \lambda} \tag{2.49}$$

by substituting $z = h\lambda$ in above equation, and after n steps we can write,

$$x_n = x_0 \left(\sum_{i=1}^{M} \frac{k_i}{z - z_i} \right)^n \tag{2.50}$$

The above equation, for any large value of n, is only stable if,

$$\left| \sum_{i=1}^{M} \frac{k_i}{z - z_i} \right| \leq 1 \tag{2.51}$$

which means the stability region of NILT is given by $[N/M]$ Padé approximation of e^z, i.e.,

$$|\xi_{N,M}(z)| \leq 1 \tag{2.52}$$

Therefore, NILT is A-stable if $N > M$ and $\Re(\lambda) < 1$. For example, if $N = 1$ and $M = 0$ the method is of order 1 and the stability region is outside of the unit circle at the right side of the imaginary axis, same as the BE. Moreover, the NILT method is L-stable if and only if $N = M - 2$ [18, Ch. 10].

2.2.3.3 Numerical Advantages of NILT

The aforementioned characteristics give the NILT method notable advantage compared to traditional time-domain integration methods used in circuit simulations. The ability to employ high-order polynomial approximations while maintaining L-stability during the time-stepping process. Additionally, the NILT method is unique in that it is a time-domain technique fundamentally based on frequency-domain principles. This dual nature allows NILT to efficiently handle circuits that are more straightforward to represent in the frequency domain but more challenging to model in the time domain.

2.2.3.4 The Modified NILT

In a separate, recent research effort, the foundational principles of NILT were adapted to create a more precise paradigm for linear circuit simulation [34]. This new method,

called NILTn, utilizes the n^{th}-order derivatives of circuit variables with respect to the Laplace domain variable s. In contrast, traditional NILT often referred to as NILT0 and relies on the 0^{th}-order derivatives (i.e., the simple values) of these variables in the Laplace domain. The increased accuracy of NILTn over the conventional NILT has resulted in greater efficiency for simulating purely linear circuits. This section briefly outlines the NILTn approach.

Assuming, it is required to approximate a vector of waveforms, $\mathbf{y}(t)$, given its vector of Laplace domain waveforms $\mathbf{Y}(s)$. That process is fully detailed in [34]. The NILTn paradigm can be illustrated by using a vector of waveforms, $\mathbf{w}(t)$, defined through,

$$\mathbf{w}(t) \coloneqq \mathscr{L}^{-1}\left\{(n+1)\mathbf{Y}((n+1)s)\right\}(t) \tag{2.53}$$

where $n > 0$ is a positive integer. The properties of the Laplace transformation thus enable writing $\mathbf{w}(t) = \mathbf{y}(t/(n+1))$, or alternatively,

$$\mathbf{y}(t) = \mathbf{w}((n+1)t) \tag{2.54}$$

The relation (2.54) highlights the fact that evaluating the inverse of the Laplace transform in (2.53) at the (scaled) time of $(n+1)t$ yields the value of $\mathbf{y}(t)$ at the normal time scale, t. With this notion in mind one can write

$$\begin{aligned} \mathbf{y}(t) &= \mathscr{L}^{-1}\left\{\mathbf{W}(s)\right\}((n+1)t) \\ &= \frac{n+1}{2\pi\jmath}\int_{c-\jmath\infty}^{c+\jmath\infty}\mathbf{Y}((n+1)s)\mathrm{e}^{(n+1)st}\mathrm{d}s \end{aligned} \tag{2.55}$$

The NILTn approach starts from (2.55) and proceeds by substituting z for st, using the partial fraction expression to expand $(\xi_{N,M}(z))^{n+1}$, multinomial expansion theorem

as another partial fraction summation involving the same poles z_i but with higher multiplicity. Finally, applying the Cauchy theorem of residues transforms the integral in (2.55) into a finite linear combination of $\mathbf{Y}(s)$. This process, which is explained in detail in Chapter 6, yields the approximation sought for $\mathbf{y}(t)$ denoted by $\tilde{\mathbf{y}}^{[n]}(t)$. As an example, the special case of $n = 1$ yields the following expression for the NILT1,

$$\tilde{\mathbf{y}}^{[1]}(t) = \frac{-8}{t} \sum_{i=1}^{M/2} \Re\left(\frac{k_i^2}{t} \frac{\mathrm{d}\mathbf{Y}(s)}{\mathrm{d}s} + k_i \mathbf{Y}(s) \sum_{\substack{j=1 \\ j \neq i}}^{M} \frac{k_j}{z_i - z_j} \right) \Bigg|_{s=\frac{2z_i}{t}} \tag{2.56}$$

The key aspect that is worthy of note in comparing the NILTn expression of $\tilde{\mathbf{y}}^{[n]}(t)$ for $n = 1$ in (2.56) to $\tilde{\mathbf{x}}(t)$ in (2.41), of the conventional NILT, is the presence of higher-order derivatives w.r.t. s in the former. This leads to higher accuracy and improving computational costs of circuit simulation.

2.3 Discussion

This chapter presented an overview of the various classes of the methods that appeared in the literature to solve a system of differential equations numerically. The first class of methods was LMS. LMS methods rely on information from multiple previous time steps (or points) to compute the solution at the next step. They "look back" at past values of the solution and its derivatives to predict the future. The second class of methods covered in this chapter is the single-step multi-stage methods which are single-step methods that compute the solution at the next time step using only the current point, but they evaluate the derivative multiple times within that step. These intermediate evaluations (called "stages") improve accuracy. The third class of methods is the

single-step method of Obreshkov which uses higher-order derivatives (higher than 1) to numerically solve the differential equation. The Obreshkov method is based on the Hermite interpolation.

The ideal method to use in solving differential equations numerically, is to have a high-order, where the coefficients of operator (2.20) are desired to vanish at high q. The advantages of the high-order in the numerical method is that allows extending the length of the time step h, and in doing so, reduces the cost of repeating the computations at too many time points that do not change significant variations. However, it is also expected that there will be a computational overhead associated with the high-order method, which could offset the gains or savings attained from the high-order method.

The issue of stability is yet another factor that cannot be ignored in the method used. As mentioned earlier, error arising in taking a single step in time by any method is inevitable. Stability of the method is determined by whether this error is amplified or damped in subsequent steps. A high-order method lacking stability can thus become quickly useless as the error from past steps rises sometimes quickly or exponentially.

The quest of the high-order stable methods in the class of LMS was abandoned with the development known in the literature as the Dahlquist barriers. Named after Germund Dahlquist, these barriers highlight trade-offs between accuracy and stability in the LMS methods. The first Dahlquist barrier does not really impact the endeavor of circuit simulation since it bears on the notion of zero stability. However, it is the second barrier that was truly consequential for transient time-domain circuit simulations, as it is pertinent to the notion of A-stability, which is a stronger but desirable condition for stiff differential equations an instance of which is the MNA equations. The Dahlquist second barrier states that [16]:

- The order of an A-stable LMS cannot exceed 2,
- Among order-2 A-stable methods, the trapezoidal rule has the smallest error constant.

The barrier stems from the interplay between the method's characteristic polynomial representation and its stability region. Higher-order methods adjust coefficients, potentially destabilizing the characteristic polynomial's roots. Dahlquist proved these are inherent limits. This is major limitation for the LMS methods in handling stiff problems because it limits how high-order you can go while keeping unconditional stability. Methods like BDF can go higher (up to order 6), but they sacrifice A-stability for weaker stability properties like $A(\alpha)$-stability.

The other class of methods that are not bounded or limited by the Dahlquist Barrier is the RK methods. RK methods can be explicit or implicit. Explicit RK methods are conditionally stable and therefore do not offer the A-stability that is desired in transient circuit simulations. Implicit RK with s stages are often A-stable and can also be L-stable, with order $2s$. RK methods depend on quadrature techniques to find nodes of quadrature c_i and the weights b_i in the Butcher tableau. It then uses the set of equations that define the order conditions to solve for the entries a_i of that tableau. The definition of the method is itself a problem to be reckoned with, but it needs to be done once.

The other type of single step methods, which are also free from the second Dahlquist barrier, is the one using the Obreshkov formula. This formula is based on Hermite interpolation. It has been proved that the method can be made A- and L-stable for any arbitrary order. In addition, its coefficients are given by closed-form where there is no need to solve a problem to compute them. The only disadvantage in

the Obreshkov formula is that it requires utilizing high-order derivatives, which is not always a straightforward task in the domain of circuit simulation due to the complexity of the models of nonlinear devices.

This chapter presented an overview of the numerical inversion of the Laplace transform. The two key characteristics that make the NILT method stands out among the other time-domain methods. First, it is free from the conflict of the order and stability that constrains the LMS methods, allowing it to be used for arbitrary order with A- and L-stability. The coefficients of the NILT method are obtained as problem-independent, from the rational Padé approximation of the exponential function. Thus it shares with the Obreshkov-based method its most important advantages. However, the second key characteristic in NILT is that it requires computing is the response of the differential equations in the complex frequency (Laplace) domain. This characteristic has a unique implication on the modeling of the system that the differential equations represents: it is no longer required to have the underlying system have a time-domain model, although the method will also work equally well with systems that have a direct representation in the time-domain. Furthermore, recent advancements in complex full-wave circuit modeling using PEEC [26, 27] have highlighted its efficiency and renewed interest in using it as a general circuit simulation engine. For PEEC circuits, NILT has proven to be more reliable than traditional time-stepping methods [28].

As discussed in Section 2.2.3, NILT is based on the Laplace domain representation of the original circuit. This means NILT is applicable only to circuits modeled by linear differential equations. While this includes a significant portion of circuits, it excludes those with elements modeled by nonlinear functions. Specifically, the MNA formulation of general circuits is represented by (2.2), where $\mathbf{f}(\mathbf{x}(t))$ describes any algebraic (memoryless) nonlinearity. Memory (capacitive/inductive) nonlinearities can

be similarly accommodated using a charge/flux-oriented MNA formulation, which adds the charge/flux in nonlinear capacitors/inductors to the list of MNA variables $\mathbf{x}(t)$ [15]. Therefore, without any loss of generality, we treat it in this work the algebraic nonlinearity.

It can be seen from (2.2) that in the presence of nonlinearity in the circuit, the NILT framework cannot be applied in this situation since the nonlinear term precludes a Laplace domain representation of the MNA formulation. This provides an opportunity to adapt NILT for time-domain analysis of general circuits that include nonlinear elements. The main obstacle, however, lies in extending NILT's applicability to nonlinear circuits. The intricacies of addressing this challenge form the core of the novel approach presented in subsequent chapters.

Chapter 3

Developing NILT for Nonlinear Circuit Simulation

3.1 Introduction

The previous chapter provided a concise overview of the primary time-domain methods commonly employed in circuit simulation for analyzing circuits. Among these, the time-domain approach based on the NILT received particular emphasis due to its distinct advantages. Despite these merits, NILT techniques remain fundamentally restricted in their application just for linear circuits. This limitation stems from the intrinsic reliance of NILT on the Laplace-domain formulation of circuits, which is inherently suited to linear systems.

The central objective of this thesis is to overcome this limitation and extend the applicability of NILT to nonlinear circuit analysis.

This chapter marks the initial step in pursuing this objective. It outlines the concep-

tual framework for developing an enhanced method capable of addressing the challenges posed by nonlinear circuit analysis. The discussion will set the stage for subsequent chapters by providing the roadmap, so to speak, for this development.

To facilitate the description of the proposed approach, we first consider its application to a relatively simple circuit which is depicted by Figure 3.1. Key ideas and notations will be established through this example, paving the way to present the treatment of general circuits.

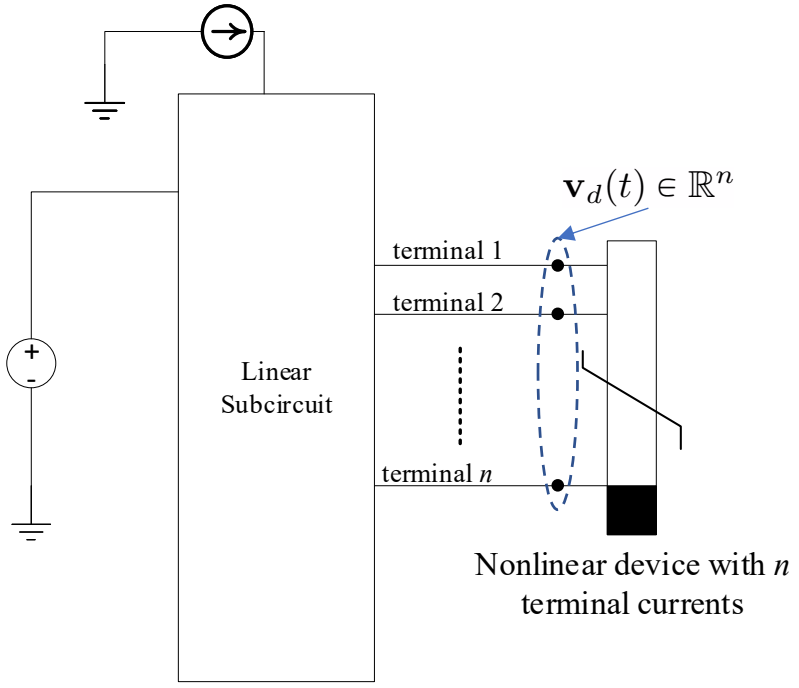


Figure 3.1: An example circuit used to illustrate the proposed approach.

The schematic in Figure 3.1 shows a circuit that consists of a large general linear circuit driven by multiple independent sources and is attached to a single nonlinear device. The representation of the nonlinear device in this circuit serves as vehicle to illustrate the handling of an arbitrary nonlinear element. It does so by assuming that

the device is controlled by n voltages at its terminals, and that the currents in those terminals are functions of those voltages captured by a mapping $\mathbf{\Phi}_R \colon \mathbb{R}^n \to \mathbb{R}^n$. The voltages at the device terminals, denoted $\mathbf{v}_d(t) \in \mathbb{R}^n$ constitute a subset of the MNA variables $\mathbf{x}(t)$ and are expressed in terms of $\mathbf{x}(t)$ through the use of an incidence matrix $\mathbf{d} \in \{0,1\}^{K \times n}$ that selects the specific n nodes representing the terminals of the device, i.e.,

$$\mathbf{v}_d(t) = \mathbf{d}^\top \mathbf{x}(t) \tag{3.1}$$

3.2 Circuit Splitting

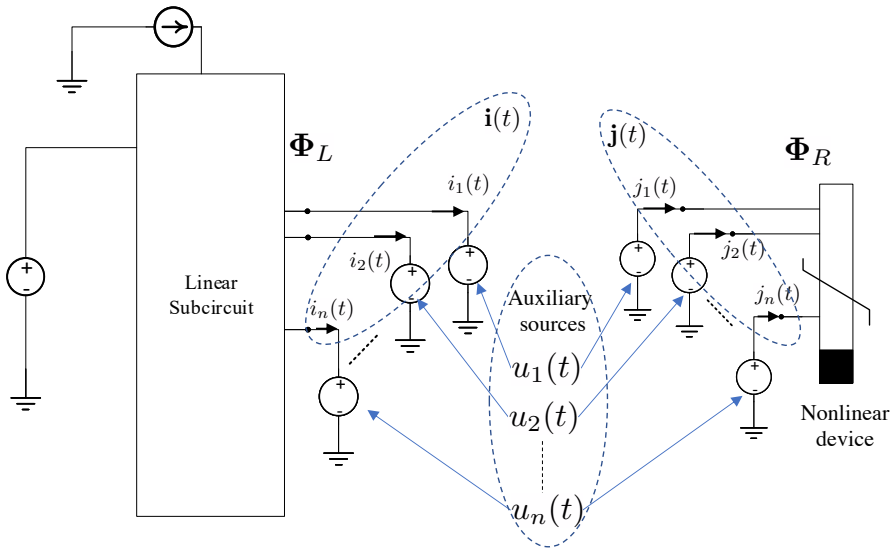


Figure 3.2: Splitting the circuit of Figure 3.1 and inserting the auxiliary sources.

The proposed approach starts by splitting the circuit in Figure 3.1 into two circuits along the n terminals and inserting a set of $2n$ voltage sources (henceforth referred to as *auxiliary sources*) to obtain the two circuits shown in Figure 3.2. The two sets

of auxiliary sources are assumed to be identical, that is, the n sources on the left- and those n sources on right-side of the split are assumed to have the same temporal waveforms which are denoted by $u_r(t), r = 1 \cdots, n$. For arbitrary $u_r(t)$, the currents in those (auxiliary) sources are, in general, not equal. Thus, the currents in the left-side of the split are denoted by $i_r(t)$ while those on the right-side are denoted by $j_r(t)$, with $r = 1, \cdots, n$. Grouping the set of voltages $u_r(t)$ and currents $j_r(t)$ and $i_r(t)$ in vectors denoted by $\mathbf{u}_{\text{aux}}(t)$, $\mathbf{j}(t)$ and $\mathbf{i}(t) \in \mathbb{R}^n$, i.e.,

$$\mathbf{u}_{\text{aux}}(t) = [u_1(t), u_2(t), \cdots, u_n(t)]^\top \tag{3.2}$$

$$\mathbf{i}(t) = [i_1(t), i_2(t), \cdots, i_n(t)]^\top \tag{3.3}$$

$$\mathbf{j}(t) = [j_1(t), j_2(t), \cdots, j_n(t)]^\top \tag{3.4}$$

we may write

$$\mathbf{i}(t) = \mathbf{\Phi}_L\left(\mathbf{u}_{\text{aux}}(t)\right), \tag{3.5}$$

$$\mathbf{j}(t) = \mathbf{\Phi}_R\left(\mathbf{u}_{\text{aux}}(t)\right) \tag{3.6}$$

where $\mathbf{\Phi}_R$ is a nonlinear mapping $\mathbb{R}^n \to \mathbb{R}^n$, and $\mathbf{\Phi}_L$ is a linear operator, as a result of the linearity of the circuit on the left-side.

The objective of the proposed approach is to carry out the simulation of the entire circuit (without splitting the nonlinear device) in the time-domain using the NILT approach. Given the difficulty of incorporating the nonlinear device in a Laplace-domain-based framework, the idea of splitting circuit as described above will be the principle tool to achieve this objective, as shown next.

The key idea in the proposed approach seeks to compute the auxiliary sources

$\mathbf{u}_{\text{aux}}(t)$ that enforce the equality $\mathbf{i}(t) = \mathbf{j}(t)$. Naturally, in the context of computer simulation, this condition can be enforced at discrete time points, $t_l : l = 1, 2, \cdots$, with variable step size $h_l = t_l - t_{l-1}$. To simplify the notation, we shall assume a constant step size, i.e., that $t_l = lh$, h being an appropriate time step. Nonetheless, enforcing a simple equality at discrete time points, i.e., $\mathbf{i}(lh) = \mathbf{j}(lh)$ *is not* sufficient to bring over the numerical advantages of NILT approach to the problem at hand. Instead of enforcing simple equality in the currents, the proposed approach does enforce the equality in the high-order derivatives (up to a given order $p - 1$) of the currents at those discrete time points. In other words, the proposed approach casts the problem as that of computing the auxiliary sources $\mathbf{u}_{\text{aux}}(t)$ such that

$$\hat{\mathbf{i}}_l^{(m)} = \hat{\mathbf{j}}_l^{(m)}; \quad \begin{cases} l = 0, 1, 2, 3 \cdots \\ m = 0, 1, 2, \cdots, p-1 \end{cases} \tag{3.7}$$

where[1]

$$\hat{\mathbf{i}}_l^{(m)} := h^m \left. \frac{d^m \mathbf{i}(t)}{dt^m} \right|_{t=lh}, \tag{3.8}$$

$$\hat{\mathbf{j}}_l^{(m)} := h^m \left. \frac{d^m \mathbf{j}(t)}{dt^m} \right|_{t=lh} \tag{3.9}$$

As will be shown in Chapter 5, enforcing a simple equality of the currents at discrete points, i.e., by setting $p = 1$, in (3.7), will *effectively* reduce m to 1, even though the actual implementation may have been carried out with a value $m > 1$.

It should also be mentioned here that the condition enforced by (3.7) implies that the interface voltage waveforms need to satisfy a certain smoothness requirement, with

[1]Scaling the derivatives by powers of h in (3.8) is meant to improve the numerical conditioning of the resulting equations and prevent the exponential growth in the numerical analysis.

derivatives up to order p being defined and computable. This requirement was found to be satisfied for all circuits used in the experimental testing in this thesis.

The system in (3.7) represents a system of $n \times p$ nonlinear equations whose solution, using the Newton-Raphson (NR) method at $l = 0, 1, 2, 3, \cdots$ is used in the proposed approach to steer, so to speak, the simulation of the entire (non-split) circuit in the time domain. The computation of the currents in the linear sub-network and their derivatives $\hat{\mathbf{i}}_l^{(m)}$ will be carried out using the NILT approach, while the currents in the nonlinear elements and their derivatives $\hat{\mathbf{j}}_l^{(m)}$ will be addressed through the idea of RT method [45] which is explained in detail in Sub-section 3.5.2.

As quick illustration of the splitting technique, we use the simple diode circuit on the left in Figure 3.3 to show how the splitting is carried out producing two separate circuits, on the right, coupled through a single auxiliary source.

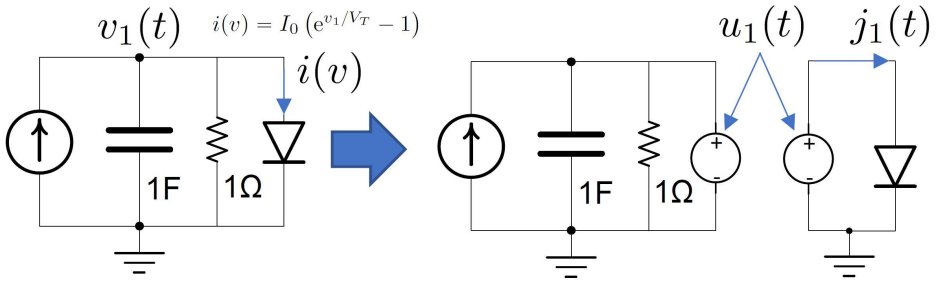


Figure 3.3: A simple circuit and its split parts.

Before delving into the details of solving (3.7) using NR, a brief note on the relation between the proposed technique and earlier approaches related to ideas based on circuit splitting is in order.

3.3 Relation to Other Circuit Splitting Approaches

The idea of partitioning a circuit into smaller subcircuits is well-known in circuit [51–53] and electromagnetic simulations [54]. It has been used under different names such as domain decomposition, waveform relaxation [55, 56], tearing and diakoptic analysis. Those methods are motivated by the fact that in many situations the simulations of the subcircuits individually is easier than the simulation of the whole, or can be run in parallel to achieve speedups. This idea was implemented in several approaches, such as branch tearing, node tearing and hierarchical decomposition [57–61]. Nonetheless, and in spite of the wide variety of those techniques, they all shared a common approach to simulate the response of the whole circuit based on the simulation of the subcircuits, namely, through enforcing a simple (zero-order) equality on the boundary variables.

Seen from that perspective, those approaches can be considered as a special case of the proposed approach, that is, when implemented with $p = 1$ in (3.7). Indeed, as will be shown in Chapter 5, the ability to explicitly compute and enforce the equality of higher-order derivatives for the port voltages represents the key feature that enables the proposed approach to inherit the main NILT characteristics (high order and numerical stability).

3.4 Temporal Form of the Auxiliary Sources

This section presents the general template used to represent the auxiliary voltage sources in the time-domain. The used template of these sources will be in the form of

piece-wise polynomial of the following form

$$\mathbf{u}_{\text{aux}}(t) = \sum_{l=0,1,2,\cdots} \mathbf{u}_l(t), \qquad (l-1)h \leq t \leq lh, \tag{3.10}$$

with each piece being a polynomial in t with degree expressed in terms of two integers p and q,

$$\mathbf{u}_l(t) = \sum_{i=0}^{p+q+1} \bar{\bar{\mathbf{u}}}_{l,i}\,(t-lh)^i \tag{3.11}$$

where $\bar{\bar{\mathbf{u}}}_{l,i} \in \mathbb{R}^n$ are the coefficients of the polynomial. The reason for expressing the degree of the polynomial by two separate integers will be made clear in the course of the presentation of the new approach.

The choice of the polynomial form of the auxiliary sources points to the fact that it will be the coefficients of the polynomials, $\bar{\bar{\mathbf{u}}}_{l,i}$, that will play the key role in enforcing the equality expressed in (3.7). For convenience, the problem expressed in (3.7) will be modified to read,

$$\begin{aligned} &\text{Compute:} \qquad \bar{\bar{\mathbf{u}}}_{l,i}\,; \quad i = 0, 1, \cdots, p+q+1 \\ &\text{such that :} \\ &\hat{\mathbf{i}}_l^{(m)} = \hat{\mathbf{j}}_l^{(m)}; \quad \begin{cases} l = 0, 1, 2, 3 \cdots \\ m = 0, 1, 2, \cdots, p-1 \end{cases} \end{aligned} \tag{3.12}$$

More details on the coefficients $\bar{\bar{\mathbf{u}}}_{l,i}$ will be provided in Chapter 4.

3.5 Computing $\hat{\mathbf{i}}_l^{(m)}$ and $\hat{\mathbf{j}}_l^{(m)}$

It is obvious that solving the problem defined in (3.12) requires, in the first place, a method to compute $\hat{\mathbf{i}}_l^{(m)}$ and $\hat{\mathbf{j}}_l^{(m)}$. In general, that method has to start with the polynomial $\mathbf{u}_l(t)$ or basically its coefficients, $\bar{\bar{\mathbf{u}}}_{l,i}$. This process is illustrated next.

3.5.1 Computing $\hat{\mathbf{i}}_l^{(m)}$ using NILT

The currents $\mathbf{i}(t)$ arise in response to the stimulus of the external sources as well as the added auxiliary sources in a linear circuit. Hence, one can utilize the concept of NILT along with the principle of superposition to compute $\hat{\mathbf{i}}_l^{(m)}$, by first applying NILT to compute the portion of $\hat{\mathbf{i}}_l^{(m)}$ arising from the external sources, and then applying NILT to compute the other portion in $\hat{\mathbf{i}}_l^{(m)}$ arising from the auxiliary sources, and then adding the two portions to arrive at the required $\hat{\mathbf{i}}_l^{(m)}$. For the sake of brevity and clarity, we demonstrate in this section computing the response arising only from auxiliary sources, as it will be clearly seen that the approach used with auxiliary sources can be utilized with the external sources in an almost identical manner.

Inserting the auxiliary voltage sources at the ports of the circuit on the left side of the divide in Figure 3.2, then shifting the time origin to lh via the change of time variable $t \rightarrow h\hat{t}+h(l-1)$ and taking the Laplace transform with respect to $\hat{t}$, the circuit on the left side can then be represented in the Laplace-domain using an augmented

MNA formulation

$$\left(\hat{s}\overbrace{\begin{bmatrix}\frac{1}{h}\mathbf{C} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}}^{\bar{\mathbf{C}}}+\overbrace{\begin{bmatrix}\mathbf{G} & \mathbf{d}\\ \mathbf{d}^{\top} & \mathbf{0}\end{bmatrix}}^{\bar{\mathbf{G}}}\right)\overbrace{\begin{bmatrix}\mathbf{X}(\hat{s})\\ \mathbf{I}_{\text{aux}}(\hat{s})\end{bmatrix}}^{\bar{\mathbf{X}}(\hat{s})}=\underbrace{\begin{bmatrix}\mathbf{0}\\ \mathbf{U}_l(\hat{s})\end{bmatrix}}_{\bar{\mathbf{B}}(\hat{s})}+\bar{\mathbf{C}}\bar{\mathbf{x}}(0) \tag{3.13}$$

where $\mathbf{C}$ and $\mathbf{G}$ are the same matrices representing the linear part of the original circuit (before the splitting action), $\bar{\mathbf{C}}$ and $\bar{\mathbf{G}}$ being their augmented matrices after enacting the splitting process, $\bar{\mathbf{B}}(\hat{s})$ is a zero vector augmented by the Laplace transform of the auxiliary voltage sources only, $\mathbf{I}_{\text{aux}}(\hat{s}) \in \mathbb{C}^n$ are the Laplace-domain currents in the auxiliary sources, $\mathbf{I}_{\text{aux}}(\hat{s}) = \mathscr{L}\{\mathbf{i}(\hat{t})\}$, and $\mathbf{U}_l(\hat{s}) = \widehat{\mathscr{L}}\{\mathbf{u}_l(\hat{t})\}$ and $\bar{\mathbf{X}}(\hat{s}) = \widehat{\mathscr{L}}\{\bar{\mathbf{x}}(\hat{t})\}$, $\bar{\mathbf{x}}(t)$ being the variables of the circuits after augmenting them with the currents in the auxiliary sources.

The NILT computational process can next be invoked to obtain $\hat{\mathbf{i}}_l^{(m)}$. We first consider the case where $m = 0$.

3.5.1.1 Computing $\hat{\mathbf{i}}_l^{(m)}$ for $m = 0$.

Note that by definition

$$\begin{aligned}\hat{\mathbf{i}}_l^{(0)} &= \widehat{\mathscr{L}}^{-1}_{\text{NILT}}\left\{\mathbf{I}_{\text{aux}}(\hat{s})\right\}(\hat{t})\Big|_{\hat{t}=1}\\ &= -\sum_{i=1}^{M/2} 2\Re\left(k_i\hat{\mathbf{I}}_{\text{aux}}(z_i)\right)\end{aligned} \tag{3.14}$$

where

$$\hat{\mathbf{I}}_{\text{aux}}(z_i) = \underbrace{[\mathbf{0} \quad \mathbb{I}_n]}_{\bar{\mathbf{d}}} \left(z_i \bar{\mathbf{C}} + \bar{\mathbf{G}}\right)^{-1} \begin{bmatrix} \frac{1}{h}\bar{\mathbf{C}}\tilde{\mathbf{x}}_{l-1} \\ \mathbf{U}_l\left(z_i\right) \end{bmatrix} \tag{3.15}$$

and $\bar{\mathbf{d}}$ is a selector matrix containing only zeros and $\mathbb{I}_n$ is an identity matrix with size n.

3.5.1.2 Computing $\hat{\mathbf{i}}_l^{(m)}$ for $m > 0$

This case is handled through the following identity of the Laplace transform

$$\frac{\mathrm{d}^m \mathbf{i}(\hat{t})}{\mathrm{d}\hat{t}^m} = \mathscr{L}^{-1}\left\{ \hat{s}^m \mathbf{I}_{\text{aux}}(\hat{s}) - \sum_{j=1}^{m} \left.\frac{\mathrm{d}^{j-1}\mathbf{i}(\hat{t})}{\mathrm{d}\hat{t}^{j-1}}\right|_{\hat{t}=0} \hat{s}^{m-j} \right\}(\hat{t}) \tag{3.16}$$

Through the change of variables $t \leftarrow h\hat{t} + (l-1)h$, we have

$$\left.\frac{\mathrm{d}^m \hat{\mathbf{i}}(\hat{t}\,)}{\mathrm{d}\hat{t}^m}\right|_{\hat{t}=1} = h^m \left.\frac{\mathrm{d}^m \mathbf{i}(t)}{\mathrm{d}t^m}\right|_{t=lh} \tag{3.17}$$

noting the definition of $\hat{\mathbf{i}}_l^{(m)}$ in (3.8) and invoking the NILT process to express the right side of (3.16), then $\hat{\mathbf{i}}_l^{(m)}$ can be expanded as shown in the following steps,

$$\begin{aligned}
\hat{\mathbf{i}}_l^{(m)} &= \widehat{\mathscr{L}}^{-1}_{\text{NILT}} \left\{ \hat{s}^m \hat{\mathbf{I}}_{\text{aux}}(\hat{s}) - \sum_{j=1}^{m} \left.\frac{\mathrm{d}^{j-1}\hat{\mathbf{i}}(\hat{t}\,)}{\mathrm{d}\hat{t}^{j-1}}\right|_{\hat{t}=0} \hat{s}^{m-j} \right\}(\hat{t}\,)\Bigg|_{\hat{t}=1} \\
&= -\sum_{i=1}^{M/2} 2\Re\left(k_i z_i^m \hat{\mathbf{I}}_{\text{aux}}\left(z_i\right)\right) \sum_{i=1}^{M/2}\sum_{j=1}^{m-1} 2\Re\left(k_i z_i^{m-j}\, \hat{\mathbf{i}}_{l-1}^{(j-1)}\right) + \hat{\mathbf{i}}_l^{(m)} \overbrace{\sum_{i=1}^{M/2} \Re(k_i)}^{0} \\
&= -\sum_{i=1}^{M/2} 2\Re\left(k_i z_i^m \hat{\mathbf{I}}_{\text{aux}}\left(z_i\right) - k_i \sum_{j=1}^{m-1} z_i^{m-j}\, \hat{\mathbf{i}}_{l-1}^{(j-1)}\right)
\end{aligned} \tag{3.18}$$

where the property of NILT that $\sum_{i=1}^{M/2} k_i = 0$ was used in the second step [18, Ch. 10].

The derivation in (3.18) demonstrates that computing $\hat{\mathbf{i}}_l^{(m)}$, the m^{th}-order derivatives of currents in the linear partition at time $t = lh$, requires $M/2$ frequency-domain computation of $\hat{\mathbf{I}}_{\text{aux}}(\,\hat{s}\,)$ at $s = z_i$, $i = 1, \cdots, M/2$. In addition, this computation requires the values of $\hat{\mathbf{i}}_{l-1}^{(m)}$, which are the m^{th}-order derivatives of currents in the linear partition at time $t = (l-1)h$.

3.5.2 Computing $\hat{\mathbf{j}}_l^{(m)}$

Deriving the computational process used to obtain $\hat{\mathbf{j}}_l^{(m)}$, $m = 0, 1, 2 \cdots, p-1$ from $\hat{\mathbf{v}}_l^{(n)}$, $n = 0, 1, 2 \cdots, p$ is done by first recasting $\mathbf{u}_l(\,t\,)$ of (3.11) in the form

$$\mathbf{u}_l(\hat{t}\,) = \sum_{r=0}^{p+q+1} \hat{\mathbf{u}}_{l,r} \left(\hat{t}\, - 1\right)^r \tag{3.19}$$

using the change of variables $t \leftarrow h\hat{t} + (l-1)h$, where $\hat{\mathbf{u}}_{l,r} = h^r \bar{\bar{\boldsymbol{u}}}_{l,r}$.

Computation of $\hat{\mathbf{j}}_l^{(m)}$ from $\hat{\mathbf{u}}_{l,r}$ is performed using the notion of RTs, which is detailed next using an illustrative example.

3.5.2.1 Operation of Rooted Trees

The purpose of employing the concept of RT in computing the high-order derivatives of the nonlinear part is to make the proposed approach applicable with arbitrarily complex, user-defined form of non-linearity. This section illustrates the process of computing $\hat{\mathbf{j}}_l^{(m)}$ though the example shown in Figure 3.3 where the current in the diode

is assumed to be given by $I_0\left(\mathrm{e}^{v_1(t)/V_T}-1\right)$. This process takes the $\hat{\mathbf{u}}_{l,r}$ as the input and produces $\hat{\mathbf{j}}_l^{(m)}$ as the output.

The splitting approach, illustrated on Figure 3.3, uses only one auxiliary voltage source ($n=1$), whose current is

$$j_1(t)=\phi_1(v(t))\equiv I_0\left(\mathrm{e}^{v(t)/V_T}-1\right).$$

Therefore, the objective of computing $\hat{\mathbf{j}}_l^{(m)}$, for this example, becomes the objective of computing $h^m\frac{\mathrm{d}^m j_1(t)}{\mathrm{d}t^m}$ or, alternatively, $h^m\frac{\mathrm{d}^m\phi_1(v(t))}{\mathrm{d}t^m}$. Computing the derivatives of ϕ_1, is done by first representing it in the form of a RT. The RT representing the nonlinear expression of $\phi_1(v(t))$ is shown in Figure 3.4.

The nodes in a general RT fall into one of three categories:

(a) Root node. This is a unique node that sits at the bottom of the tree (parent-less). The tree contains only one such node representing the nonlinear expression, or ϕ_1 in the example taken,

(b) Leaf nodes existing at the highest level of the tree (henceforth termed *childless* nodes). Leaf nodes are depicted on the tree using triangles, and represent two types of terms: constant terms in the nonlinearity, e.g., I_0 (node 2), which are independent of time t, and the controlling voltages, e.g., node 7 for $v(t)$ in the example used. For clarity, the former is assigned letter '`C`' and the latter is assigned the letter '`W`' on Figure 3.4, and

(c) Functional nodes. Those nodes (designated using circles on the tree) represent the different operations involved in the nonlinear expression. Functional nodes have only one parent node but could have more than one child node and represent

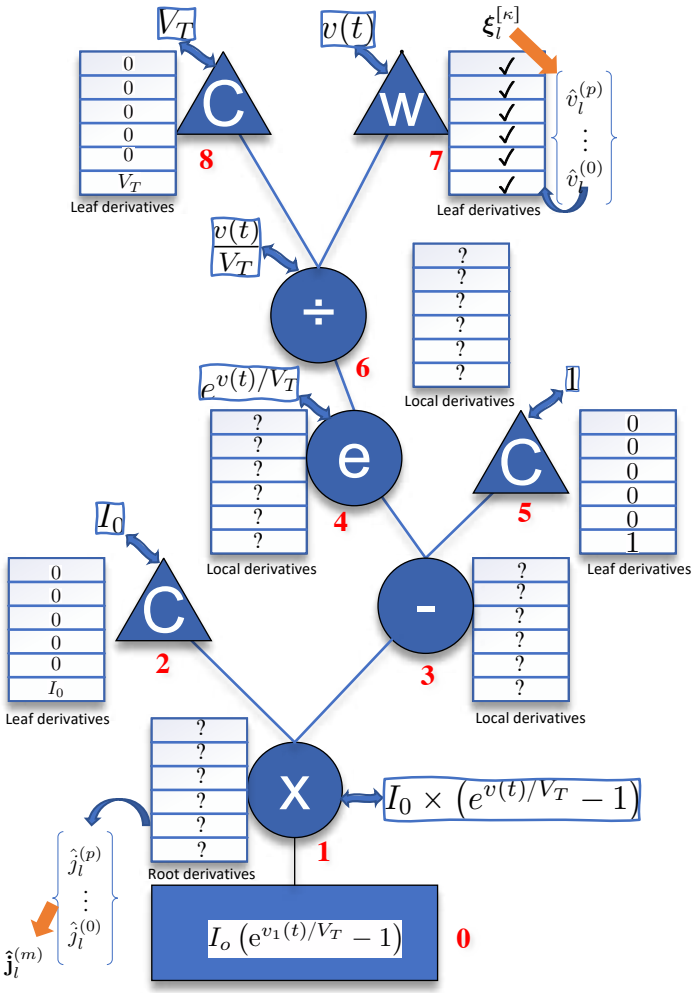


Figure 3.4: Representing $I_0 \left(\mathrm{e}^{v(t)/V_T} - 1\right)$ as a RT.

either atomic functions, such the exponential or the logarithm function, see node 4, or arithmetic operators such as multiplication and division, such as nodes 1,3 and 6, in Figure 3.4.

Each node in the tree maintains a local storage that will receive p values of the derivatives, at that node level. For example, consider node 4 in Figure 3.4 which represents the exponent $e^{v(t)/V_T}$: its local storage (depicted as an array on the figure) will hold the derivatives of that term $e^{v(t)/V_T}$, w.r.t. t. At the start of the process, leaf nodes will have those derivatives known *apriori*. Take for example the nodes representing the constant terms in the tree. The derivatives w.r.t. t of those terms are trivial, by virtue of being constants. For example node 2, representing the constant I_0, have all zero high-order derivatives. This fact is reflected on array attached to node 2 in Figure 3.4.

Other leaf nodes will have their derivatives taken directly from $\hat{\mathbf{u}}_{l,r}$. This is mainly due to the fact that $\hat{\mathbf{u}}_{l,r}$ represent the derivatives of the voltages applied at the terminals of the nonlinear devices (the auxiliary sources)[2]. Indeed this is the case with node 7 which will have its local storage filled with the derivatives of the voltage $v_1(t)$ that are accessible from the $\hat{\mathbf{u}}_{l,r}$. More generally, all leaf nodes will have the derivatives (of the terms they represent) readily stored in their local storage at beginning of the computation in a given NR iteration. At this stage, however, derivatives of the terms represented by the other non-leaf nodes are still unknowns (marked by '?') that need to be computed. In particular, the derivatives of the root node, which is the end goal of the entire process, still need to be computed.

[2]This representation can be seen from the relation between derivative of the polynomial and its coefficients, where the m^{th}-order derivative of a polynomial is the value of its m^{th} coefficient multiplied by $m!$

The fundamental operating principle in the RT is that derivatives at any non-leaf node can be computed from the derivatives of the children (higher level) nodes. The formulae that are used for this purpose vary according to the function or the arithmetic operator that the node represents in the RT. Those formulae have been developed for primitive functions, see [8], and pre-programmed in each node.

Thus, the derivatives on the root node 0 are obtained from the derivative of its children, or node 1, which represents the multiplication operator of the constant I_0 and the term $\mathrm{e}^{v(t)/V_T} - 1$. The first phase of computation, therefore, starts at the root node, by invoking a command, or a function, that queries the node about its zero-order derivative, or $\hat{j}_l^{(0)}$. The root node, not having computed this derivative yet, passes this command onto the next level. The command thus propagates upward until it reaches the uppermost level of the RT where the leaf nodes exist. The leaf nodes, having their derivatives preassigned, respond by passing the required derivative to the immediate lower level (its parent node) in the tree. Upon receiving the derivatives from all the children nodes, the parent nodes use those derivatives in the pre-programmed formula to compute its own derivative, i.e., the derivative demanded by their lower level ancestral node. The computed derivative is stored in the local storage next to the node for later use, and passed down one level in the tree. This is the second phase of computation, which propagates downward until it reaches the root node providing the required derivative needed by the algorithm, or $\hat{j}_l^{(m)}$. This two-phase process is launched for every derivative needed, $\hat{j}_l^{(m)}$, $m = 0, 1, \cdots, p$.

Construction of the RTs has been fully automated in the implementation of the proposed approach, requiring only the description of the nonlinear devices in the circuit netlist. This description is provided as MATLAB-like script that specifies the nonlinear behavior through a sequence of assignment and conditional statements, thereby

allowing representation of fairly complex device behavior. A compiler, designed using ANother Tool for Language Recognition (ANTLR) [62], compiles the MATLAB script, converting it into a rooted tree structure. The RT is built in an Object-Oriented Programming (OOP) using the C++ language, which represent nodes as objects of classes with specified inheritance architecture. The OOP allows encapsulating in each node the required pre-programmed formula, as a virtual function resolved at run-time according to the type of expression it represents in the tree, along with the data that are used to in storing the derivatives. The above description thus sums up the process of computing $\hat{\mathbf{j}}_l^{(m)}$ given $\hat{\mathbf{v}}_l^{(m)}$.

3.6 Discussions

This chapter outlined the skeletal framework for the simulation process, introducing a novel method for solving the nonlinear differential equations arising from the circuit MNA formulation. While this description provides the foundation, key implementation details are still absent.

One of the important details that will be provided in Chapter 4 is the expression of the polynomial coefficients $\bar{\bar{\mathbf{u}}}_{l,i}$ in terms of the circuit MNA variables.

These missing elements will be addressed in the subsequent chapters, where they will be introduced and elaborated upon as needed.

Chapter 4

Stabilization and Implementation of NILT-Nonlinear Simulation

Chapter 3 introduced the basic methodology for employing a NILT-based framework to simulate nonlinear circuits in the time domain. The key ideas presented in this chapter revolved around the following steps:

1. Partitioning the circuit at the ports of the nonlinear devices, resulting in two sub-networks: an n-port linear sub-network and a multi-port network with n ports. The latter is algebraic, lacking memory elements, and models the currents in the nonlinear elements as outputs of an algebraic nonlinear function of the input voltages.

2. Inserting n time-domain polynomial auxiliary voltage sources at the $2n$ ports.

3. Utilizing the coefficients of the time-domain polynomial in the auxiliary voltage sources to enforce the equality of the time-domain derivatives of the currents

through these sources.

These ideas are used to cast the simulation process as a sequence of nonlinear algebraic problems that are solved, iteratively using NR method, at each time step during the time-marching process. At the l^{th} time step, corresponding to $t = lh$ (where h is the step size and l is an integer $l > 0$), the constraints to be satisfied are derived from enforcing the equality of the m^{th}-order derivatives ($m = 0, \ldots, p-1$) of the currents in the n auxiliary sources at both sides of the partition. In other words, these constraints form a system of $n \times p$ equations.

To finalize the problem setup, the following steps are still required:

1. Selection of the variables that will be used to satisfy the constraints, or the $n \times p$ equations. Those variables must be selected from the circuit variables, i.e., from its voltages or currents. Moreover, those variables need to be used in the computation of the polynomial coefficients representing the driving function of the auxiliary voltage sources.

2. Ensuring that the number of those variables is equal to the number of constraints. This step is mandatory to have a well-determined system of equations and to facilitate utilizing the NR iterative process to solve the system. It should be noted that the degree of the n polynomials, $\mathbf{u}_l(t)$, that was selected to represent the n auxiliary voltage sources was set at $q+p+1$, thereby providing essentially $n(p+q+2)$ degrees of freedom or independent variables, so as to satisfy the $n \times p$ constraints. Hence, one needs to revisit the number of independent variables and the number of equations again in this chapter.

3. Establishing a mechanism to guarantee the numerical stability of the time-stepping

process. This step is indispensable if the new method is to be practically viable for solving differential equations as a time stepping method.

The strategy used to enact the above steps will make the following choices:

- Use only $n(p+1)$ degrees of freedom to satisfy the problem constraints. The choice of those variables will detailed in Section 4.1.
- Augment the number of constraints with an additional set of n constraints. Those constraints will be pivotal in guaranteeing the numerical stability of the overall simulation, as will be shown in the proof of stability.

4.1 Selection of the Independent Variables

The independent variables used to ensure compliance with the constraints are derived from the voltages driving the nonlinear device, which were denoted by $\mathbf{v}_d(t)$. These voltages will be used as the basis for determining the coefficients of the polynomials $\mathbf{u}_l(t)$ which defined $\mathbf{u}_{\text{aux}}(t)$ in the interval $(l-1)h \leq t \leq lh$. For convenience, this polynomial is reproduced here

$$\mathbf{u}_l(t) = \sum_{i=0}^{p+q+1} \bar{\bar{\mathbf{u}}}_{l,i} \, (t - lh)^i, \quad (l-1)h \leq t \leq lh \tag{4.1}$$

The coefficients $\bar{\bar{\mathbf{u}}}_{l,i}$ are computed to meet the following conditions:

1. The values of $\mathbf{v}_d(t)$ and their time-domain derivatives, up to the p^{th} order, must be matched by $\mathbf{u}_l(t)$ at $t = lh$.

2. The values of $\mathbf{v}_d(t)$ and their time-domain derivatives, up to the q^{th} order, must be matched by $\mathbf{u}_l(t)$ at $t = (l-1)h$.

In mathematical terms, $\mathbf{u}_l(t)$ should satisfy

$$\frac{\mathrm{d}^m \mathbf{u}_l(t)}{\mathrm{d}t^m} = \frac{\mathrm{d}^m \mathbf{v}_d(t)}{\mathrm{d}t^m}, \qquad t = lh \quad ; \qquad m = 0, \cdots, p \tag{4.2}$$

$$\frac{\mathrm{d}^m \mathbf{u}_l(t)}{\mathrm{d}t^m} = \frac{\mathrm{d}^m \mathbf{v}_d(t)}{\mathrm{d}t^m}, \qquad t = (l-1)h \quad ; \qquad m = 0, \cdots, q \tag{4.3}$$

To mirror the notation developed earlier, we will use $\hat{\mathbf{v}}_l^{(m)}$ add $\hat{\mathbf{v}}_{l-1}^{(m)}$ to denote the following:

$$\hat{\mathbf{v}}_l^{(m)} \;\coloneqq\; h^m \left.\frac{\mathrm{d}^m \mathbf{v}_d(t)}{\mathrm{d}t^m}\right|_{t=lh}, \tag{4.4}$$

$$\hat{\mathbf{v}}_{l-1}^{(m)} \;\coloneqq\; h^m \left.\frac{\mathrm{d}^m \mathbf{v}_d(t)}{\mathrm{d}t^m}\right|_{t=(l-1)h}, \tag{4.5}$$

For convenience, we will use the scaled and shifted time variable $\hat{t}$ to express $\mathbf{u}_l(t)$. Thus, we have $\mathbf{u}_l(\hat{t}\,)$ which maps $\mathbf{u}_l(t)$ to the $\hat{t}$ domain through $t = h\hat{t} + (l-1)h$.

$$\mathbf{u}_l(\hat{t}\,) = \sum_{i=0}^{p+q+1} \hat{\bar{\mathbf{u}}}_{l,i} \, (\hat{t}-1)^{\,i}, \quad 0 \le \hat{t} \le 1 \tag{4.6}$$

where $\hat{\bar{\mathbf{u}}}_{l,i} = h^i \bar{\mathbf{u}}_{l,i}$. Furthermore, the two specified requirements in (4.2) and (4.3) can also be formulated in the $\hat{t}$ domain as follows

$$\frac{\mathrm{d}^m \mathbf{u}_l(\hat{t}\,)}{\mathrm{d}\hat{t}^m} = \hat{\mathbf{v}}_l^{(m)}, \qquad \hat{t} = 1 \quad ; \qquad m = 0, \cdots, p \tag{4.7}$$

$$\frac{\mathrm{d}^m \mathbf{u}_l(\hat{t}\,)}{\mathrm{d}\hat{t}^m} = \hat{\mathbf{v}}_{l-1}^{(m)}, \qquad \hat{t} = 0 \quad ; \qquad m = 0, \cdots, q \tag{4.8}$$

The notion of constructing a polynomial that interpolates a given function, such as $\mathbf{v}_d(t)$, and its derivatives at discrete points is known as Hermite interpolation, a concept that will be described next.

4.1.1 Hermite Interpolation

Hermite interpolation is a method of constructing a polynomial that not only matches the function values at given points but also the derivative values (and possibly higher derivatives) at those points. This makes it a more powerful tool than standard Lagrange interpolation [63], which only matches function values.

It is a well-known fact that there is a unique polynomial that can achieve this task. For example, there is a unique polynomial in x of degree 6 [64] that can be made equal to two pre-specified values at two points, say $x = x_1$ and $x = x_2$, while satisfying pre-specified derivatives of orders 1 and 2 at x_1 and orders 1,2, and 3 at x_2.

The task of constructing the Hermite interpolating polynomial is simply the task of computing the polynomial coefficients. Computing the polynomial coefficients has been typically carried out numerically through constructing and inverting a matrix that is determined based on the displacement between the points of interpolation. However, a relatively recent technique that appeared in [64] showed that this task need not be performed numerically. Rather, the interpolation coefficients can be analytically computed from the discrete time points and the values of the derivatives at those points.

The approach presented in [64] is based on the notion of cycle index polynomials. To introduce this notion, we use the notation $y_n\left(x_i | i \in [n]\right)$ to represent the multinomial $y_n\left(x_1, x_2, \cdots, x_n\right)$, and use that to describe the cycle index polynomials $z_n\left(x_i | i \in [n]\right)$

through the recurrence relation

$$\begin{aligned} z_0 &= 1 \\ n z_n\left(x_k | k \in [n]\right) &= \sum_{k=1}^{n} x_k z_{n-k}\left(x_i | i \in [n-k]\right) \end{aligned} \tag{4.9}$$

Next, define the following time points $t_1, t_2, \cdots, t_n$ as being different ($t_m \neq t_n$, if $m \neq n$) and not necessarily equi-distant. Furthermore, assume that a given function $\theta(t)$ is known through its derivatives: $\theta\left(t_\alpha\right)^{(\beta)}$ at those points, where $\alpha = 1, 2, \cdots, H$ and $\beta = 0, \cdots, m_\alpha - 1$. The Hermite interpolation problem is finding the polynomial $\Gamma(t)$ such that

$$\Gamma\left(t_\alpha\right)^{(\beta)} = \theta\left(t_\alpha\right)^{(\beta)}, \quad \alpha = 1, \cdots, H, \quad \beta = 0, \cdots, m_\alpha - 1 \tag{4.10}$$

The work presented in [64] has shown that this problem can be expressed analytically in terms of $\theta\left(t_\alpha\right)^{(\beta)}$ using the cycle index polynomials as illustrated by (4.11):

$$\Gamma(t) = \sum_{\alpha=1}^{H} \sum_{\beta=0}^{m_\alpha - 1} \frac{\theta\left(t_\alpha\right)^{(\beta)}}{\beta!} \prod_{\substack{\gamma=1 \\ \gamma \neq \alpha}}^{H} \left(\frac{t - t_\gamma}{t_\alpha - t_\gamma}\right)^{m_\gamma} \sum_{k=\beta}^{m_\alpha - 1} (t - t_\alpha)^k \, z_{k-\beta} \left(\sum_{\gamma \neq \alpha} \frac{m_\gamma}{(t_\gamma - t_\alpha)^r} \middle| \, r \in [k - \beta] \right) \tag{4.11}$$

It is to be noted here that the interpolant uses the coefficients of the cycle index polynomial along with the time points and the values/derivatives of the waveform to be interpolated, $\theta(t)$ at those points to construct the desired interpolant.

Section 4.1.2 build on the result given in (4.11) to construct a form of $\mathbf{u}_l(\hat{t}\,)$ that is expressed in terms of $\hat{\mathbf{v}}_l^{(i)}$ and $\hat{\mathbf{v}}_{l-1}^{(j)}$, $i = 0, 1, \cdots, p$ and $j = 0, 1, \cdots, p$.

4.1.2 Using Hermite Interpolation to construct $\mathbf{u}_l(\hat{t})$

Using (4.11) to come up with $\mathbf{u}_l(t)$ that satisfies (4.7) and (4.8) is achieved by noting the correspondence between the parameters in (4.11) and those requirements. In particular, setting those parameters as follows

$$H = 2 \tag{4.12}$$

$$t_1 = 1 \tag{4.13}$$

$$t_2 = 0 \tag{4.14}$$

$$\theta(t_1)^{(\beta)} = \hat{\mathbf{v}}_l^{(\beta)}, \qquad \beta = 0, 1, \cdots, p \tag{4.15}$$

$$\theta(t_2)^{(\beta)} = \hat{\mathbf{v}}_{l-1}^{(\beta)}, \qquad \beta = 0, 1, \cdots, q \tag{4.16}$$

in and using $\hat{t}$ in place of t in (4.11) provides the desired $\mathbf{u}_l(\hat{t})$ which takes the following form

$$\mathbf{u}_l(\hat{t}) = \sum_{j=0}^{q} \frac{\hat{\mathbf{v}}_{l-1}^{(j)}}{j!}(1-\hat{t})^{p+1} \sum_{k=j}^{q} \hat{t}^k \frac{(p+1)_{[k-j]}}{(k-j)!} + \sum_{j=0}^{p} \frac{\hat{\mathbf{v}}_l^{(j)}}{j!}\hat{t}^{q+1} \sum_{k=j}^{p} (-1)^{k-j} \left(\hat{t}-1\right)^k \frac{(q+1)_{[k-j]}}{(k-j)!} \tag{4.17}$$

where $m_{[n]}$ is used to denote the Pochhammer symbol defined by

$$m_{[n]} = \frac{(m+n-1)!}{(m-1)!}.$$

4.1.3 The Independent Variables

Section 4.1.2 introduced a process that transformed a subset of the circuit's MNA variables—specifically, the voltages controlling currents in the nonlinear device—into a polynomial expression used to drive auxiliary voltage sources.

Nonetheless, it is important to keep in mind that these circuit variables remain in the realm of "unknowns" that to be determined by the algorithm under development. Specifically, $\hat{\mathbf{v}}_{l-1}^{(m)}$ and $\hat{\mathbf{v}}_{l}^{(m)}$ are variables that the algorithm, meant to solve a system of differential equations (as an initial value problem), must compute.

However, for initial-value problems, $\hat{\mathbf{v}}_{l-1}^{(m)}$ can often be assumed as known. For instance, when $l = 1$, $\hat{\mathbf{v}}_{0}^{(m)}$ can be derived from the initial condition or the results of a Direct Current (DC) operating point analysis, at least for $m = 0$. For $m > 0$, a similar approach to that in [45] can be applied. Consequently, $\hat{\mathbf{v}}_{l}^{(m)}$ becomes the sole set of unknown used to formulate the expressions for the auxiliary sources and that needs to be computed.

The above observation thus points to $\hat{\mathbf{v}}_{l}^{(m)} \in \mathbb{R}^n, m = 0, 1, 2, \cdots, p$ as the set of independent variables that should be used to satisfy the $n \times p$ constraints represented by (3.12).

The problem (3.12) should thus be updated to reflect this choice. It should therefore read,

Given $\hat{\mathbf{v}}_{l-1}^{(m)}, m = 0, 1, \cdots, q$, (available from the past time point, $t = (l-1)h$)

Compute $\hat{\mathbf{v}}_{l}^{(m)}, m = 0, 1, \cdots, p$ (problem unknowns)

such that :

$$\hat{\mathbf{i}}_{l}^{(m)} = \hat{\mathbf{j}}_{l}^{(m)}, \quad m = 0, 1, \cdots, p-1 \tag{4.18}$$

The solution of this problem is considered in the subsequent sections.

4.2 Balancing the Unknowns and Constraints

The problem defined by (4.18) is an underdetermined one, as it includes fewer equations or constraints ($n \times p$) than unknowns ($n(p+1)$). Consequently, the formulation must be revised to ensure a unique solution can be determined. The required adjustment will be through including an additional set of n constraints. The additional set of constraints will thus serve a two-fold purpose. It will first create a well-determined problem with equal number of equations as the number of unknowns. More importantly, it will impart on the problem the sufficient condition that will guarantee the numerical stability.

The extra set of n constraints are given by

$$\sum_{j=0}^{p}(-1)^j \alpha_{q,p,j}\hat{\mathbf{v}}_l^{(j)} - \sum_{j=0}^{q}\alpha_{q,p,j}\hat{\mathbf{v}}_{l-1}^{(j)} = \mathbf{0} \tag{4.19}$$

where the triple sub-scripted coefficient $\alpha_{m,n,j}$ is defined by $\alpha_{m,n,j} = \frac{(m+n-j)!m!}{(m+n)!j!(m-j)!}$. Equation (4.19) is directly derived from Obreshkov equation.

The problem now reads as

$$\begin{aligned} &\text{Given } \hat{\mathbf{v}}_{l-1}^{(m)},\ m = 0, 1, \cdots, q, \\ &\text{compute } \hat{\mathbf{v}}_l^{(m)},\ m = 0, 1, \cdots, p, \quad \text{such that:} \\ &\qquad \hat{\mathbf{i}}_l^{(m)} \quad = \quad \hat{\mathbf{j}}_l^{(m)}, \quad m = 0, 1, \cdots, p-1 \\ &\sum_{j=0}^{p}(-1)^j \alpha_{p,q,j}\hat{\mathbf{v}}_l^{(j)} \quad = \quad \sum_{j=0}^{q}\alpha_{q,p,j}\hat{\mathbf{v}}_{l-1}^{(j)} \end{aligned} \tag{4.20}$$

The problem described in (4.20) represents the final form, whose solution is required to steer the simulation process. While $\hat{\mathbf{i}}_l^{(m)}$ is a linear function of $\hat{\mathbf{v}}_l^{(m)}$, $\hat{\mathbf{j}}_l^{(m)}$ introduces

nonlinearity as it depends on $\hat{\mathbf{v}}_l^{(m)}$ in a nonlinear manner. This characterization makes the overall problem nonlinear. Consequently, solving it requires standard approaches for handling nonlinear equations, such as the NR method. By applying this iterative technique, a solution for $\hat{\mathbf{v}}_l^{(m)}$ can be obtained once the method converges, ensuring the problem's unique solution is determined. Section 4.3 provides the complete details of an implementation strategy for solving this problem.

4.3 Complete Implementation Details

Utilizing the NR method to solve the problem (4.20) necessitates establishing fundamental computational steps.

- As with any NR, the first iteration requires a guess for the solution. The guess needs to be predicted sufficiently close to the exact solution to facilitate the convergence.

- The NR algorithm should be able to compute the error arising from the initial guess, or the subsequent trials. To do so, it must have an access to a method to compute this error via computing $\hat{\mathbf{i}}_l^{(m)}$ and $\hat{\mathbf{j}}_l^{(m)}$. Although this process has been sufficiently detailed in Section 3.5, it still lacks including details about $\mathbf{u}_l(\,\hat{t}\,)$ that were made in this chapter.

- The final requirement needed to run the NR is related to updating the initial guess and subsequent trial vectors. This process requires computing the Jacobian matrix, i.e., the matrix of partial derivatives, of both $\hat{\mathbf{i}}_l^{(m)}$ and $\hat{\mathbf{j}}_l^{(m)}$ with respect to the independent variables selected in this formulation, $\hat{\mathbf{v}}_l^{(m)};\ m = 0, 1, \cdots, p$.

The subsequent chapter addresses the above three items in more details.

4.3.1 The Initial NR Guess

The vector used as the initial guess is denoted here by $\boldsymbol{\xi}_l^{[0]}$. That vector will be constructed by taking the derivatives obtained from the last converged iterations in the previous time-step $t = (l-1)h$; i.e. $\hat{\mathbf{v}}_{(l-1)}^{(m)}, m = 0, 1, \cdots, q$; and padded by zeros to form the initial guess for $\hat{\mathbf{v}}_l^{(m)}$, $m = 0, 1, \cdots, p$.[1] Therefore, The initial guess based on the previous time point becomes,

$$\boldsymbol{\xi}_l^{[0]} \leftarrow \left[\hat{\mathbf{v}}_{l-1}^{(0)\top}, \hat{\mathbf{v}}_{l-1}^{(1)\top}, \cdots, \hat{\mathbf{v}}_{l-1}^{(q)\top}, \mathbf{0}, \cdots, \mathbf{0}\right]^\top \tag{4.21}$$

Upon convergence of the NR iterations, say after κ iterations, the solution $\boldsymbol{\xi}_l^{[\kappa]}$, arrived at the κ^{th} iteration will be used for $\hat{\mathbf{v}}_l^{(m)}$; $m = 1, 2 \cdots, p$, i.e.,

$$\left[\hat{\mathbf{v}}_l^{(0)\top}, \hat{\mathbf{v}}_l^{(1)\top}, \cdots, \hat{\mathbf{v}}_l^{(p)\top}\right]^\top \leftarrow \boldsymbol{\xi}_l^{[\kappa]} \tag{4.22}$$

4.3.2 Detailed Computation of $\hat{\mathbf{i}}_l^{(m)}$ and $\hat{\mathbf{j}}_l^{(m)}$

The other essential step needed in execution of the NR iteration is the evaluation of $\hat{\mathbf{i}}_l^{(m)}$ and $\hat{\mathbf{j}}_l^{(m)}$, $m = 0, 1 \cdots, p-1$. The implementation of this step was described in Section 3.5.

The process of computing $\hat{\mathbf{i}}_l^{(m)}$ will proceed using the NILT framework as explained in Section 3.5.1. The additional details that should be included in this section is the specification of $\mathbf{u}_l(\hat{t})$ given by (4.17) and its Laplace-domain form $\mathbf{U}_l(\hat{s})$.

[1] As we shall be demonstrated later q has to be chosen such that, $q \leq p$ to satisfy the numerical stability requirement.

It would be convenient for the sake of conciseness of the derivations in the subsequent sections to assemble the relation between the independent variables $\hat{\mathbf{v}}_l^{(j)}$; $j = 0, 1, \cdots, p$ and the auxiliary currents in linear partition $\hat{\mathbf{i}}_l^{(m)}$; $m = 0, 1, \cdots, p-1$ in a single expression. This can be done by first writing the Laplace transform expression of $\mathbf{u}_l(\hat{t})$ with respect to $\hat{t}$, explicitly in terms of $\hat{s}$. To this end, we define $\theta_{l-1,j}(\hat{s})$ and $\theta_{l,j}(\hat{s})$, through

$$\begin{aligned}\theta_{l-1,j}(\hat{s}) &= \frac{1}{j!}\sum_{v=0}^{p+1}\sum_{k=j}^{q}\frac{(-1)^v(p+1)!(k+v)!(p+1)_{[k-j]}}{(p+1-v)!(k-j)!v!\hat{s}^{k+v+1}}\\ \theta_{l,j}(\hat{s}) &= \frac{1}{j!}\sum_{k=j}^{p}\sum_{v=0}^{k}\frac{(-1)^{v+j}(v+q+1)!k!(q+1)_{[k-j]}}{(k-v)!(k-j)!v!\hat{s}^{v+q+2}}\end{aligned} \tag{4.23}$$

The above two expressions are obtained from taking the Laplace transformations for the j^{th} term under the two summations in (4.17). Based on that, $\mathbf{U}_l(\hat{s})$ can be written as,

$$\mathbf{U}_l(\hat{s}) = \sum_{j=0}^{q}\theta_{l-1,j}(\hat{s})\hat{\mathbf{v}}_{l-1}^{(j)} + \sum_{j=0}^{p}\theta_{l,j}(\hat{s})\hat{\mathbf{v}}_l^{(j)} \tag{4.24}$$

Now that $\mathbf{u}_l(\hat{t})$ is identified, $\hat{\mathbf{i}}_l^{(m)}$ can be explicitly expressed in terms of $\hat{\mathbf{v}}_l^{(j)}$ as shown by,

$$\hat{\mathbf{i}}_l^{(m)} = -\sum_{i=1}^{M/2} 2\Re\left\{k_i z_i^m \bar{\mathbf{d}}\left(z_i\bar{\mathbf{C}}+\bar{\mathbf{G}}\right)^{-1}\begin{bmatrix}\frac{1}{h}\bar{\mathbf{C}}\tilde{\mathbf{x}}_{l-1}\\ \sum_{j=0}^{q}\theta_{l-1,j}(z_i)\hat{\mathbf{v}}_{l-1}^{(j)} + \sum_{j=0}^{p}\theta_{l,j}(z_i)\hat{\mathbf{v}}_l^{(j)}\end{bmatrix} - k_i\sum_{j=1}^{m-1} z_i^{m-j}\,\hat{\mathbf{i}}_{l-1}^{(j-1)}\right\} \tag{4.25}$$

As to computing of $\hat{\mathbf{j}}_l^{(m)}$, this is carried out in a similar way to the description given in Section 3.5.2. The only addition to be made here is that the leaf nodes in the rooted tree marked as 'W' will get their values assigned directly from the elements in $\boldsymbol{\xi}_l^{[\kappa]}$ at the

beginning of each iteration. It should also be noted here that an expression in which $\hat{\mathbf{v}}_l^{(j)}$ is explicitly present, analogous to (4.25), is not possible to develop for $\hat{\mathbf{j}}_l^{(m)}$. This is attributable to the fact that the relation between $\hat{\mathbf{v}}_l^{(j)}$ and $\hat{\mathbf{j}}_l^{(m)}$ depends on the actual expression of the nonlinear function $\phi(\cdot)$, that models the nonlinearity, and the rooted tree used to represent that nonlinearity.

4.3.3 Computing the Jacobian Matrix

The third key detail required in the implementation process is computing the Jacobian matrix. First, we look at the outline structure of this matrix.

The Jacobian matrix of the set of equations in (3.7) appended by the set of constraints in (4.19) will assume the following structure,

$$\mathbf{J} = \left[\begin{array}{c} \underset{np \times n(p+1)}{\boxed{\mathbf{J}_L + \mathbf{J}_R}} \\ \hline (\alpha_{p,q,0}\mathbb{I}_n) \quad (-\alpha_{p,q,1}\mathbb{I}_n) \quad \cdots \quad ((-1)^p\alpha_{p,q,p}\mathbb{I}_n) \end{array}\right] \tag{4.26}$$

where $\mathbf{J}_L$ and $\mathbf{J}_R$ denote, respectively, the contributions of the left (linear) and right (nonlinear) partition of the split structure, and $\mathbb{I}_n$ is an identity matrix of size n. The matrix arising from each partition is derived in the next sections.

4.3.3.1 The Linear Part of The Jacobian Matrix $\mathbf{J}_L$

The Jacobian matrix $\mathbf{J}_L \in \mathbb{R}^{np \times n(p+1)}$, is a block-structured matrix defined by

$$\mathbf{J}_L \quad = \quad \left[\frac{\partial \hat{\mathbf{i}}_l^{(\rho_1)}}{\partial \hat{\mathbf{v}}_l^{(\rho_2)}}\right]_{\substack{\rho_1=0,1,2,\cdots,p-1 \\ \rho_2=0,1,2,\cdots,p}} \tag{4.27}$$

where $\frac{\partial \hat{\mathbf{i}}_l^{(\rho_1)}}{\partial \hat{\mathbf{v}}_l^{(\rho_2)}}$ is a block in $\mathbb{R}^{n\times n}$.

Since this matrix belongs to the linear part of the circuit, it is independent from $\hat{\mathbf{v}}_l^{(m)}$, and therefore remains constant throughout the simulation. $\frac{\partial \hat{\mathbf{i}}_l^{(\rho_1)}}{\partial \hat{\mathbf{v}}_l^{(\rho_2)}}$ can be obtained from (4.25) by differentiating the right-side with respect to $\hat{\mathbf{v}}_l^{(\rho_2)}$, resulting in

$$\frac{\partial \hat{\mathbf{i}}_l^{(\rho_1)}}{\partial \hat{\mathbf{v}}_l^{(\rho_2)}} = -\sum_{i=1}^{M/2} 2\Re\left(z_i^{\rho_1} k_i \bar{\mathbf{d}} \left(\frac{z_i}{h}\bar{\mathbf{C}} + \bar{\mathbf{G}}\right)^{-1} \begin{bmatrix} \mathbf{0} \\ \\ \theta_{l,\rho_2}(z_i)\mathbb{I}_n \end{bmatrix} \right) \tag{4.28}$$

4.3.3.2 The Nonlinear Part of the Jacobian matrix $\mathbf{J}_R$

The matrix $\mathbf{J}_R \in \mathbb{R}^{np\times n(p+1)}$ is given by

$$\mathbf{J}_R = \left[\frac{\partial \hat{\mathbf{j}}_l^{(\rho_1)}}{\partial \hat{\mathbf{v}}_l^{(\rho_2)}}\right] = \left[\frac{1}{\rho_2!}\cdot\frac{\partial \hat{\mathbf{j}}_l^{(\rho_1)}}{\partial \hat{\mathbf{u}}_{l,\rho_2}}\right]_{\substack{\rho_1=0,1,2,\cdots,p-1 \\ \rho_2=0,1,2,\cdots,p}} \tag{4.29}$$

Consider an arbitrary block in $\mathbf{J}_R$, such as $\frac{\partial \hat{\mathbf{j}}_l^{(\rho_1)}}{\partial \hat{\mathbf{u}}_{l,\rho_2}}$, where $\rho_1 > 0$. Starting with the definition of $\hat{\mathbf{j}}_l^{(\rho_1)}$ given in (3.8), this block can be analyzed through the following steps

$$\begin{aligned} \frac{\partial \hat{\mathbf{j}}_l^{(\rho_1)}}{\partial \hat{\mathbf{u}}_{l,\rho_2}} &= h^{\rho_1}\frac{\partial}{\partial \hat{\mathbf{u}}_{l,\rho_2}}\left.\frac{\mathrm{d}^{\rho_1}\mathbf{j}_l\left(\mathbf{u}_l(t)\right)}{\mathrm{d}t^{\rho_1}}\right|_{t=lh} \\ &= h^{\rho_1}\frac{\partial}{\partial \hat{\mathbf{u}}_{l,\rho_2}}\frac{\mathrm{d}^{\rho_1-1}}{\mathrm{d}t^{\rho_1-1}}\left(\frac{\mathrm{d}\mathbf{j}_l\left(\mathbf{u}_l(t)\right)}{\mathrm{d}t}\right)\Bigg|_{t=lh},\ \rho_1 > 0 \end{aligned} \tag{4.30}$$

Using the chain rule of differentiation in the matrix between the brackets,

$$\frac{\mathrm{d}\mathbf{j}_l\left(\mathbf{u}(t)\right)}{\mathrm{d}t} = \frac{\partial \mathbf{j}_l\left(\mathbf{u}_l(t)\right)}{\partial \mathbf{u}_l}\frac{\mathrm{d}\mathbf{u}_l(t)}{\mathrm{d}t} \tag{4.31}$$

where $\frac{\partial \mathbf{j}_l(\mathbf{u}_l(t))}{\partial \mathbf{u}_l}$ is a matrix of partial derivatives,

$$\frac{\partial \mathbf{j}_l\left(\mathbf{u}_l(t)\right)}{\partial \mathbf{u}_l} = \left[\frac{\partial j_{n_r}}{\partial u_{n_c}}\right]_{\substack{n_r=1,2\cdots,n \\ n_c=1,2\cdots,n}} \tag{4.32}$$

This matrix, considered a function of time t, can be expanded in a Taylor series in $\hat{t}$ using

$$\frac{\partial \mathbf{j}_l\left(\mathbf{u}(t)\right)}{\partial \mathbf{u}} = \sum_{\mu_2=0}^{\infty} \frac{1}{\mu_2!} \hat{\mathbf{R}}_l^{(\mu_2)} \left(\hat{t}-1\right)^{\mu_2} \tag{4.33}$$

where $\hat{\mathbf{R}}_l^{(\mu_2)}$ is the (matrix-valued) derivative, w.r.t. $\hat{t}$, of $\frac{\partial \mathbf{j}(\mathbf{u}(t))}{\partial \mathbf{u}}$ computed at $\hat{t}=1$ or $t=lh$.

Substituting from (4.33) and (3.19) into (4.31) and proceeding in the domain of $\hat{t}$, we get

$$\frac{\mathrm{d}\mathbf{j}_l\left(\mathbf{u}(t)\right)}{\mathrm{d}\hat{t}} = \sum_{\mu_1=1}^{p+q+1} \sum_{\mu_2=0}^{\infty} \frac{\mu_1}{\mu_2!} \hat{\mathbf{R}}_l^{(\mu_2)} \hat{\mathbf{u}}_{l,\mu_1} \left(\hat{t}-1\right)^{\mu_1+\mu_2-1} \tag{4.34}$$

Differentiating the above expression $\rho_1 - 1$ times with respect to $\hat{t}$ substituting $\hat{t}=1$ (equivalent to $t=lh$), leaves only terms where the power $\mu_1+\mu_2-1=\rho_1-1$ or $\mu_2=\rho_1-\mu_1$, with $1 \le \mu_1 \le \rho_1$. Thus

$$\frac{\mathrm{d}^{\rho_1-1}}{\mathrm{d}\hat{t}^{\rho_1-1}} \left(\frac{\mathrm{d}\mathbf{j}_l\left(\mathbf{u}(t)\right)}{\mathrm{d}\hat{t}}\right) = \sum_{\mu_1=1}^{\rho_1} \frac{\mu_1(\rho_1-1)!}{(\rho_1-\mu_1)!} \hat{\mathbf{R}}_l^{(\rho_1-\mu_1)} \hat{\mathbf{u}}_{l,\mu_1} \tag{4.35}$$

Substituting from (4.35) into (4.30) retains the term with $\mu_1=\rho_2$ producing

$$\frac{\partial \hat{\mathbf{j}}_l^{(\rho_1)}}{\partial \hat{\mathbf{u}}_{l,\rho_2}} = \begin{cases} \frac{\rho_2\,(\rho_1-1)!}{(\rho_1-\rho_2)!} \hat{\mathbf{R}}_l^{(\rho_1-\rho_2)} & \rho_1 \ge \rho_2 \\ \mathbf{0} & \rho_1 < \rho_2 \end{cases} \tag{4.36}$$

A similar procedure can be repeated for the case $\rho_1 = 0$,

$$\frac{\partial \hat{\mathbf{j}}_l^{(0)}}{\partial \hat{\mathbf{u}}_{l,\rho_2}} = \begin{cases} \hat{\mathbf{R}}_l^{(0)} & \rho_2 = 0 \\ \mathbf{0} & \rho_2 > 0 \end{cases} \tag{4.37}$$

Using (4.36) and (4.37) into (4.29), the desired Jacobian matrix is formulated in the following block-structured matrix $\mathbf{J}_R$, expressed as

$$\mathbf{J}_R = \sum_{\rho=0}^{p} \mathbf{T}^{\rho} \otimes \hat{\mathbf{R}}_l^{(\rho)} \tag{4.38}$$

where $\mathbf{T} \in \mathbb{R}^{p \times p+1}$ is zero matrix except for the first sub-diagonal whose entries are $1, 2, \cdots p-1$ defined as

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & \cdots & p-1 & 0 \end{bmatrix} \tag{4.39}$$

Computation of the blocks $\hat{\mathbf{R}}_l^{(\mu_2)}$ is done similar to computing $\hat{\mathbf{v}}_l^{(m)}$, based on the concept of rooted trees. Here, as in the case of $\hat{\mathbf{v}}_l^{(m)}$, a rooted tree is constructed for each nonlinear expression entry in the matrix of partial derivatives $\frac{\partial \mathbf{j}_l(\mathbf{u}_l(t))}{\partial \mathbf{u}_l}$.

The operation of rooted trees, explained in Section 3.5.2, can also be used to compute $\hat{\mathbf{R}}_l^{(\mu_2+1)}$ from $\hat{\mathbf{R}}_l^{(m)}$ and $\hat{\mathbf{v}}_l^{(m)}$; $m = 0, 1, \cdots, \mu_2$.

4.3.4 Summary and Pseudo-code Representation

The procedure outlined in Algorithm 1 depicts the main computational building blocks of the proposed approach in the form of a pseudo-code. A point worthy of note on this procedure is the input parameters to the algorithm which include $\mathbf{x}_0^{(m)}$ or the m^{th}-order time-domain derivative (up to order $m = q$) of the circuit variables $\mathbf{x}(t)$ at the initial point $t = 0$. Computing those derivatives is carried out through the concept of RT given a set of initial condition $\mathbf{x}_0^{(0)}$, which can be obtained using a DC operating point analysis. High-order derivatives, $m > 0$, are computed recursively from the lower order ones. This process is described in more details in [45]. Another point worthy of note is the assignment on line 7, which shows the initialization of the trial vector from the first q derivatives found from the previous $(l-1)$ time step.

It should also be added that the proposed approach can be used to recover the value of any variable within the linear sub-network. Indeed, upon the convergence of the NR method, then using a suitable selector matrix, other than $\bar{\mathbf{d}}$ in (4.25) for $m = 0$, will recover the time domain of the desired variable at the required point in time.

Algorithm 1: Outline of the Proposed Algorithm

input : $h, p, L, \mathbf{x}_0^{(m)}, m = 0, 1, \cdots, q,$

$\hat{\mathbf{v}}_0^{(m)} \leftarrow \mathbf{d}^\top h^m \mathbf{x}_0^{(m)}$;

Split the circuit as indicated in Figure 3.2 ;

Create RTs for the n nonlinear expression in $\mathbf{\Phi}_R$;

Create RTs for the $n \times n$ entries in the matrix $\frac{\partial \mathbf{\Phi}_R}{\partial \mathbf{u}}$;

Set $\epsilon \leftarrow 10^{-14}, \quad l \leftarrow 1$;

while $l \leq L$ **do**

  $\boldsymbol{\xi}_l^{[0]} \leftarrow \left[\hat{\mathbf{v}}_{l-1}^{(0)\top}, \hat{\mathbf{v}}_{l-1}^{(1)\top}, \cdots, \hat{\mathbf{v}}_{l-1}^{(q)\top}, \mathbf{0}, \cdots, \mathbf{0} \right]^\top$;

  Set $\phi \leftarrow \epsilon + 10$ and $k \leftarrow 1$;

  while $\phi > \epsilon$ and $k \leq \kappa$ **do**

    $\left[\hat{\mathbf{v}}_l^{(0)\top}, \hat{\mathbf{v}}_l^{(1)\top}, \cdots, \hat{\mathbf{v}}_l^{(p)\top} \right]^\top \leftarrow \boldsymbol{\xi}_l^{[k]}$;

    Use (4.25) to compute $\hat{\mathbf{i}}_l^{(m)}$, $m = 0, 1, \cdots, p-1$;

    Leaf nodes of RTs of $\mathbf{\Phi}_R \leftarrow \hat{\mathbf{v}}_l^{(m)}$;

    Compute $\hat{\mathbf{j}}_l^{(m)}$ for $m = 0, 1, \cdots, p-1$;

    $\boldsymbol{\delta} \leftarrow \sum_{j=0}^{p} (-1)^j \alpha_{p,q,j} \hat{\mathbf{v}}_l^{(j)} - \sum_{j=0}^{q} \alpha_{q,p,j} \hat{\mathbf{v}}_{l-1}^{(j)}$;

    $\mathbf{\Psi} \leftarrow \left[\left(\hat{\mathbf{i}}_l^{(0)} - \hat{\mathbf{j}}_l^{(0)} \right)^\top \cdots \left(\hat{\mathbf{i}}_l^{(p-1)} - \hat{\mathbf{j}}_l^{(p-1)} \right)^\top \boldsymbol{\delta}^\top \right]^\top \quad \phi \leftarrow ||\mathbf{\Psi}||$;

    Use (4.27) and (4.28) to compute $\mathbf{J}_L$;

    Using RTs of $\frac{\partial \mathbf{\Phi}_R}{\partial \mathbf{u}}$ compute $\hat{\mathbf{R}}_l^{(m)}$, then substitute in (4.38) to compute $\mathbf{J}_R$;

    Use (4.26) to compute $\mathbf{J}$;

    $\boldsymbol{\xi}_l^{[k+1]} \leftarrow \boldsymbol{\xi}_l^{[k]} - \mathbf{J}^{-1} \mathbf{\Psi}$;

    $k \leftarrow k + 1$;

  $l \leftarrow l + 1$;

Chapter 5

Order and Stability Characterization

The algorithm introduced in the previous chapter represents a novel approach to numerically solving general differential equations, with a specific focus on equations derived from modeling nonlinear circuits in the time domain (the MNA formulation). Moreover, the framework established for implementing this method has been designed to facilitate a fully automated process.

Nevertheless, for the new method to be critically examined on par with existing methods, it needs to be characterized using the tools established in the literature under the classical theory of the numerical methods in differential equations [23,40,65], many elements of which have been reviewed in Chapter 2.

The goal of this chapter is to provide the theoretical characterization of the method. This characterization will consider the approximation order of the method as well as its stability criteria.

Studying the order and stability criteria of the proposed method is done using the Dahlquist approach [16] which characterizes a given method for solving differential

equations through studying its behavior in approximating the solution to the scalar test problem

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \lambda x(t) \tag{5.1}$$

with some initial condition $x(0) = x_0$ and a complex parameter λ. The exact solution to this problem is well-known and is given by

$$x(t) = \mathrm{e}^{\lambda t}\, x_0 \tag{5.2}$$

This solution converges to 0 as $t \to \infty$ for $\Re\{\lambda\} < 0$.

5.1 Order of Convergence Characterization

Given that the method was proposed in the context of a circuit simulation process, its characterization in terms of studying its behaviour in solving the test problem (5.1) requires adapting that problem as a circuit structure.

The proposed method can be applied to the scalar test problem (5.1) through considering a circuit of a resistor of $(-1/\lambda)\Omega$ and a capacitor of $1F$ connected in parallel. In this case $x(t)$ is the voltage across the parallel structure, $v(t)$, and $v(0) = x_0$. Such a circuit is shown in Figure 5.1.

The idea of splitting the circuit can be carried out by taking the capacitor as the circuit on the left side, connecting an auxiliary source with one terminal, i.e., $n = 1$, while leaving the resistor to the circuit on the right side of the split with the same auxiliary source. The split circuit is illustrated in Figure 5.2.

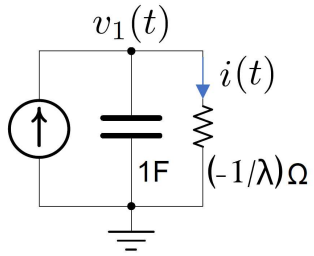


Figure 5.1: Circuit used to represent the scalar test problem of (5.1)

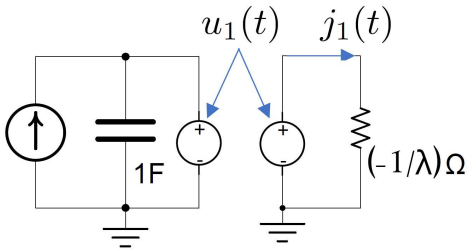


Figure 5.2: Splitting the circuit of problem (5.1) to implement the proposed approach

The order of a given numerical method is typically defined as the number of the first Taylor series coefficients in the exact solution of (5.1), $x(t)$, at $t = h$ ($x(h) = \mathrm{e}^{\lambda h} x_0$) that are matched by the approximation generated from the method, x_1 when expanded as a Taylor series in h.

The initial conditions in starting the proposed method, or $\hat{\mathbf{v}}_0^{(m)}$, are scalars and given by $\lambda^m x_0$, where $m = 0, 1, 2, \cdots, q$. Furthermore, the scalar test problem settings entail that the terms used in the proposed approach be given by $\bar{\mathbf{d}} \left(z_i \bar{\mathbf{C}} + \bar{\mathbf{G}} \right)^{-1} = \left[1 \ -\frac{z_i}{h} \right]$, $\frac{1}{h} \mathbf{C} \tilde{\mathbf{x}}_0 = \frac{x_0}{h}$, with scalar auxiliary source $u(t)$ (Laplace-domain $U(s)$) and scalar currents $i(t)$ and $j(t)$, noting that $j(t) = -\lambda u(t)$.

The starting point in the characterization of the order of the method is to state an important property of NILT using the following lemma.

Lemma 5.1. *Let NILT be used with integer parameters N and M such that $N + M + 1 \geq m$ for some integer m. Then,*

$$\mathscr{L}^{-1}\left\{\frac{1}{s^m}\right\}(t)\Big|_{t=h} = \widehat{\mathscr{L}}^{-1}_{\text{NILT}}\left\{\frac{1}{s^m}\right\}(\hat{t})\Big|_{\hat{t}=1} \tag{5.3}$$

Proof of this lemma can be found in [18, Ch 10].

Next, the following theorem provides the order of the method and the sufficient conditions that establish this order.

Theorem 5.1. *Let the proposed splitting method be implemented with parameters q and p, and assume that the NILT technique is used on the linear part with parameters N and M. Then the order of the method is given by $p + q$.*

Proof of Theorem 5.1. From the scalar test problem settings, applying (4.24) and (3.15) in (3.14) while setting $l = 1$, and noting that $\sum_{i=1}^{M/2} k_i = 0$, yields

$$i_1^{(0)} = \frac{1}{h}\sum_{i=2}^{M/2} z_i k_i U_1(z_i) \tag{5.4}$$

Seeing that the right-side of (5.4) is equal to $-\widehat{\mathscr{L}}^{-1}_{\text{NILT}}\{\hat{s}U_l(\hat{s})\}(\hat{t}\,)$ at $\hat{t} = 1$ ($t = lh, l = 1$) and noting that $\hat{s}U_1(\hat{s})$ is a summation of terms proportional to $\frac{1}{\hat{s}^m}$ with $m \leq p+q+1$, then it follows from the assumption of the theorem ($N + M + 1 \geq p + q + 1$) and the result of Lemma 5.1 that $-\widehat{\mathscr{L}}^{-1}_{\text{NILT}}\{\hat{s}U_1(\hat{s})\}(1)$ must be equal to $-\mathscr{L}^{-1}\{sU_1(s)\}(t)$ at $t = h$. Since, by the differentiation property of the Laplace transformation, we have $\mathscr{L}^{-1}\{sU_1(s)\}(t) = \frac{\mathrm{d}u_1(t)}{\mathrm{d}t}$ for $t > 0$, then

$$i_1^{(0)} = \frac{-1}{h}\frac{\mathrm{d}u_1(t)}{\mathrm{d}t}\Big|_{t=lh} \tag{5.5}$$

Since $u_1(t)$ is created such that it has the same (but hitherto unknown) first derivative as the auxiliary source at $t = h$, or $v_1^{(0)}$, then

$$i_1^{(0)} = -\frac{v_1^{(1)}}{h} \tag{5.6}$$

In the same vein, the above manipulation can be repeated by substituting into (4.25) for $m > 0$, leading to

$$i_1^{(m)} = -\frac{v_1^{(m+1)}}{h}, \quad m = 1, 2, \cdots, p-1 \tag{5.7}$$

The fact that $j(t) = -\lambda u(t)$ necessitates $j_1^{(m)} = -\lambda \left.\frac{\mathrm{d}^m u(t)}{\mathrm{d}t^m}\right|_{t=lh}$, $m = 0, 1, \cdots, p$. By the properties built in $u(t)$, namely that its derivatives interpolates $v_1^{(m)}$, we have $\left.\frac{\mathrm{d}^m u(t)}{\mathrm{d}t^m}\right|_{t=lh} = v_1^{(m)}$, and hence,

$$j_1^{(m)} = \lambda v_1^{(m)}, \quad m = 0, 1, 2, \cdots, p-1 \tag{5.8}$$

Substituting from (5.6), (5.7), and (5.8) into (3.7) and then appending the constraints of (4.19) ($n = 1, l = 1$) leads to the matrix form $\mathbf{B}\bar{\tilde{\mathbf{v}}}_{1,1} = \mathbf{e}_{p+1}\boldsymbol{\Psi}_0^\top \bar{\tilde{\mathbf{v}}}_{1,0}$ where $\mathbf{e}_i$ denotes i^{th} column in an identity matrix of size $p+1$, and

$$\begin{aligned}
\bar{\tilde{\mathbf{v}}}_{1,0} &= \begin{bmatrix} x_0 & \lambda x_0 & \cdots & \lambda^q x_0 \end{bmatrix}^\top \\
\bar{\tilde{\mathbf{v}}}_{1,1} &= \begin{bmatrix} v_1^{(0)} & v_1^{(1)} & \cdots & v_1^{(p)} \end{bmatrix}^\top \\
\boldsymbol{\Psi}_0 &= \begin{bmatrix} \alpha_{q,p,0} & \alpha_{q,p,1} & \cdots & \alpha_{q,p,q} \end{bmatrix}^\top
\end{aligned}$$

$$\mathbf{B} = \begin{bmatrix} -\lambda & 1/h & 0 & \cdots & 0 & \\ 0 & -\lambda & 1/h & 0 & \cdots & 0 \\ & & \ddots & \ddots & & \\ \alpha_{p,q,0} & -\alpha_{p,q,1} & \cdots & & & (-1)^p \alpha_{p,q,p} \end{bmatrix}$$

For the matrix $\mathbf{B}$, its determinant $\det(\mathbf{B})$ is

$$\det(\mathbf{B}) = (-1)^p \sum_{i=0}^{p} \alpha_{p,q,i} \, (-\lambda)^i \, h^{(i-p)} \tag{5.9}$$

whereas the last column of its adjugate is given by,

$$(-1)^p \begin{bmatrix} h^{-p} & \lambda h^{-p+1} & \cdots & \lambda^{p-1} h^{-1} & \lambda^p \end{bmatrix}^\top$$

This makes $v_1^{(0)}$ computed by the proposed method equal to

$$v_1^{(0)} = \mathbf{e}_1^\top \mathbf{B}^{-1} \mathbf{e}_{p+1} \boldsymbol{\Psi}_0^\top \bar{\bar{\mathbf{v}}}_0 = \underbrace{\left(\frac{\sum_{i=0}^{q} \alpha_{q,p,i} (\lambda h)^i}{\sum_{i=0}^{p} (-1)^i \alpha_{p,q,i} (\lambda h)^i} \right)}_{R_{q,p}(\lambda h)} x_0$$

The above fraction between the brackets can be recognized as the (q, p) rational Padé approximant to the exponential function [47, 48], $R_{q,p}(\lambda h)$.

The order of the method can be deduced from the following identity of $R_{q,p}(z)$ [23], namely,

$$\mathrm{e}^z - R_{q,p}(z) = \frac{(-1)^q q! p! z^{q+p+1}}{(q+p)! \, (q+p+1)!} + \mathcal{O}\left(z^{q+p+2}\right) \tag{5.10}$$

which shows clearly that the approximation obtained by the proposed method after

taking time step h (i.e., $v_1^{(0)}$) has a Taylor series expansion that agrees with the Taylor series of the exact waveform $e^{\lambda h}x_0$ in the first $q+p$ terms. This fact establishes that the proposed method is indeed of order $q+p$, and therefore completes the proof of the theorem. □

5.2 Stability Characterization

The following theorem states the sufficient condition that makes the proposed method of the A-stable or L-stable type.

Theorem 5.2. *Let the proposed splitting method be implemented with parameters q and p, and assume that the NILT technique is used on the linear part with parameters N and M. Then the method is A-stable if $N+M \geq p+q$ and $p-2 \leq q \leq p$ and L-stable if $N+M \geq p+q$ and $p-2 \leq q < p$.*

Proof of Theorem 5.2. It follows in the same steps as shown in the proof of theorem 5.1, leading up to (5.1). Based on that result in (5.1), it follows that the approximations obtained for the voltage using the proposed method on the scalar test problem after marching in time n steps will be

$$v_n^{(0)} = (R_{q,p}(\lambda h))^n \; x_0 \tag{5.11}$$

An important property of the Padé approximants is that, for all $z \in \mathbb{C}^-$, $|R_{q,p}(z)| < 1$ if, and only if, $p-2 \leq q \leq p$, and also that, for all $z \in \mathbb{C}^- \cup \{\infty\}$, $|R_{q,p}(z)| < 1$ if, and only if, $p-2 \leq q < p$, [22, 23]. This property thus ensures that $v_i^{(0)} < v_j^{(0)}$, $(i > j)$ for all $z \in \mathbb{C}^-$ and hence the A-stability under the conditions that $p-2 \leq q \leq p$ is proved.

The L-stability is guaranteed for $p-2 \leq q < p$, proving that the proposed method is both L-stable and A-stable. □

5.3 Experimental Validation of Order of Convergence

Theorem 5.1 establishes the theoretical order of convergence for the newly proposed NILT method. It is important to note, however, that this proof is based on a scalar test problem. While the test problem is linear and the proposed method is designed for general nonlinear circuits, this approach is widely regarded by researchers as an adequate characterization of the method's convergence behavior for broader problem classes. Nonetheless, it is often times of interest to validate this theoretical order of convergence through experimental verification, particularly in the context of nonlinear problems.

The objective of this section is to demonstrate experimentally that the theoretical results of theorem 5.1 are valid beyond the scalar test problem used in the proof of this theorem. The circuit in this example is the circuit shown in Figure 3.3, and is executed by exciting it with a sinusoidal current source steady-state, and computing its steady-state response, denoted here $\mathbf{y}_{\text{ss}}(t)$. In order to avoid contaminating $\mathbf{y}_{\text{ss}}(t)$ with the error that inevitably arises from any numerical time stepping technique, the HB technique [7,66,67] has been used. HB first expresses $\mathbf{y}_{\text{ss}}(t)$ as a Fourier series expansion with unknown coefficients and then computes these coefficients through solving an algebraic problem, which is free from the approximation errors that erupt in the time stepping[1]. Thus, $\mathbf{y}_{\text{ss}}(t)$ computed from HB can be used as a reference in this special

[1]Needless to say that this technique is not applicable to the circuits targeted in this paper, since the stimulus in those circuits is typically non-sinusoidal and the circuits are typically large to make HB practical.

circuit against which the results of the proposed approach can be compared. It is important to stress here that HB is still considered a numerical solution, and is only selected for comparison here due to it being free from the truncation error that comes with regular methods used in solving differential equations.

To demonstrate numerically the validity of Theorem 5.1 (given in (5.10)), we start the proposed approach from any arbitrary point on $\mathbf{y}_{\rm ss}(t)$, say at $\mathbf{y}_{\rm ss}(0)$, by setting $\mathbf{x}_0^{(0)} = \mathbf{y}_{\rm ss}(0)$ and computing the higher derivatives $\mathbf{x}_0^{(m)}, m = 1, \cdots, q$ at $t = 0$ as explained. The proposed approach, using the splitting indicated on the figure, is then used to advance to time $t = h$, computing $\mathbf{x}_1^{(m)}$. The value of $\left|\mathbf{x}_1^{(0)} - \mathbf{y}_{\rm ss}(h)\right|$ may therefore be used to examine the approximation error resulting from single step h and compare its behavior, with regards to h, against the behavior predicted by the above theorem.

Figure 5.3 plots, using a log graph, $\left|\mathbf{x}_1^{(0)} - \mathbf{y}_{\rm ss}(h)\right|$ versus the length of the time step h, for the two cases $p = 2$, $q = 0$ and $p = 3$, $q = 1$. Here, the NILT part of the proposed approach was implemented with $M = 6$ and $N = 4$. According to the result of the theorem in (5.10), this value (the approximation error $\left|\mathbf{x}_1^{(0)} - \mathbf{y}_{\rm ss}(h)\right|$) should asymptotically be proportional to h^{p+q+1} in the limit $h \to 0$. Indeed, as shown in Figure 5.3, the error displays a linear slope (due to the log graph) that is in agreement with the predicted error behavior, i.e., slopes of 3 and 5, respectively.

5.4 Discussions

This chapter presented the theoretical foundation of the proposed method, which extends the NILT approach—traditionally applied to linear circuits—to the simulation of nonlinear circuits. A key insight from this work is that the proposed method can

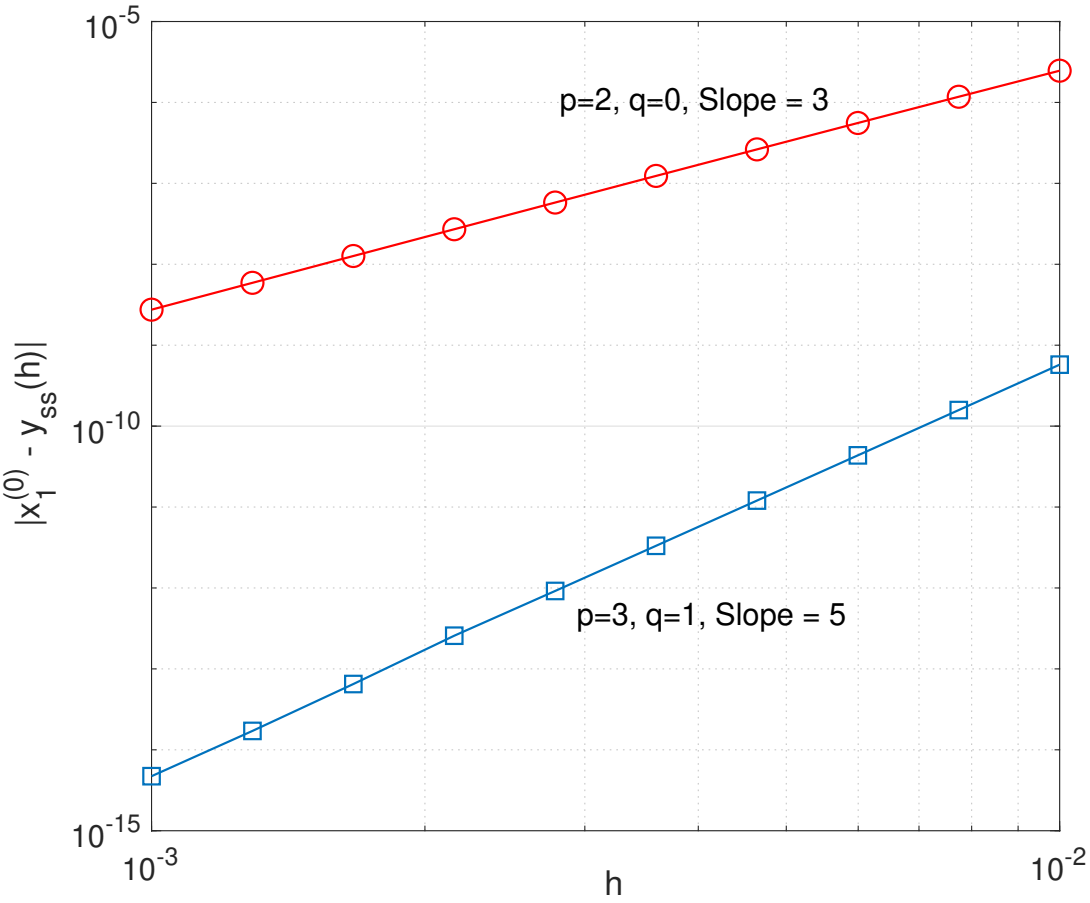


Figure 5.3: Asymptotic error behavior

inherit the essential characteristics of NILT, including its order of convergence and stability properties, under specific parameter settings. The critical parameters p and q determine the number of derivatives matched at the previous and current time points, respectively. Specifically, setting $q = N$ and $p = M$ enables the method to achieve the same high order of accuracy as the NILT method. This ability to extend NILT-like properties to nonlinear simulations represents a significant contribution of this thesis.

Chapter 6

NILTn For Nonlinear Circuit Simulation

Chapter 2 presented a concise introduction to an innovative approach called NILTn, developed to enhance the performance of the NILT method for time-domain simulation of linear circuits [34]. The method employs higher-order derivatives in *Laplace domain* that reduces the truncation error which in turn allows to increase time marching step size and therefore achieving improved accuracy and overall efficiency.

This chapter delves into the detailed work undertaken to expand the application of the NILTn to nonlinear circuits. The core idea follows the steps of the novel approach presented in Chapter 3, by first splitting the circuit and using the iterative NR method to solve for the voltages and their higher order derivatives at the ports of the nonlinear devices via enforcing the equality in (3.7). Section 6.1 presents in details utilization of NILTn in computing the currents and their higher-order derivatives of the auxiliary sources in the linear subcircuit, as well as the Jacobian matrix required during NR iter-

ations, then, Section 6.2 addresses the computational complexity involved in deploying NILTn methodology.

6.1 Utilization of NILTn for Nonlinear Circuit Simulation

The proposed approach in this chapter aims to utilize NILTn in computing the currents of the auxiliary sources flowing into the linear subcircuit, that is $\mathbf{i}(t)$ in Figure 3.2, and also their higher-order derivatives, at times $t = lh$, $l = 1, 2, \cdots$ scaled by powers of the step size h. Those scaled currents were denoted by $\hat{\mathbf{i}}_l^{(m)}$ in (3.7) when the conventional NILT methodology (hereinafter referred to as NILT0) was used in their computation. To distinguish the new NILTn-based current derivatives from the ones computed by (3.7) using NILT0, the notation is extended to denote the NILTn ones by $\hat{\mathbf{i}}_l^{[n,m]}$.

The following sections outline the development of the new approach in detail. First, the requirements for applying the NILTn method to approximate a general $\mathbf{y}(t)$ will be outlined in Section 6.1.1. This will form the basis for deriving the 0^{th}-order derivative approximation of the current $\mathbf{i}(t)$, at times $t = lh$, or what is denoted by $\hat{\mathbf{i}}_l^{[n,0]}$ that is described in Section 6.1.2. In Section 6.1.3, the development will demonstrate how the NILTn method approximates higher-order derivatives of the general function $\mathbf{y}(t)$. This will then guide, in Section 6.1.4, the approximation of higher-order derivatives of the current $\mathbf{i}(t)$ in the linear partition of the circuit, denoted by $\hat{\mathbf{i}}_l^{[n,m]}$, for $m > 0$.

To build intuition, we will initially illustrate the development steps with the simple scenarios of $n = 1, 2$ and then proceed to derive the general equations in Section 6.1.6.

6.1.1 Using NILTn in Evaluating Time-Domain Responses

In order to approximate a vector of waveforms $\mathbf{y}(t)$, given its Laplace-domain waveforms of $\mathbf{Y}(s)$, a new function $\mathbf{w}(t)$ by scaling the time t is defined as,

$$\mathbf{w}(t) = \mathbf{y}(\frac{t}{n+1}) \quad ; \quad n = 1, 2, 3 \ldots \tag{6.1}$$

Using the Laplace transform and its scaling property, we have

$$\mathbf{w}(t) \coloneqq \mathscr{L}^{-1}\{(n+1)\mathbf{Y}((n+1)s)\} = \frac{n+1}{2\pi\jmath}\int_{c-\jmath\infty}^{c+\jmath\infty}\mathbf{Y}((n+1)s)\mathrm{e}^{st}\mathrm{d}s \tag{6.2}$$

where $\jmath = \sqrt{-1}$. Considering that (6.1) can be written as $\mathbf{y}(t) = \mathbf{w}((n+1)t)$ the above integral will become,

$$\mathbf{y}(t) = \frac{n+1}{2\pi\jmath}\int_{c-\jmath\infty}^{c+\jmath\infty}\mathbf{Y}((n+1)s)\mathrm{e}^{(n+1)st}\mathrm{d}s \tag{6.3}$$

by taking $st = z$ and replacing e^z with its rational Padé approximation $\xi_{N,M}(z)$, i.e. $(\xi_{N,M}(z))^{n+1}$ for $\mathrm{e}^{(n+1)z}$, an approximation $\tilde{\mathbf{y}}(t)$ to $\mathbf{y}(t)$ is achieved by,

$$\tilde{\mathbf{y}}(t) = \frac{n+1}{2\pi\jmath t}\int_{c-\jmath\infty}^{c+\jmath\infty}\mathbf{Y}(\frac{(n+1)z}{t})(\xi_{N,M}(z))^{(n+1)}\mathrm{d}z \tag{6.4}$$

where $\xi_{N,M}(z)$ can be written in its partial fraction form using pole/residue pairs z_i, k_i,

$$\xi_{N,M}(z) = \sum_{i=1}^{M}\frac{k_i}{z - z_i} \tag{6.5}$$

Next, the multinomial expansion theorem,

$$(a_1 + a_2 + \cdots + a_m)^n = \sum_{k_1+k_2+\cdots+k_m=n} \frac{n!(a_1^{k_1} a_2^{k_2} \ldots a_m^{k_m})}{k_1! k_2! \ldots k_m!} \tag{6.6}$$

is used to expand $(\xi_{N,M}(z))^{n+1}$ which yields a new partial fraction representation, maintaining the same poles z_i as the original but with higher-order multiplicities.

Finally, applying the Cauchy theorem of residues transforms the line integral in (6.4) into a finite linear combination of $\mathbf{Y}(s)$ (and its derivatives w.r.t. s up to the n^{th}-order) computed at values of $s = \frac{(n+1)z_i}{t}$. That process yields the approximation sought for $\mathbf{y}(t)$ denoted by $\tilde{\mathbf{y}}^{[n]}(t)$. By setting $n = 1$ in (6.4) the foundation equation of NILT1 approximation yields as,

$$\tilde{\mathbf{y}}^{[1]}(t) = \frac{-8}{t} \sum_{i=1}^{M/2} \Re\left(\frac{k_i^2}{t} \frac{\mathrm{d}\mathbf{Y}(s)}{\mathrm{d}s} + k_i \mathbf{Y}(s) \sum_{\substack{j=1 \\ j\neq i}}^{M} \frac{k_j}{z_i - z_j} \right)\Bigg|_{s=\frac{2z_i}{t}} \tag{6.7}$$

By setting $n = 2$ in (6.4) and following the above steps, as mentioned for NILT1, the NILT2 in initial-value form of the approximation results as,

$$\begin{aligned} \tilde{\mathbf{y}}^{[2]}(t) \;=\; & \frac{-36}{t} \sum_{i=1}^{M/2} \Re \left\{ \frac{3k_i^3}{4t^2} \frac{\mathrm{d}^2\mathbf{Y}(s)}{\mathrm{d}s^2} + \frac{3k_i^2}{2t} \frac{\mathrm{d}\mathbf{Y}(s)}{\mathrm{d}s} \sum_{\substack{j=1 \\ j\neq i}}^{M} \frac{k_j}{(z_i - z_j)} \right. \\ & + \left. \mathbf{Y}(s) \left[k_i \sum_{\substack{j=1 \\ j\neq i}}^{M-1} \sum_{\substack{v=j+1 \\ v\neq i}}^{M} \frac{k_j k_v}{(z_i - z_j)(z_i - z_v)} + \frac{k_i}{2} \sum_{\substack{j=1 \\ j\neq i}}^{M} \frac{k_j^2}{(z_i - z_j)^2} - \frac{k_i^2}{2} \sum_{\substack{j=1 \\ j\neq i}}^{M} \frac{k_j}{(z_i - z_j)^2} \right] \right\} \Bigg|_{s=\frac{3z_i}{t}} \end{aligned} \tag{6.8}$$

If the goal was NILTn in re-initialized mode, assuming $t = lh$ via previous steps,

the resulting expression for NILT1 analogous to the one in (6.7) would become,

$$\tilde{\mathbf{y}}^{[1]}(lh) = -8\sum_{i=1}^{M/2}\Re\left(k_i^2\,\frac{\mathrm{d}\hat{\mathbf{Y}}(s)}{\mathrm{d}s} + k_i\hat{\mathbf{Y}}\,(s)\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j}{z_i - z_j}\right)\Bigg|_{s=2z_i} \tag{6.9}$$

It is worth highlighting here that the parameters N and M are selected such that $N = M - 2$. The value of M is determined based on n. Experimental observations through numerous simulations have shown that the best results for $n = 1, 2, 3$ are obtained with $M \leq 8$, $M \leq 6$, and $M \leq 4$, respectively. For $n \geq 4$, accuracy does not improve, as it requires computing fourth-order derivatives recursively from lower-order derivatives, leading to a loss of numerical accuracy [68].

6.1.2 Computing 0^{th}-order derivative of auxiliary source currents using NILT1

The application of NILTn for approximating $\mathbf{i}(t)$ necessitates the computation of the Laplace-domain $\mathbf{I}(s)$ and its derivatives with respect to s. For this purpose, we first review the process of computing $\mathbf{I}(s)$ that was explained in Chapter 3 and then proceed with computing its derivatives.

Given that $\mathbf{i}(t) = \mathscr{L}^{-1}\{\mathbf{I}(s)\}$, the current in the κ auxiliary voltage source, is incorporated as a variable within the MNA formulation of the linear partition, Figure 3.2, a selector matrix $\bar{\mathbf{d}}$ with κ columns can be employed. This selector matrix, containing only 0 and 1 entries, isolates the desired κ currents from the complete set of MNA

variables, enabling the derivation of an expression for $\mathbf{I}(s)$ given by,

$$\mathbf{I}(s) = \bar{\mathbf{d}}^{\top}\left(s\bar{\mathbf{C}} + \bar{\mathbf{G}}\right)^{-1} \begin{bmatrix} \bar{\mathbf{C}}\mathbf{x}((l-1)h) \\ \hat{\mathbf{U}}_{\text{aux}}(s) \end{bmatrix} \tag{6.10}$$

where,

- $\hat{\mathbf{U}}_{\text{aux}}(s)$ is the Laplace-domain representation of the auxiliary sources, $\mathbf{u}_{\text{aux},l}(\hat{t}\,)$, given in (4.17), with the Laplace transformation performed w.r.t. $\hat{t}$ resulting in,

$$\hat{\mathbf{U}}_{\text{aux}}(s) = \sum_{j=0}^{q} \theta_{l-1,j}(s)\ \hat{\mathbf{v}}_{l-1}^{(j)} + \sum_{j=0}^{p} \theta_{l,j}(s)\ \hat{\mathbf{v}}_{l}^{(j)} \tag{6.11}$$

- $\bar{\mathbf{G}}$ and $\bar{\mathbf{C}}$ are matrices with dimension $K + \kappa$ as shown in (3.13).

The derivation of Equation (6.10) involves: first, formulating the linear subcircuit with auxiliary sources using MNA approach in the time domain; second, performing the time variable substitution $t \to (l-1)h + \hat{t}h$; and third, applying the Laplace transform with respect to $\hat{t}$. This process aligns with the reinitialized mode of MNA approach described in (2.4), which incorporates auxiliary sources into the linear circuit partition representation

As (6.7) demonstrates, computing $\hat{\mathbf{i}}_l^{[n,0]}$, for $n = 1$, would require, not only computing the values of $\mathbf{I}(s)$ at $s = 2z_i$, but also its first-order derivative at the said values as well. Equation (6.11) shows that $\hat{\mathbf{U}}_{\text{aux}}(s)$ is available analytically, as a function of s, making finding its derivative w.r.t. s a straightforward matter. Furthermore, by noting that,

$$\frac{\mathrm{d}\left(\bar{\mathbf{G}} + s\bar{\mathbf{C}}\right)^{-1}}{\mathrm{d}s} = -\left(\bar{\mathbf{G}} + s\bar{\mathbf{C}}\right)^{-1} \bar{\mathbf{C}} \left(\bar{\mathbf{G}} + s\bar{\mathbf{C}}\right)^{-1}$$

the first-order derivative for $\mathbf{I}(s)$ can be written as

$$\frac{\mathrm{d}\mathbf{I}(s)}{\mathrm{d}s} = -\bar{\mathbf{d}}^{\top}\bar{\mathbf{A}}\bar{\mathbf{C}}\bar{\mathbf{A}}\begin{bmatrix}\bar{\mathbf{C}}\mathbf{x}((l-1)h)\\ \hat{\mathbf{U}}_{\mathrm{aux}}(s)\end{bmatrix} + \bar{\mathbf{d}}^{\top}\bar{\mathbf{A}}\begin{bmatrix}\mathbf{0}\\ \sum_{j=0}^{q}\frac{d\theta_{l-1,j}(s)}{ds}\hat{\mathbf{v}}_{l-1}^{(j)} + \sum_{j=0}^{p}\frac{d\theta_{l,j}(s)}{ds}\hat{\mathbf{v}}_{l}^{(j)}\end{bmatrix} \quad (6.12)$$

where $\bar{\mathbf{A}} = (\bar{\mathbf{G}} + s\bar{\mathbf{C}})^{-1}$.

Using (6.10) and (6.12) to substitute $\frac{\mathrm{d}\mathbf{I}(s)}{\mathrm{d}s}$ and $\mathbf{I}(s)$ for $\frac{\mathrm{d}\mathbf{Y}(s)}{\mathrm{d}s}$ and $\mathbf{Y}(s)$, respectively, in (6.7), provides the re-initialized mode of NILT1 version of the currents in the auxiliary sources,

$$\hat{\mathbf{i}}_l^{[1,0]} = 8\sum_{i=1}^{M/2}\Re\left\{\bar{\mathbf{d}}^{\top}\bar{\mathbf{A}}\left(k_i^2\left\{\bar{\mathbf{C}}\bar{\mathbf{A}}\begin{bmatrix}\bar{\mathbf{C}}\mathbf{x}_{l-1}\\ \hat{\mathbf{U}}_{\mathrm{aux}}(s)\end{bmatrix} - \begin{bmatrix}\mathbf{0}\\ \frac{\mathrm{d}\hat{\mathbf{U}}_{\mathrm{aux}}(s)}{\mathrm{d}s}\end{bmatrix}\right\} - k_i\begin{bmatrix}\bar{\mathbf{C}}\mathbf{x}_{l-1}\\ \hat{\mathbf{U}}_{\mathrm{aux}}(s)\end{bmatrix}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j}{z_i - z_j}\right)\right\}\Bigg|_{s=2z_i} \quad (6.13)$$

where $\mathbf{x}_{l-1} = \mathbf{x}((l-1)h)$.

6.1.3 Using NILTn in Evaluating High-Order Derivatives of Time-Domain Responses

In order to find an approximation of the m^{th}-order time-domain derivative of a given vector of waveforms $\mathbf{y}(t)$, using NILTn, the following identity of the Laplace transform

is used,

$$\mathscr{L}\left\{\frac{\mathrm{d}^m\mathbf{w}\left(t\right)}{\mathrm{d}t^m}\right\} = s^m\mathbf{W}(s) - \sum_{\vartheta=1}^{m-1}\mathbf{w}(0)^{(\vartheta-1)}\,s^{m-\vartheta} \tag{6.14}$$

with $\mathbf{w}(t)$ being defined as before in (6.2) where $\mathbf{w}(0)^{(\vartheta-1)}$ is the $(\vartheta-1)^{th}$ derivative of $\mathbf{w}(t)$ at $t=0$. Using the above identity, while noting the relation between $\mathbf{w}(t)$ and $\mathbf{y}(t)$ in (6.1), enables writing m^{th}-order derivative of $\mathbf{y}(t)$ as,

$$\begin{aligned}\mathbf{y}(t)^{(m)} &= \mathscr{L}^{-1}\left\{s^m\mathbf{W}(s) - \sum_{\vartheta=1}^{m-1}\mathbf{w}(0)^{(\vartheta-1)}\,s^{m-\vartheta}\right\}((n+1)t)\\ &= \underbrace{\frac{n+1}{2\pi\jmath}\int_{c-\jmath\infty}^{c+\jmath\infty} s^m\mathbf{Y}((n+1)s)\mathrm{e}^{(n+1)st}\mathrm{d}s}_{\mathbf{\Theta}_1(t)} \underbrace{-\frac{1}{2\pi\jmath}\int_{c-\jmath\infty}^{c+\jmath\infty}\sum_{\vartheta=1}^{m-1}\mathbf{y}(0)^{(\vartheta-1)}\,s^{m-\vartheta}\mathrm{e}^{(n+1)st}\mathrm{d}s}_{\mathbf{\Theta}_2(t)}\end{aligned} \tag{6.15}$$

The same principles and steps that were used in Section 6.1.1 to transform the integral in (6.4) into the summation form of (6.7) can be invoked again to transform the line integrals in the expressions for $\mathbf{\Theta}_1(t)$ and $\mathbf{\Theta}_2(t)$ in (6.15) into analogous forms that approximates the m^{th}-order derivatives of $\mathbf{y}(t)$. The resulting approximation is hereby denoted by $\tilde{\mathbf{y}}^{[n,m]}$. For the special case of $n=1$, one obtains the initial-value version of those terms as,

$$\tilde{\mathbf{y}}^{[1,m]}(t) = \tilde{\mathbf{\Theta}}_1^{[1,m]}(t) + \tilde{\mathbf{\Theta}}_2^{[1,m]}(t) \tag{6.16}$$

where,

$$\tilde{\mathbf{\Theta}}_1^{[1,m]}(t) = \frac{-8}{t}\sum_{i=1}^{M/2}\Re\left\{\frac{k_i^2}{t}\left[ms^{m-1}\mathbf{Y}(s) + s^m\frac{\mathrm{d}\mathbf{Y}(s)}{\mathrm{d}s}\right] + k_i s^m\mathbf{Y}\left(s\right)\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j}{z_i-z_j}\right\}\Bigg|_{s=\frac{2z_i}{t}} \tag{6.17}$$

$$\tilde{\mathbf{\Theta}}_2^{[1,m]}(t) = \frac{-4}{t}\sum_{\vartheta=1}^{m-1}\mathbf{y}(0)^{(\vartheta-1)}\sum_{i=1}^{M/2}\Re\left\{(m-\vartheta)\frac{k_i^2}{2}s^{m-\vartheta-1} + k_i s^{m-\vartheta}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j}{z_i - z_j}\right\}\Bigg|_{s=\frac{2z_i}{t}} \tag{6.18}$$

The re-initialized version can also be derived for both $\mathbf{\Theta}_1(t)$ and $\mathbf{\Theta}_2(t)$ at $t = lh$. These re-initialized terms are denoted as $\hat{\mathbf{\Theta}}_{1,l}^{[1,m]}$ and $\hat{\mathbf{\Theta}}_{2,l}^{[1,m]}$, following the conventions used in denoting re-initialized NILTn-based terms. The expressions for $\hat{\mathbf{\Theta}}_{1,l}^{[1,m]}$ and $\hat{\mathbf{\Theta}}_{2,l}^{[1,m]}$ are structurally identical to $\tilde{\mathbf{\Theta}}_1^{[1,m]}(t)$ and $\tilde{\mathbf{\Theta}}_2^{[1,m]}(t)$, as given in (6.17) and (6.18), respectively, except for the following modifications:

1. Replace t with a unity, i.e, "1" value.
2. Substitute the Laplace variable s with $s = 2z_i$ instead of $s = 2z_i/t$.
3. Replace $\mathbf{y}(0)^{(\vartheta-1)}$ in (6.18) with $\mathbf{y}((l-1)h)^{(\vartheta-1)}$, which represents the value of $\frac{\mathrm{d}^{\vartheta-1}\mathbf{y}(t)}{\mathrm{d}t^{\vartheta-1}}$ at $t = (l-1)h$.

6.1.4 Computing m^{th}-order derivative of auxiliary source currents using NILTn

To obtain the NILTn approximation $\hat{\dot{\mathbf{i}}}_l^{[n,m]}$ of the m^{th}-order derivative of $\mathbf{i}(t)$ at $t = lh$, necessary for the NR iteration, we utilize the process outlined in Section 6.1.3. For the special case of $n = 1$, the sequence used to arrive at expression (6.19) for $\hat{\dot{\mathbf{i}}}_l^{[1,m]}$ are the following steps,

1. Replacing $\mathbf{Y}(s)$ in (6.17) with $\mathbf{I}(s)$ from (6.10),
2. Substituting of $\frac{\mathrm{d}\mathbf{I}(s)}{\mathrm{d}s}$ from (6.12) in place of $\frac{\mathrm{d}\mathbf{Y}(s)}{\mathrm{d}s}$ in (6.17),

3. Exchanging $\mathbf{y}(0)^{(\vartheta-1)}$ in (6.18) for $\hat{\mathbf{i}}_{l-1}^{[1,j-1]}$,

4. Replacing t with a unity "1" value and substituting the Laplace variable s with $s = 2z_i$ instead of $s = \frac{2z_i}{t}$, in both of (6.17) and (6.18), and, finally,

5. Adding the expressions obtained from all of the previous manipulations.

$$\hat{\mathbf{i}}_l^{[1,m]} = -2\sum_{i=1}^{M/2} \Re\left\{ 2s^m \bar{\mathbf{d}}^\top \bar{\mathbf{A}} \left(k_i^2(-\bar{\mathbf{C}}\bar{\mathbf{A}} + \frac{m}{s}) \begin{bmatrix} \bar{\mathbf{C}}\mathbf{x}_{l-1} \\ \hat{\mathbf{U}}_{\text{aux}}(s) \end{bmatrix} + k_i^2 \begin{bmatrix} \mathbf{0} \\ \frac{\mathrm{d}\hat{\mathbf{U}}_{\text{aux}}(s)}{\mathrm{d}s} \end{bmatrix} \right.\right.$$
$$\left.\left. + k_i \begin{bmatrix} \bar{\mathbf{C}}\mathbf{x}_{l-1} \\ \hat{\mathbf{U}}_{\text{aux}}(s) \end{bmatrix} \sum_{\substack{j=1 \\ j\neq i}}^{M} \frac{k_j}{z_i - z_j} \right) - \sum_{\vartheta=1}^{m-1} k_i s^{m-\vartheta} \left(\frac{k_i(m-\vartheta)}{s} \hat{\mathbf{i}}_{l-1}^{[1,j-1]} + \sum_{\substack{j=1 \\ j\neq i}}^{M} \frac{k_j}{z_i - z_j} \right) \right\} \Bigg|_{s=2z_i} \tag{6.19}$$

For the case of $n = 2$, NILT2 approximation in (6.8), necessitates finding the second order derivative of the current $\frac{\mathrm{d}^2\mathbf{I}(s)}{\mathrm{d}s^2}$. We achieve this by repeatedly differentiating of (2.4) with respect to s,

$$(\mathbf{G} + s\mathbf{C})\mathbf{X}(s)^{(1)} = \frac{\mathrm{d}\mathbf{B}(s)}{\mathrm{d}s} - \mathbf{C}\mathbf{X}(s)^{(0)} \tag{6.20}$$

$$(\mathbf{G} + s\mathbf{C})\mathbf{X}(s)^{(2)} = \frac{\mathrm{d}^2\mathbf{B}(s)}{\mathrm{d}s^2} - 2\mathbf{C}\mathbf{X}(s)^{(1)} \tag{6.21}$$

Then, as explained for deriving (6.12), the second order derivative of the current can be written as,

$$\mathbf{I}(s)^{(2)} = \bar{\mathbf{d}}^\top \bar{\mathbf{A}} \left(2\bar{\mathbf{C}}\bar{\mathbf{A}}\bar{\mathbf{C}}\bar{\mathbf{A}} \begin{bmatrix} \frac{1}{h}\bar{\mathbf{C}}\tilde{\mathbf{x}}_{l-1} \\ \hat{\mathbf{U}}_{\text{aux}}(s) \end{bmatrix} \right.$$

$$-2\bar{\mathbf{C}}\bar{\mathbf{A}}\left[\begin{array}{c}\frac{1}{h}\bar{\mathbf{C}}\tilde{\mathbf{X}}_{l-1}^{(1)}\\ \sum_{j=0}^{q}\frac{d\theta_{l-1,j}(s)}{ds}\hat{\mathbf{v}}_{l-1}^{(j)}+\sum_{j=0}^{p}\frac{d\theta_{l,j}(s)}{ds}\hat{\mathbf{v}}_{l}^{(j)}\end{array}\right]+\left[\begin{array}{c}\frac{1}{h}\bar{\mathbf{C}}\tilde{\mathbf{X}}_{l-1}^{(2)}\\ \sum_{j=0}^{q}\frac{d^2\theta_{l-1,j}(s)}{ds^2}\hat{\mathbf{v}}_{l-1}^{(j)}+\sum_{j=0}^{p}\frac{d^2\theta_{l,j}(s)}{ds^2}\hat{\mathbf{v}}_{l}^{(j)}\end{array}\right]\Bigg) \tag{6.22}$$

As for the above 4^{th} step, the Laplace variable is replaced by $s = 3z_i$. Finally, NILT2 approximation of the auxiliary source current is achieved as,

$$\hat{\mathbf{i}}^{[2,m]}(t) = \hat{\mathbf{\Theta}}_1^{[2,m]}(t) + \hat{\mathbf{\Theta}}_2^{[2,m]}(t) \tag{6.23}$$

where,

$$\hat{\mathbf{\Theta}}_1^{[2,m]}(t) = -36\sum_{i=1}^{M/2}\Re\left(\frac{3k_i^3}{4}(s^m\mathbf{I}(s))^{(2)}+\frac{3k_i^2}{2}(s^m\mathbf{I}(\mathbf{s}))^{(1)}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j}{(z_i-z_j)}\right.$$
$$\left.+s^m\mathbf{I}(s)\left(k_i\sum_{\substack{j=1\\ j\neq i}}^{M-1}\sum_{\substack{\vartheta=j+1\\ \vartheta\neq i}}^{M}\frac{k_jk_v}{(z_i-z_j)(z_i-z_v)}+\frac{k_i}{2}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j^2}{(z_i-z_j)^2}-\frac{k_i^2}{2}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j}{(z_i-z_j)^2}\right)\right)\Bigg|_{s=3z_i} \tag{6.24}$$

$$\hat{\mathbf{\Theta}}_2^{[2,m]}(t) = 12\sum_{\vartheta=1}^{m-1}i(0)^{(\vartheta-1)}\sum_{i=1}^{M/2}\Re\left(\left(\frac{3k_i^3}{4}(s^{m-\vartheta})^{(2)}+\frac{3k_i^2}{2}(s^{m-\vartheta})^{(1)}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j}{(z_i-z_j)}\right.\right.$$
$$\left.\left.+s^{m-\vartheta}\left\{k_i\sum_{\substack{j=1\\ j\neq i}}^{M-1}\sum_{\substack{o=j+1\\ o\neq i}}^{M}\frac{k_jk_o}{(z_i-z_j)(z_i-z_o)}+\frac{k_i}{2}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j^2}{(z_i-z_j)^2}-\frac{k_i^2}{2}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j}{(z_i-z_j)^2}\right\}\right)\right)\Bigg|_{s=3z_i} \tag{6.25}$$

6.1.5 Computing the Jacobian Matrix

The final important part is the Jacobian matrix needed to run the NR iteration and is given by $(\mathbf{J}_{\text{lin}} + \mathbf{J}_{\text{nonlin}})$, where for NILT1 as an example, defined as,

$$\mathbf{J}_{\text{lin}} = \frac{\partial \hat{\mathbf{i}}_l^{[1,\rho_1]}}{\partial \hat{\mathbf{v}}_l^{(\rho_2)}}, \quad \mathbf{J}_{\text{nonlin}} = \frac{\partial \hat{\mathbf{j}}_l^{(\rho_1)}}{\partial \hat{\mathbf{v}}_l^{(\rho_2)}} \tag{6.26}$$

with ρ_1, ρ_2 being indices assigned $0, 1, \cdots, p$. The structure of this matrix was shown in Section 4.3.3, in which $\mathbf{J}_{\text{nonlin}}$ was derived based on the notion of RT. That derivation will be used here without change. On the other hand, a new expression for $\mathbf{J}_{\text{lin}}$ is required to take into account the NILTn methodology used for the currents from the linear part.

For the case of $n = 1$, that expression can be easily found by noting from (6.19) that $\hat{\mathbf{v}}_l^{(\rho_2)}$ influences $\hat{\mathbf{i}}_l^{[1,\rho_1]}$ through its role in the construction of the auxiliary sources $\hat{\mathbf{U}}_{\text{aux}}(s)$ expressed by (6.11). These observations, along with knowing derivative of inverse of a matrix $\mathbf{N}$ is defined as $-\mathbf{N^{-1}}\left(\frac{\mathrm{d}\mathbf{N}}{\mathrm{d}t}\right)\mathbf{N^{-1}}$, yield $\mathbf{J}_{\text{lin}}$ for the NILT1 as given by,

$$\mathbf{J}_{\text{lin}}^{[1]} = -4\sum_{i=1}^{M/2} \Re\left(k_i s^{\rho_1} \bar{\mathbf{d}}^\top \bar{\mathbf{A}} \left(k_i(-\bar{\mathbf{C}}\bar{\mathbf{A}} + \frac{\rho_1}{s}) \begin{bmatrix} \mathbf{0} \\ \theta_{l,\rho_2}(s)\mathbb{I}_\kappa \end{bmatrix} + k_i \begin{bmatrix} \mathbf{0} \\ \frac{d\theta_{l,\rho_2}(s)}{ds}\mathbb{I}_\kappa \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \theta_{l,\rho_2}(s)\mathbb{I}_\kappa \end{bmatrix} \sum_{\substack{j=1 \\ j\neq i}}^{M} \frac{k_j}{(z_i - z_j)} \right) \right) \Bigg|_{s=2z_i} \tag{6.27}$$

where $\mathbb{I}_\kappa$ denotes an identity matrix of size κ.

The above steps can be applied for the case of $n = 2$ to compute the linear part of

the Jacobian for NILT2 to yield,

$$\begin{aligned}\mathbf{J}_{\text{lin}}^{[2]} = -36\sum_{i=1}^{M/2}\Re\Bigg\{\bar{\mathbf{d}}^{\top}\bar{\mathbf{A}}\Bigg(\frac{3k_i^3}{4}\Bigg(&\rho_1(\rho_1-1)s^{\rho_1-2}\begin{bmatrix}\mathbf{0}\\ \theta_{l,\rho_2}(s)\mathbb{I}_\kappa\end{bmatrix}\\ &+2\rho_1 s^{\rho_1-1}\Bigg\{-\bar{\mathbf{C}}\bar{\mathbf{A}}\begin{bmatrix}\mathbf{0}\\ \theta_{l,\rho_2}(s)\mathbb{I}_\kappa\end{bmatrix}+\begin{bmatrix}\mathbf{0}\\ \frac{\mathrm{d}\theta_{l,\rho_2}(s)}{\mathrm{d}s}\mathbb{I}_\kappa\end{bmatrix}\Bigg\}\\ &+s^{\rho_1}\Bigg\{2\bar{\mathbf{C}}\bar{\mathbf{A}}\bar{\mathbf{C}}\bar{\mathbf{A}}\begin{bmatrix}\mathbf{0}\\ \theta_{l,\rho_2}(s)\mathbb{I}_\kappa\end{bmatrix}-2\bar{\mathbf{C}}\bar{\mathbf{A}}\begin{bmatrix}\mathbf{0}\\ \frac{\mathrm{d}\theta_{l,\rho_2}(s)}{\mathrm{d}s}\mathbb{I}_\kappa\end{bmatrix}+\begin{bmatrix}\mathbf{0}\\ \frac{\mathrm{d}^2\theta_{l,\rho_2}(s)}{\mathrm{d}s^2}\mathbb{I}_\kappa\end{bmatrix}\Bigg\}\Bigg)\\ &+\frac{3k_i^2}{2}\Bigg\{\rho_1 s^{\rho_1-1}\begin{bmatrix}\mathbf{0}\\ \theta_{l,\rho_2}(s)\mathbb{I}_\kappa\end{bmatrix}+s^{\rho_1}\Bigg(-\bar{\mathbf{C}}\bar{\mathbf{A}}\begin{bmatrix}\mathbf{0}\\ \theta_{l,\rho_2}\mathbb{I}_\kappa\end{bmatrix}+\begin{bmatrix}\mathbf{0}\\ \frac{\mathrm{d}\theta_{l,\rho_2}(s)}{\mathrm{d}s}\mathbb{I}_\kappa\end{bmatrix}\Bigg)\Bigg\}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j}{(z_i-z_j)}\\ &+s^{\rho_1}\begin{bmatrix}\mathbf{0}\\ \theta_{l,\rho_2}(s)\mathbb{I}_\kappa\end{bmatrix}\Bigg\{k_i\sum_{\substack{j=1\\ j\neq i}}^{M-1}\sum_{\substack{\vartheta=j+1\\ \vartheta\neq i}}^{M}\frac{k_jk_v}{(z_i-z_j)(z_i-z_v)}+\frac{k_i}{2!}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j^2}{(z_i-z_j)^2}-\frac{k_i^2}{2!}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{k_j}{(z_i-z_j)^2}\Bigg\}\Bigg)\Bigg\}\Bigg|_{s=3z_i}\end{aligned} \tag{6.28}$$

6.1.6 General form of auxiliary source currents using NILTn

To more effectively describe the general expressions for a given n, an expanded set of notations is necessary. These expanded notations can be summarized as follows:

- Define $\mathcal{P} = n + 1$.
- Employ the notion of the *multi-index* which is defined as a vector in $\mathbb{N}^M$, i.e., a vector whose M components are non-negative integers. This notation uses a bold font lowercase letter, e.g. $\mathbf{l}$, to denote a given multi-index, while its individual components will be denoted by a sub-scripted italic (non-bold) lowercase of the same letter, i.e. $l_i, i = 1, 2, \ldots, M$, where $l_i \in \mathbb{N}$,

- A set of multi-indices is a set defined by

$$\Lambda_{\mathcal{P}} := \left\{ \mathbf{l} : \sum_{i=1}^{M} l_i = \mathcal{P} \right\}$$

- The *factorial* of a multi-index will be defined as the product of the factorials of its components, i.e.,

$$\mathbf{l}! := \prod_{i=1}^{M} l_i!$$

- The *power* of a multi-index, $\mathbf{l}$, for a given vector $\boldsymbol{d} \in \mathbb{R}^M$ is defined as the operation

$$\boldsymbol{d}^{\mathbf{l}} := \prod_{i=1}^{M} d_i^{l_i}$$

Using the above notations, multinomial theorem (6.6), applying the generalized Leibniz formula [69] to describe the μ^{th} derivative of the term $s^m \mathbf{Y}((n+1)s)$ in (6.15), and through a sequence of steps similar to the steps used in sub-sections 6.1.1 to 6.1.4, the general expression of the NILTn-based auxiliary source currents (and their time-domain derivatives) are obtained in (6.32). Furthermore, the corresponding Jacobian term $\mathbf{J}_{\text{lin}}^{[n]}$ is provided in (6.35).

The terms used in describing (6.32) and (6.35) are defined as follows. $\mathscr{W}_{i,\mu}$ is obtained from

$$\mathscr{W}_{i,\mu} = \sum_{\boldsymbol{\vartheta} \in \Lambda_{l_i-1-\mu}^{M-1}} \frac{(l_i - 1 - \mu)!}{\boldsymbol{\vartheta}!} \prod_{\substack{j=1 \\ j \neq i}}^{M} \frac{(-1)^{\vartheta_j} \prod_{\theta=0}^{\vartheta_j - 1} (l_j + \theta)}{(z_i - z_j)^{l_j + \vartheta_j}} \tag{6.29}$$

The n^{th}-order derivative of the matrix $\bar{\mathbf{A}} = \left(\bar{\mathbf{G}} + s\bar{\mathbf{C}}\right)^{-1}$ is denoted as $\bar{\mathbf{A}}^{(n)}$ and is

given by

$$\bar{\mathbf{A}}^{(n)} = (-1)^n n! \overbrace{\bar{\mathbf{A}}\bar{\mathbf{C}}\bar{\mathbf{A}}\bar{\mathbf{C}}\dots\bar{\mathbf{A}}\bar{\mathbf{C}}}^{n \text{ times}}\bar{\mathbf{A}} \tag{6.30}$$

The n^{th}-order derivatives of the auxiliary current sources are given by

$$\mathbf{I}(s)^{(n)} = \bar{\mathbf{d}}^\top \sum_{\gamma=0}^{n} \binom{n}{\gamma} \bar{\mathbf{A}}^{(n-\gamma)} \begin{bmatrix} \frac{1}{h}\bar{\mathbf{C}}\tilde{\mathbf{X}}_{l-1}^{(\gamma)} \\ \sum\limits_{j=0}^{q} \theta_{l-1,j}^{(\gamma)}(s)\hat{\mathbf{v}}_{l-1}^{(j)} + \sum\limits_{j=0}^{p} \theta_{l,j}^{(\gamma)}(s)\hat{\mathbf{v}}_{l}^{(j)} \end{bmatrix} \tag{6.31}$$

and $\mathbf{k} = [k_1\, k_2\, \dots\, k_M]^\top$.

$$\hat{\mathbf{i}}^{[n,m]}(t) = \hat{\mathbf{\Theta}}_1^{[n,m]}(t) + \hat{\mathbf{\Theta}}_2^{[n,m]}(t) \tag{6.32}$$

$$\hat{\mathbf{\Theta}}_1^{[n,m]}(t) = -\mathcal{P}\mathcal{P}! \sum_{\mathbf{l}\in\Lambda_{\mathcal{P}}^M} \sum_{\substack{i=1 \\ l_i\neq 0}}^{M} \frac{\mathbf{k}^{\mathbf{l}}}{\mathbf{l}!} \left[\frac{1}{(l_i-1)!} \sum_{\mu=0}^{l_i-1} \binom{l_i-1}{\mu} \mathcal{P}^\mu \left(\sum_{r=0}^{min(m,\mu)} \binom{\mu}{r} \frac{m!}{(m-r)!} (\mathcal{P}z_i)^{m-r} \mathbf{I}(\mathcal{P}z_i)^{(\mu-r)} \right) \mathscr{W}_{i,\mu} \right] \tag{6.33}$$

$$\hat{\mathbf{\Theta}}_2^{[n,m]}(t) = \mathcal{P}\mathcal{P}! \sum_{\vartheta=1}^{m-1} i(0)^{(\vartheta-1)} \sum_{\mathbf{l}\in\Lambda_{\mathcal{P}}^M} \sum_{\substack{i=1 \\ l_i\neq 0}}^{M} \frac{\mathbf{k}^{\mathbf{l}}}{\mathbf{l}!} \left[\frac{1}{(l_i-1)!} \sum_{\mu=0}^{l_i-1} \binom{l_i-1}{\mu} \left(\frac{(m-\vartheta)!}{(m-\vartheta-\mu)!} (\mathcal{P}z_i)^{m-\vartheta-\mu} \right) \mathscr{W}_{i,\mu} \right] \tag{6.34}$$

$$\mathbf{J}_{\text{lin}}^{[n]} = -\mathcal{P}\mathcal{P}! \sum_{\mathbf{l}\in\Lambda_{\mathcal{P}}^M} \sum_{\substack{i=1 \\ l_i\neq 0}}^{M} \frac{\mathbf{k}^{\mathbf{l}}}{\mathbf{l}!} \left[\frac{1}{(l_i-1)!} \sum_{\mu=0}^{l_i-1} \binom{l_i-1}{\mu} \mathcal{P}^\mu \left(\sum_{r=0}^{min(\rho_1,\mu)} \binom{\mu}{r} \frac{\rho_1! s^{\rho_1-r}}{(\rho_1-r)!} \bar{\mathbf{d}}^\top \sum_{\gamma=0}^{\mu-r} \binom{\mu-r}{\gamma} \bar{\mathbf{A}}^{(\mu-r-\gamma)} \begin{bmatrix} \bar{\mathbf{C}}\tilde{\mathbf{x}}_{l-1}^{(\gamma)} \\ \theta_{l,\rho_2}^{(\gamma)}(s)\mathbb{I}_\kappa \end{bmatrix} \right) \mathscr{W}_{i,\mu} \right] \Bigg|_{s=\frac{\mathcal{P}z_i}{t}} \tag{6.35}$$

6.2 Computational Complexity

The computational complexity of the proposed approach is assessed by considering the computational effort at each time-point and the total computational burden across the entire simulation.

At the level of the overall simulation, the proposed approach is advantaged by incorporating the high-order, stability-guaranteed, NILT-based time-stepping engine. NILT, has been known before [18] to enable an order of approximation equal to $p+q+2$ that is numerically and provably stable provided that $p-2 \leq q \leq p$, where q and p are the order of the derivatives matched at two consecutive time points. This feature needs to be contrasted with the conventional time-stepping techniques, such as TR, where numerical stability is only guaranteed for orders equal to, or less than 2 [15]. The higher order enables the step size h to be made larger in NILT than with the conventional solvers.

The main computational efforts in the proposed approach, as evidenced by (6.13)-(6.35), is the computation of the matrix $\bar{\mathbf{A}}$ which is equivalent to the inverse of $\left(\bar{\mathbf{G}}+s\bar{\mathbf{C}}\right)$ for values of $s=z_i/t$, in the initial-value mode, or $s=z_i$ in the re-initialized mode. This fact can be utilized to make the simulation more efficient through executing the algorithm in the re-initialized mode, by Lower–Upper (LU) factorizing the matrix $\left(\bar{\mathbf{G}}+z_i\bar{\mathbf{C}}\right)^{-1}$ for $i=1,\cdots,M/2$ at the beginning of the simulation, storing the $\mathbf{L}$ and $\mathbf{U}$ factors and then using the process of Forward/Backward substitution to multiply with the appropriate vectors at each time step.

At each time step, the proposed approach runs via iterating in a NR framework that requires computing and factorizing a Jacobian matrix of size $\kappa(p+1)$, where, typically, p is chosen from $0 \leq p \leq 2$, and κ is the number of nonlinear devices. This

fact makes the proposed approach significantly advantageous for applications where the system is comprised of large linear circuits, such as those arising from full-wave modeling of cables and Printed Circuit Board (PCB)s, and few nonlinear driver and buffer circuitry.

The proposed NILTn method in this chapter offers an additional improvement over the conventional NILT (NILT0) by reducing its approximation error. Specifically, when both NILTn and NILT0 are used with the same integer values of N and M, the approximation error for NILTn is $\frac{1}{(n+1)^{(M+N)}}$ of that for NILT0. This result was proved in [34], whereas the estimation of the approximation error was developed in [70]. This characteristic enables NILTn to employ larger time steps than NILT0, or equivalently achieve comparable approximation errors with a smaller value of M. Consequently, NILTn requires fewer matrix factorizations than the NILT0 [32].

The numerical results in the next chapter will provide a clear demonstration of this benefit.

Chapter 7

Numerical Simulation and Results

The results of the research conducted for this thesis are presented in this chapter, utilizing simulations of various example circuits to illustrate the accuracy, efficiency and achieved speedup of the proposed methods.

The proposed methods are implemented and executed in MATLAB® 9.13.0 (R2022b) environment. The construction and implementation of the RT, for the purpose of computing $\hat{\mathbf{j}}_l^{(m)}$ and $\mathbf{J}_{\text{nonlin}}$, is carried out using an OOP framework in the C++ language interfaced to the main MATLAB® implementation.

A range of nonlinear circuit components, including diode, BJT and MOSFET are selected for simulation to validate the efficiency of the proposed approach. Additionally, some examples incorporated these components along with Transmission Line (TL)s. This allowed to thoroughly demonstrate the accuracy, stability, and overall performance of this thesis's contributions in the time-domain simulation of highly complex circuits, particularly those exhibiting both nonlinearity and distributed frequency-domain parameters.

In the followings, first, Section 7.1 demonstrates the results of deploying proposed NILT0 approach for time-domain simulation of nonlinear circuits. Then, Section 7.2 showcases the efficiency, accuracy, and the performance gained from using the NILTn method in nonlinear circuit simulation. In all numerical simulations, results that are used as the reference for comparison were obtained by executing the TR method with sufficiently small step size.

7.1 Time-Domain Simulation of Nonlinear Circuits Using NILT0

This section provides three examples to show the performance of the proposed method NILT0 and the developed framework to simulate large circuits including nonlinear elements.

7.1.1 Comparison with Existing Splitting Methods

This experiment demonstrates the key distinction between the splitting method in our proposed approach and existing circuit partitioning techniques. Specifically, we compare our method to a splitting technique based on the node tearing method of [71]. This comparison is significant because the node tearing method is equivalent to setting $p = 1$ in our approach, effectively eliminating the matching of high-order current derivatives at nonlinear device ports. Figure 7.1 presents the results obtained using this $p = 1$ setting, with 13 steps (same step-size as in Figure 7.4), and compares it to the TR with a small time step. As Figure 7.1 clearly illustrates, conventional splitting

based on node tearing, when used with the larger step size of our proposed approach, results in a significant loss of accuracy.

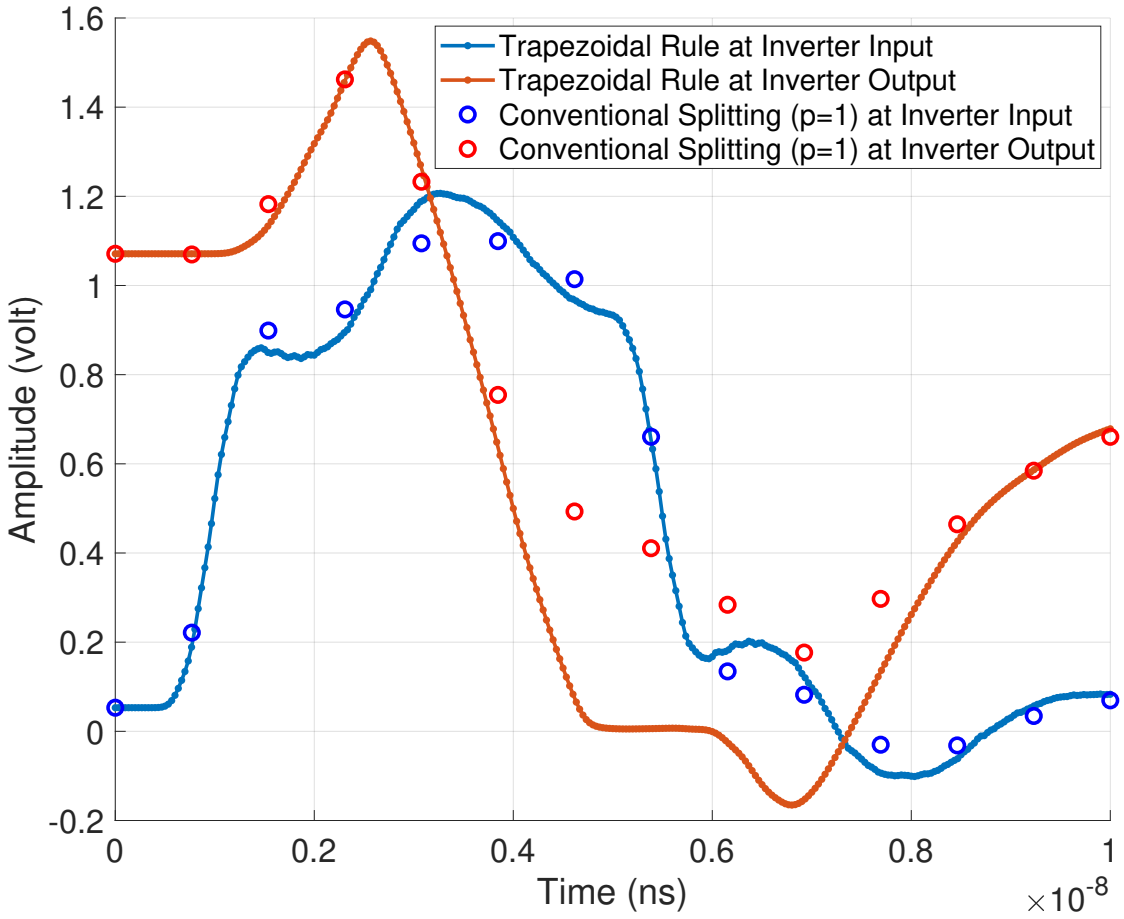


Figure 7.1: Waveforms of BJT inverter circuit using $p = 1$

7.1.2 A BJT Circuit with 3 Transmission Lines

Figure 7.2 shows the schematic of the circuit used in the examples. This circuit contains three TL segments consisting of 4 coupled conductors each. The physical structure of

the TL is depicted in the lower left of the figure along with the its parameters. The TL model used in this experiment is based on the lumped segmentation [19]. Each TL segment is modeled using 100 sections of lumped RLGC elements. The circuit is excited by a trapezoidal voltage source that is attached to the near end of the first line of the first transmission line segment.

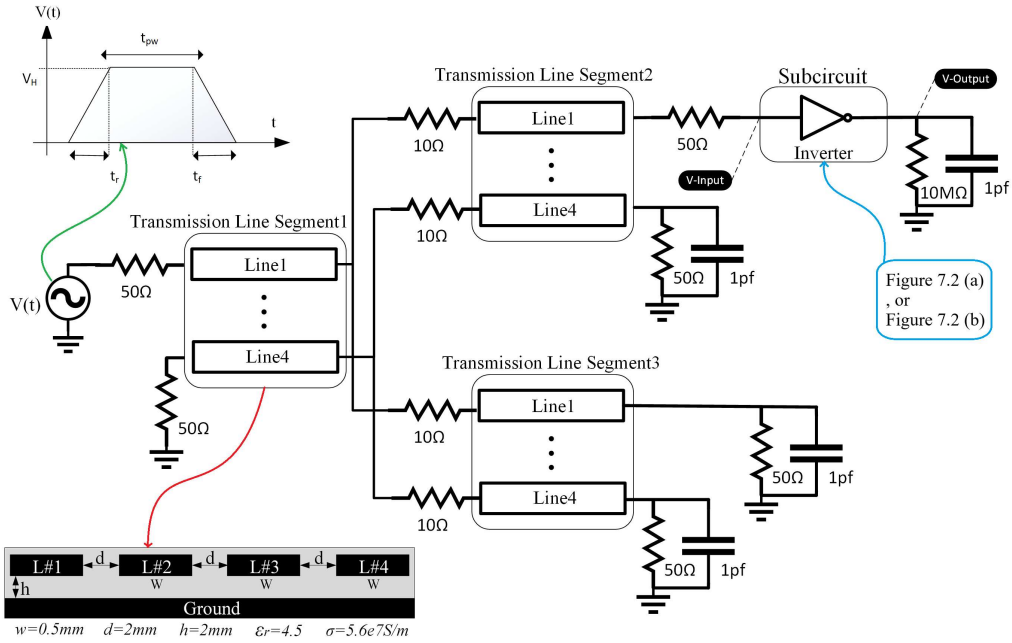


Figure 7.2: Test Circuit Diagram

In this example, the terminating inverter in subcircuit of Figure 7.2 was implemented using BJT inverter shown in Figure 7.3 (a). The BJT model is based on Ebers-Moll Model including all PN junction capacitances [72], [18]. The size of the MNA formulation is 3652. The circuit is excited using a 1.8 v trapezoidal pulse with 0.1 ns rise/fall times and pulse width of 4 ns. The time-domain response at both outputs of transmission line and inverter are obtained by NILT0 using $M = 4$, $N = 2$,

$p = 2$ and $q = 2$ and displayed in Figure 7.4.

This figure compares the waveforms obtained from NILT0 (circles) against a SPICE-like solver implemented using the Trapezoidal Rule method (dots), which clearly shows that the proposed NILT0 method could efficiently and accurately follow the TR results with much less number of time-steps.

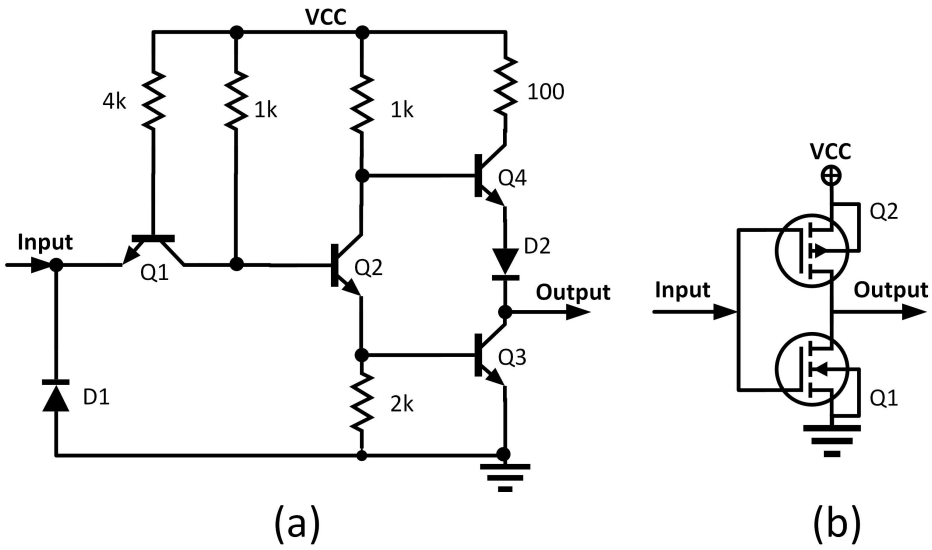


Figure 7.3: BJT and CMOS Inverters

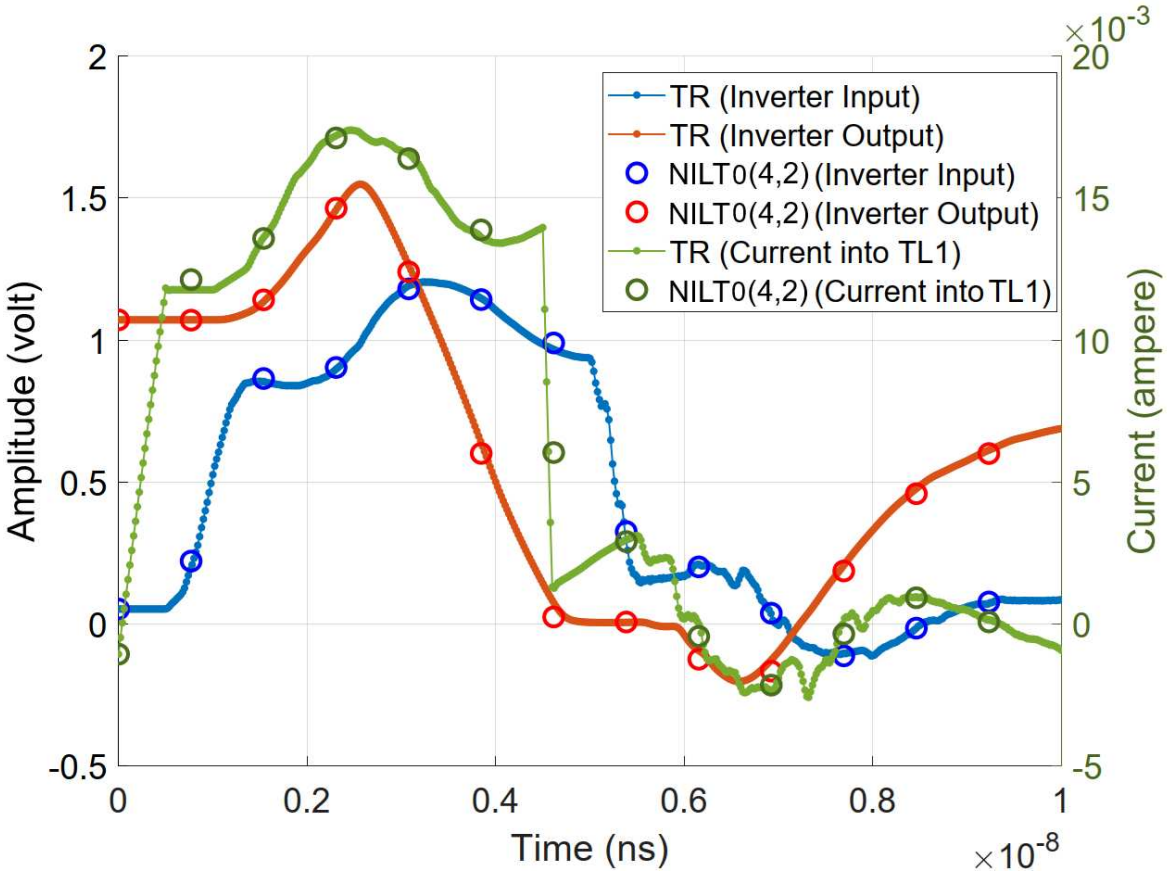


Figure 7.4: Waveforms obtained from BJT inverter circuit.

7.1.3 A CMOS Inverter with 3 Transmission Lines

In this example, the inverter in subcircuit of Figure 7.2 is implemented using a CMOS circuit shown in Figure 7.3(b). The MOSFETs' model are taken from [73]. [1] The size of the MNA formulation in this example is 3623. The circuit is excited with a 1.8 v trapezoidal pulse with 0.1 ns rise/fall times and pulse width of 5 ns. The time-domain

[1] C_∞-Continues is a compact model whose mathematical equations and all their derivatives are continuous to an infinite order across all regions of operation.

response at both outputs of transmission line and CMOS inverter are obtained by the proposed approach. In this experiment, NILT0 was implemented using a $M = 6$, $N = 4, p = 2$ and $q = 2$.

Figures 7.5 compare the waveforms obtained from the proposed NILT0 (circles) against a SPICE-like solver implemented using the TR method (dots). These graphs clearly show that NILT0 can efficiently and accurately generate circuit waveforms with larger step-size and much less number of steps.

The first row in Table 7.1 compares the key performance metrics in the SPICE solver and proposed approach, both based on MATLAB® implementation. The table displays that while SPICE required 517 points, the proposed approach needed only 14 points. The reduction in the number of time points is attributable to the larger step size h enabled by the high-order approximation of the proposed approach. Table 7.1 also compares the CPU MATLAB® computational time between TR and proposed approach for this example indicating a speedup factor of 33 times.

The circuit in Figure 7.2 was simulated in the commercial circuit simulator of HSPICE®. The runtime statistics reported by the simulator show a total number of NR iterations of 1810 and simulation time of 2.1 seconds.

Table 7.1: Performance comparison of proposed NILT0.

Ex.	MNA Size	TR		NILT0		Speedup
		#Points	CPU (sec)	#Points	CPU (sec)	
7.1.2	3623	517	6.67	14	0.2	33x

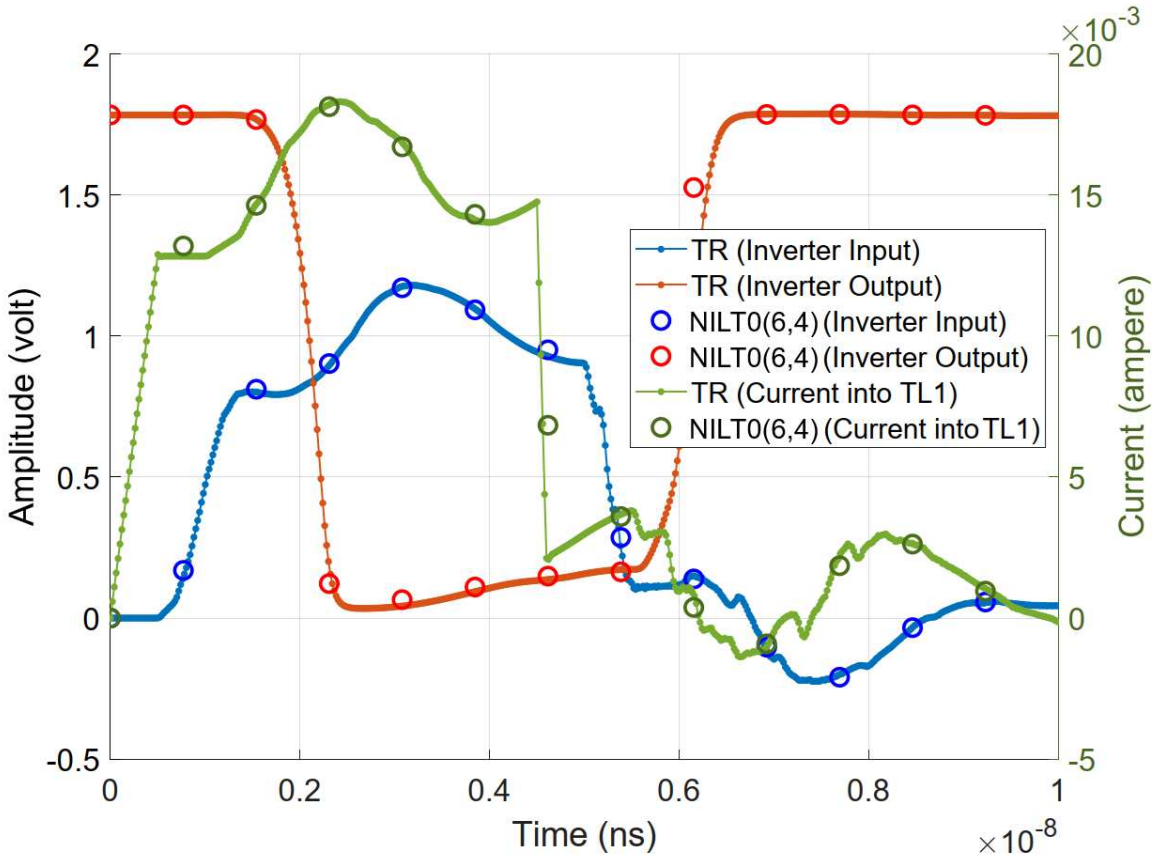


Figure 7.5: Waveform obtained in the CMOS inverter circuit

7.2 Time-Domain Simulation of Nonlinear Circuits Using NILTn

In this section, first, a simple example establishes the performance improvement achieved using NILTn by comparing the residual errors of different orders of NILT. Next, two examples show performance of the proposed NILTn and affect of its parameters M and N by simulating very large circuits including nonlinear components.

7.2.1 A Diode-based RC Circuit

This example is used to illustrate the advantages of the NILTn approach. It has a 2^{nd}-order low-pass RC filter terminated by two PN junction diodes and a 10 kΩ load resistor as shown in Figure 7.6. The circuit is excited by a 1.8 v trapezoidal pulse with 1 ns pulse width and rise/fall time of 0.1 ns.

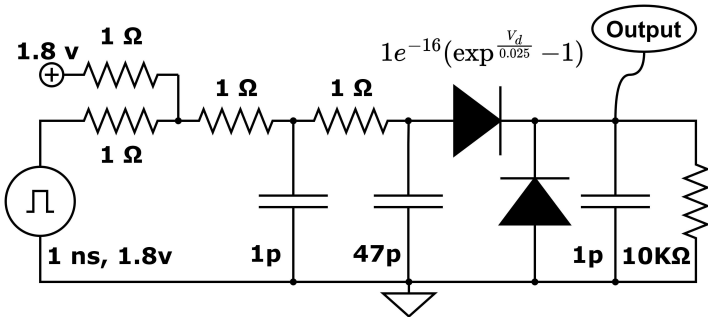


Figure 7.6: An RC circuit with PN junction diodes

The TR was first used to compute the time-domain response and create the data set used as a reference for comparison with the proposed approach. For that purpose a step size of 13 ps has been used in the TR.

Next, the proposed approach based on NILTn was used with $M = 4$, $N = 2$, $p = 2$, $q = 1$. The results in Figure 7.7 display the performance of the reinitialized-mode NILTn for $n = 0, 1$ and 2 with $h = 286\ ps$. It should be noted again that the case of $n = 0$ is equivalent to the conventional NILT.

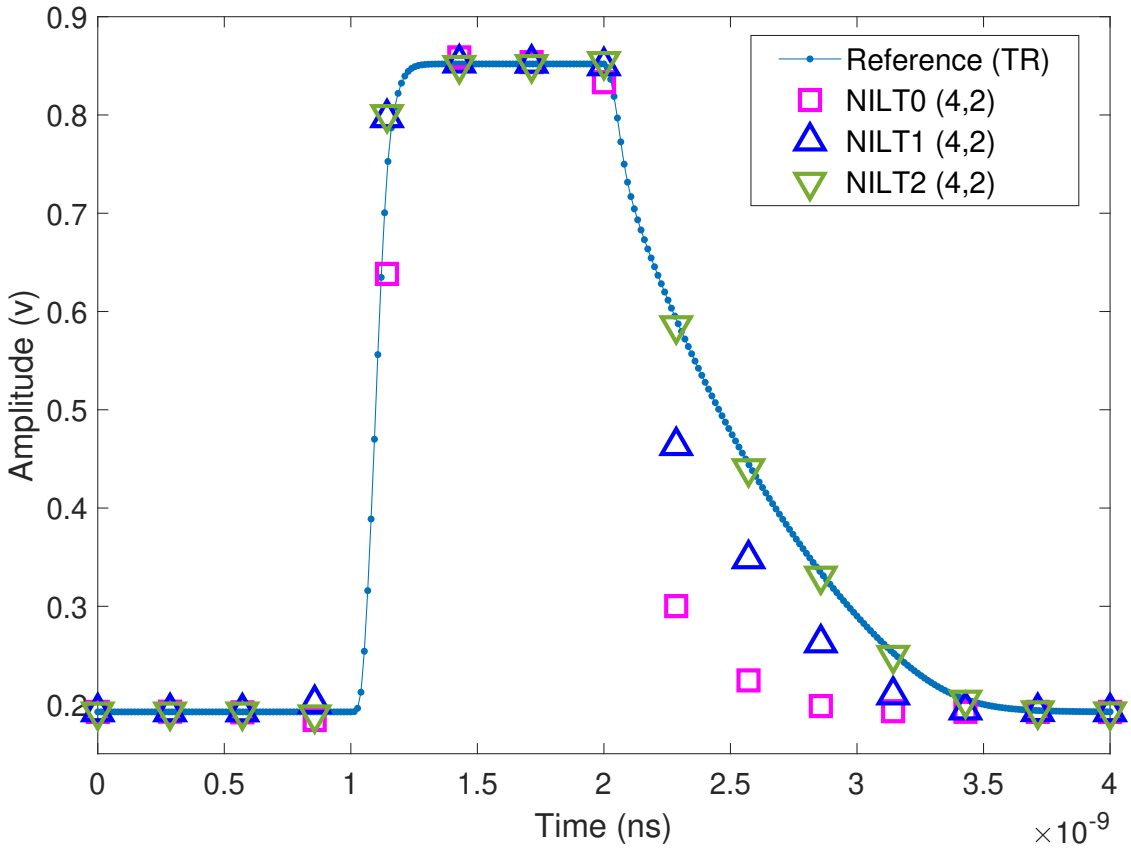


Figure 7.7: Comparison of the simulation results for the circuit in Figure 7.6.

To further highlight the efficacy of the new NILTn approach in the simulation, the circuit in Figure 7.6 was simulated with different values for the step size h using NILTn. The error arising from NILTn, and the TR (with step-size 13 ps) as the reference, are plotted in Figure 7.8, comparing the maximum error observed for $n = 0, 1$ and 2. This

result showcases the accuracy of NILTn compared to the conventional NILT0, and aligning with the underlying theory in [34] that proves the reduction in approximation error while the step-size increases, as discussed in Section 6.2.

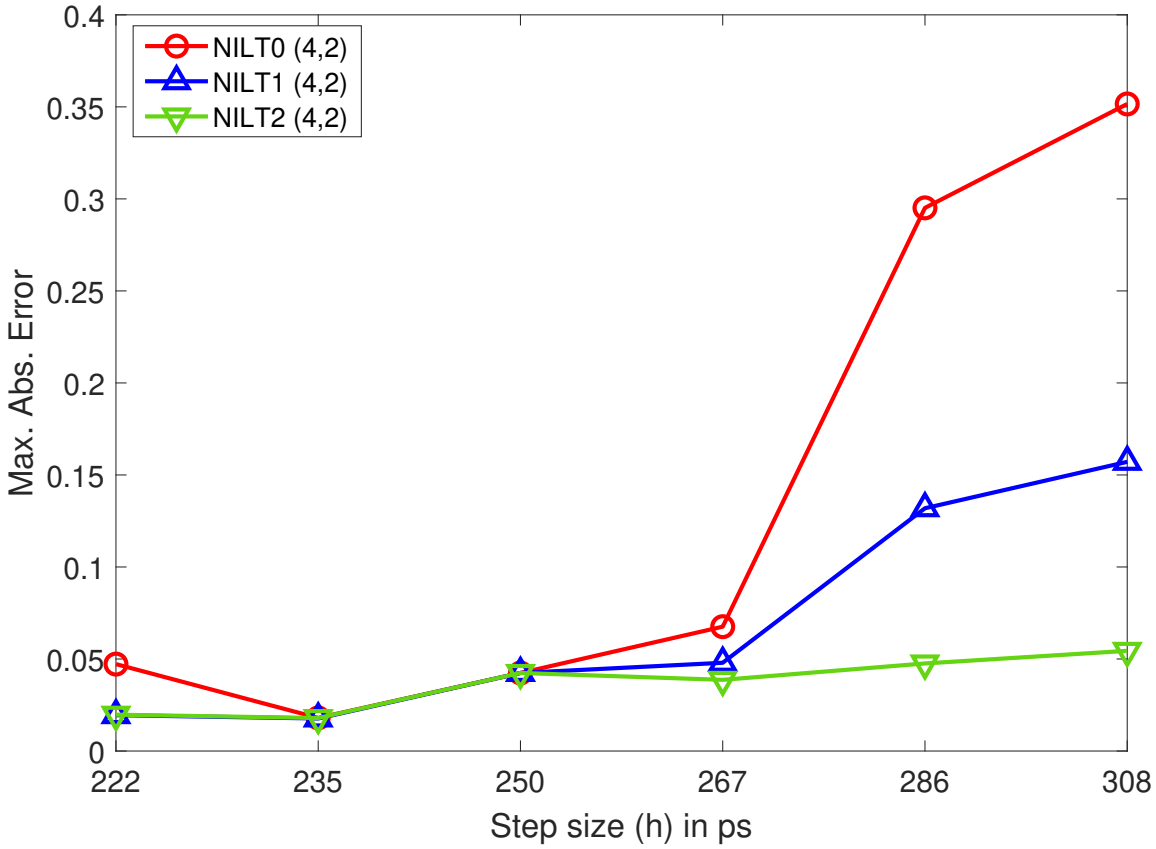


Figure 7.8: Performance of the NILTn versus length of the step size h.

7.2.2 BJT-based Transmission-Line Network

This example uses the circuit shown in Figure 7.2 which incorporates the BJT-based inverters shown on the figure and 3 networks of a coupled conductor TLs. The circuit

Table 7.2: Performance comparison

Ex.	MNA size (K)	Reference (TR)		NILT0 (8,6)			NILT2 (4,2)		
		#Time-steps	CPU(s)	#Time-steps	CPU(s)	Speedup w.r.t (TR)	#Time-steps	CPU(s)	Speedup w.r.t (TR)
7.2.2	3652	517	6.67	15	0.42	16	15	0.21	31
7.2.3	12171	630	93.9	16	1.34	69	16	0.67	139

is excited by a 1.8 v trapezoidal pulse voltage source, with 0.1 ns rise/fall times and pulse width of 1 ns.

Figures 7.9 and 7.10 show the waveforms at the input and output of the inverter, respectively. The reference waveform is obtained by computing TR method with 517 time-steps whereas the NILT-based results are obtained using only 15 time steps using $p = 2$ and $q = 1$.

These results illustrate that the accuracy achieved by the conventional method (NILT0) with settings $M = 8$ and $N = 6$ can be replicated by NILT1 or NILT2 with settings $M = 4$ and $N = 2$. This result highlights that using NILTn in the proposed method effectively reduces by half the computational cost of factorizing the matrix $\left(\bar{\mathbf{G}} + z_i\bar{\mathbf{C}}\right)$ compared to NILT0.

One should note that each additional degree of M and N requires two more LU factorization. Table 7.2 shows the performance comparison between proposed approach and the SPICE solver TR, both based on MATLAB® implementation. The first row displays that while SPICE required 517 points, the proposed approach needed only 15 points. The reduction in the number of time points is obtained by the larger step size h enabled by the high-order approximation of the proposed approach. It also compares the CPU MATLAB® computational time for this example indicating NILT0 with $M = 8$, $N = 6$ has speedup factor of 16 times over TR and NILT2 with $M = 4$ and $N = 2$ has speedup factor of 31.

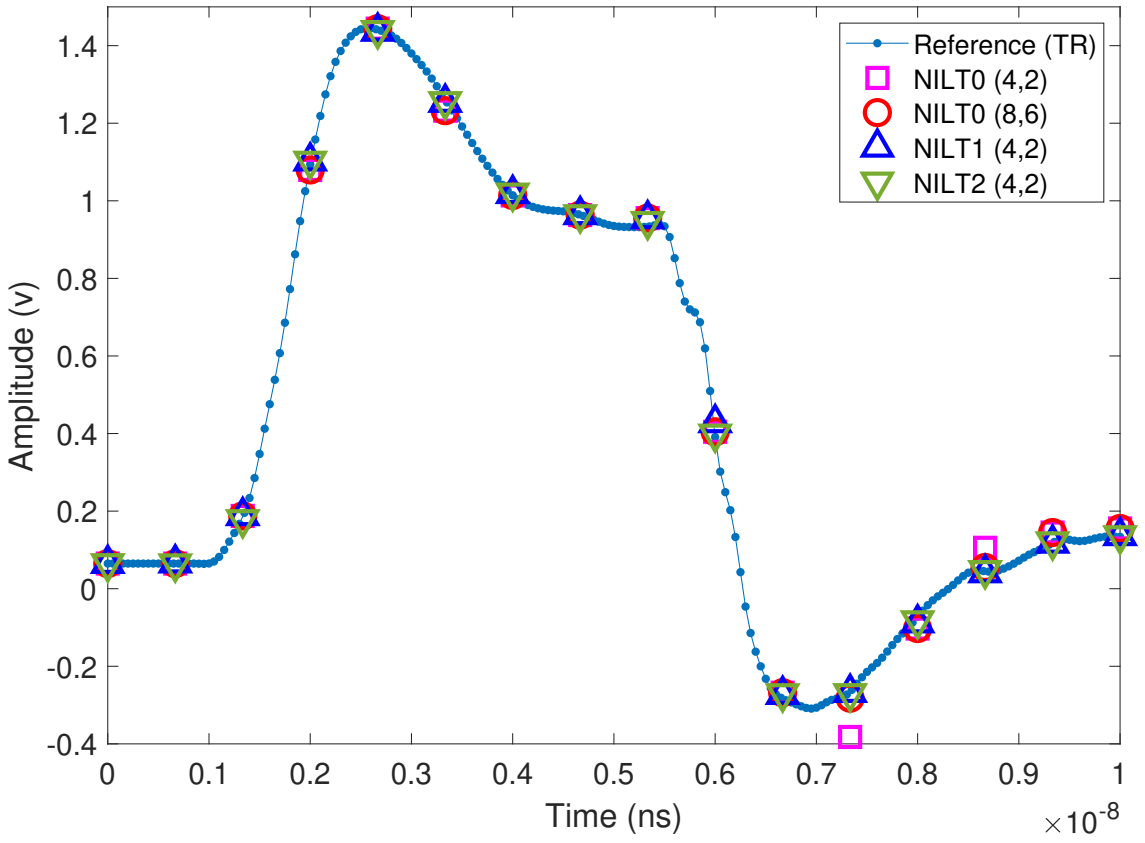


Figure 7.9: Waveform at the inverter input for NILTn.

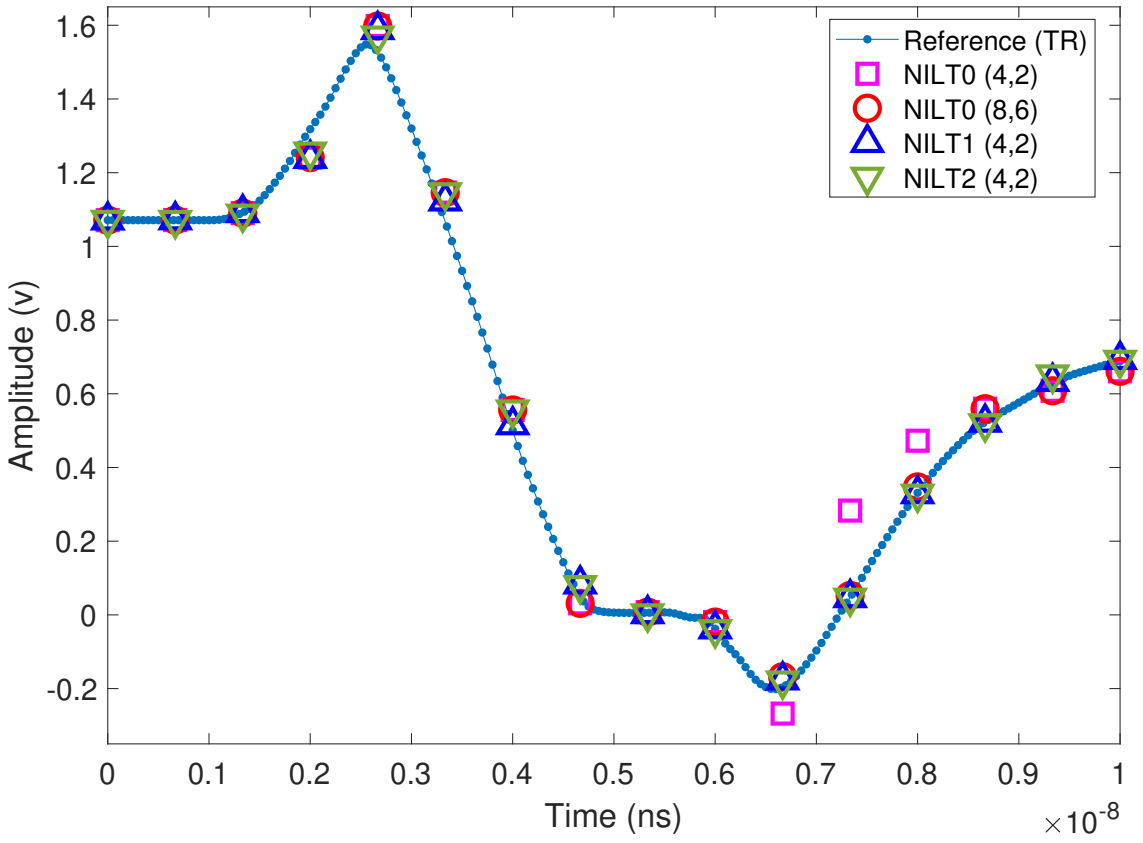


Figure 7.10: Waveform at the inverter output for NILTn.

7.2.3 CMOS-based Transmission-Line Network

This example is based on the circuit in Figure 7.11 which includes 10 TL circuits with the same parameters as previous example, and 11 CMOS inverters each constructed by one N and one P channel MOSFETs depicted on top right. The circuit MNA size is 12171 and is excited by a trapezoidal pulse similar to the previous example.

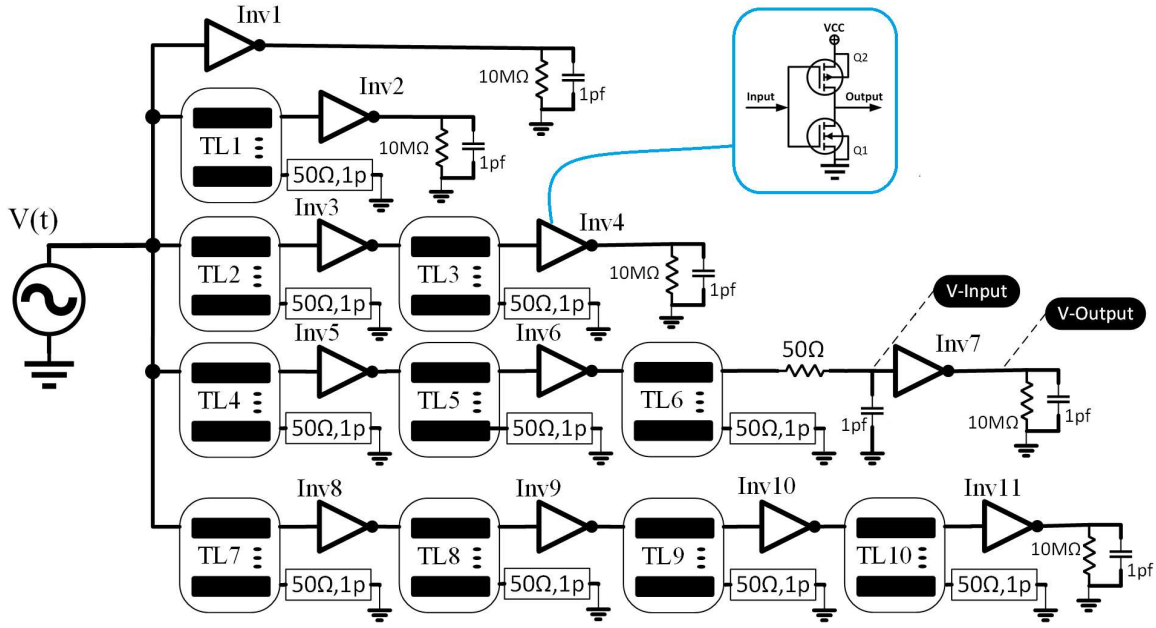


Figure 7.11: A Circuit with 10 TLs and 11 CMOS inverters.

Figures 7.12 and 7.13 show the waveform at input and output of the inverter (Inv7). The reference result is computed from the TR executed with 630 time-steps while the NILT-based methods required only 16 time-steps, with $p = 2, q = 1$.

Similar to the previous example, while NILT0 with $M = 4$ and $N = 2$ cannot accurately capture the waveforms, NILT1 and NILT2 results with $M = 4$ and $N = 2$

matches the same accuracy of the NILT0 that uses higher order of Padé approximation with $M = 8$ and $N = 6$.

The second row of the Table 7.2 shows that NILT0 with $M = 8$ and $N = 6$ is 69 times faster than TR and NILT2 with $M = 4$ and $N = 2$ is signifying a speedup of 139 times with respect to TR.

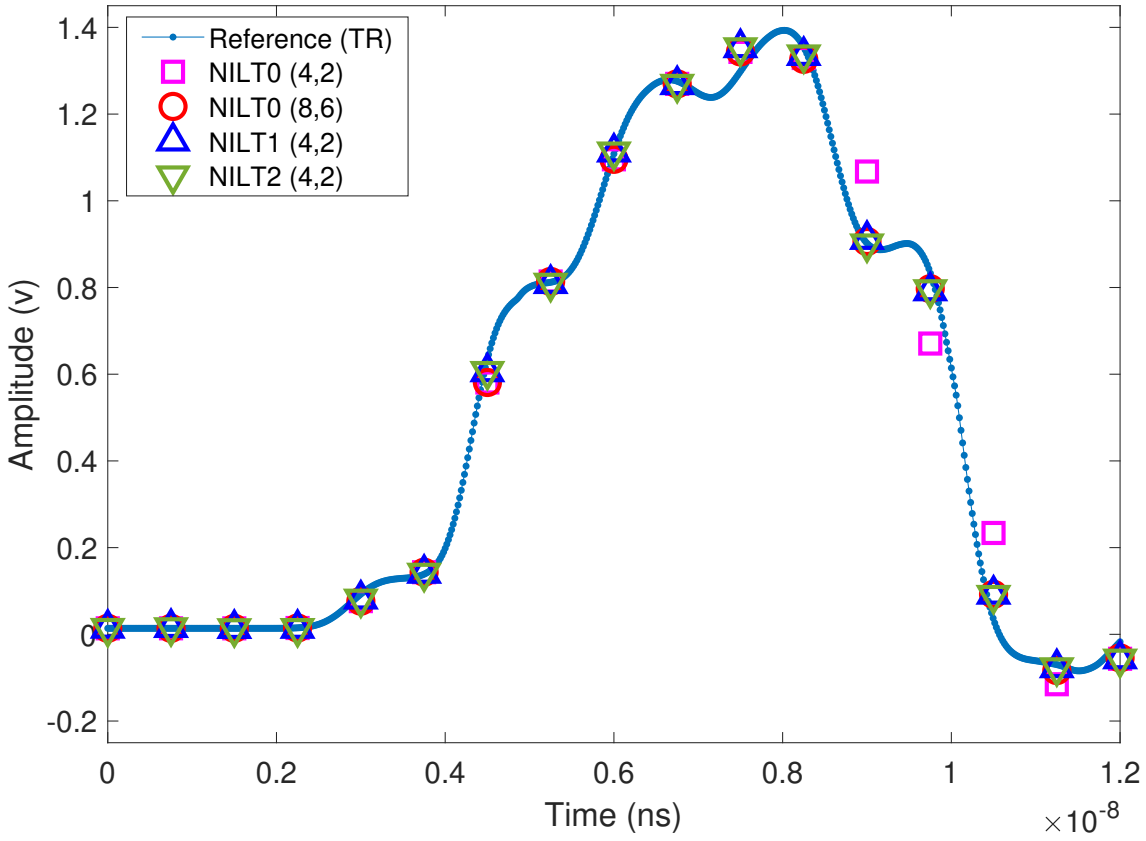


Figure 7.12: Waveform at the inverter-7 input for NILTn.

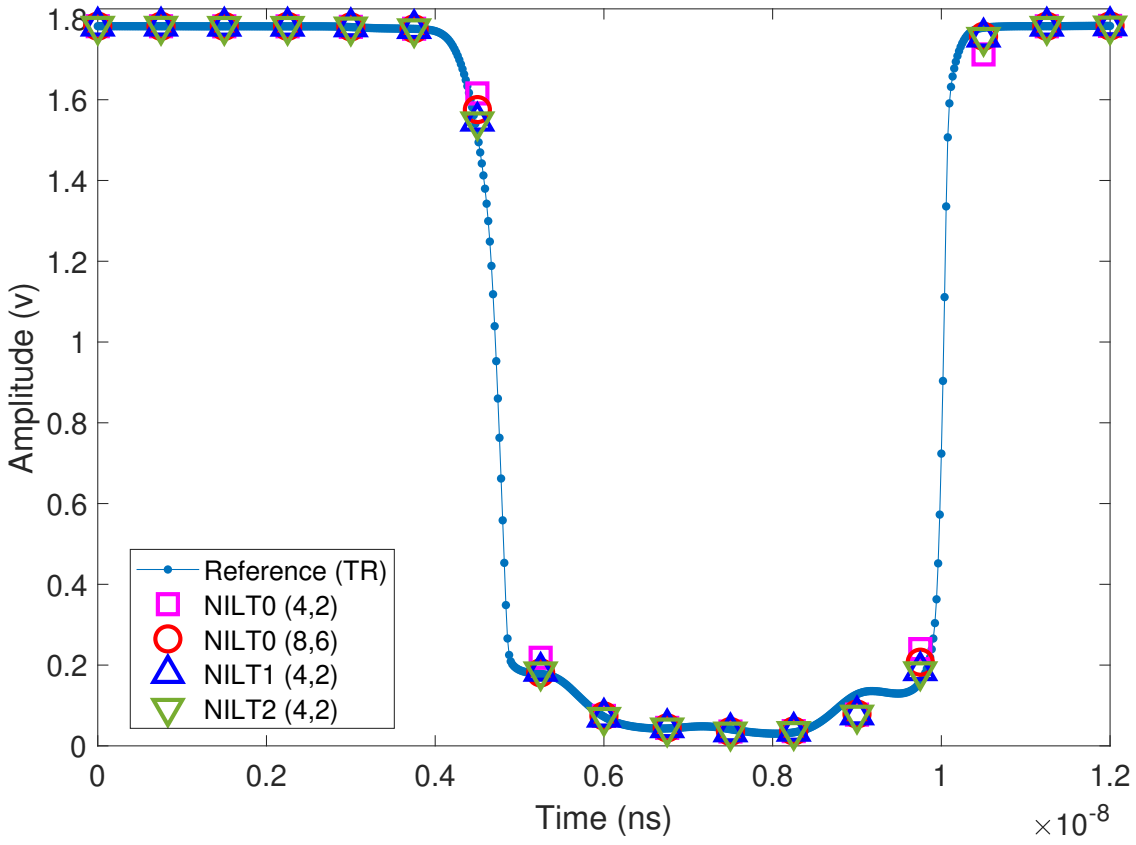


Figure 7.13: Waveform at the inverter-7 output for NILTn.

7.3 Summary

Numerical examples in this chapter showcase the contributions of this thesis through several key demonstrations. First, the effectiveness of the proposed partitioning scheme is proved by showing its utility in computing higher-order derivatives of interface currents. Next, the accuracy, stability, and efficiency of the proposed NILT method are highlighted using large nonlinear circuits. These circuits include components with complex nonlinear functions and frequency-domain distributed parameters. Furthermore, the improved accuracy of NILTn, the second key contribution of this thesis is illustrated. This is achieved by comparing the residual errors of different orders of NILTn on a simple nonlinear circuit. Finally, the superior accuracy, efficiency, stability, and speedup of the proposed method are demonstrated by simulating a very large nonlinear circuit and comparing the results with the conventional TR integration method.

Chapter 8

Conclusion and Future Work

This thesis introduced a novel approach for time-domain simulation of nonlinear circuits. A brief summary and ideas for future works are described in the following two sections.

8.1 Summary

The presented approach aimed to generalize the concept of Numerical Inversion of the Laplace Transform (NILT), traditionally applied to linear circuits, to the domain of nonlinear circuit analysis. This idea is built upon the foundational principles of time-domain analysis outlined in Chapter 2. The methodological details of this approach, including notation and the derivation of comprehensive mathematical equations, were thoroughly detailed in Chapter 3. A rigorous investigation into the stability of the proposed method, leveraging classical test equations and considering the inherent characteristics of NILT, was conducted in Chapter 4, demonstrating its robust stability

properties in the complex domain over simulation time. Furthermore, Chapter 5 established the high-order accuracy of the proposed method as a numerical integration solver through a detailed determination of its order and approximation error. Moreover, Chapter 6 explored the adaptation and extension of the traditional NILT methodology, specifically called the NILTn method, to effectively address the complexities of general nonlinear circuits within the proposed framework. This adaptation proved to be an important step, leading to significant enhancements in both the accuracy and efficiency of the developed method. Finally, Chapter 7 provided compelling illustrative examples that effectively demonstrated the precision, efficiency, and stability of the proposed method. These examples also highlighted the achieved speedup in comparison to conventional TR method, underscoring the practical advantages of the presented approach.

8.2 Future Work

The future work following this thesis could develop the presented idea further and employ it in addressing other computational challenges. In particular, the immediate objectives that are considered in the future are summarized below.

8.2.1 Multi-time Scale Analysis

Certain circuits exhibit processes operating concurrently at significantly different speeds. For instance, a closed-loop switching power converter can have fast dynamics related to feedback and slower dynamics governing overall system behavior.

NILT, being a time-domain approach grounded in frequency domain principles and

possessing L-stability, offers significant promise for analyzing systems with multiple time scales. Future research could explore and extend the NILT methodology specifically for these applications.

8.2.2 Adopting NILT in Mainstream Simulators

While this thesis contributed to the utilization of NILT in settings that are more general than before, several steps remain to be completed to allow this work to be used in mainstream applications. Some of these steps are listed below for future assistance.

- Completing the RT library by adding nonlinear equation of common electronic components and providing the capability to easily embed any new nonlinear device.
- Employing a powerful software language such as C++ to port the developed framework from MATLAB® code into a fast and optimized execution library.
- Enhancing computational efficiency by deploying adaptive step sizes and parallelization techniques.
- Testing against a wide range of circuits and comparing the results with known solvers to assess accuracy and robustness.

References

[1] J.-M. Jin, *The Finite Element Method in Electromagnetics*. John Wiley & Sons, 2015. 1

[2] J. L. Volakis, A. Chatterjee, and L. C. Kempel, *Finite Element Method Electromagnetics: Antennas, Microwave Circuits, and Scattering Applications*. John Wiley & Sons, 1998. 1

[3] D. Jiao, J.-M. Jin, E. Michielssen, and D. J. Riley, "Time-domain finite-element simulation of three-dimensional scattering and radiation problems using perfectly matched layers," *IEEE Transactions on Antennas and Propagation*, vol. 51, pp. 296–305, Feb 2003. 1

[4] C. Gear, "Simultaneous numerical solution of differential-algebraic equations," vol. 18, pp. 89–95, Jan. 1971. 2

[5] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Probelms in Differential-Algebraic Equations*. SIAM, 1996. 2

[6] J. C. Butcher, "General linear methods," *Acta Numerica*, vol. 15, pp. 157–256, 2006. 2

[7] M. Nakhla and J. Vlach, "A piecewise harmonic balance technique for determination of periodic response of nonlinear systems," *IEEE Transactions on Circuits and Systems*, vol. 23, no. 2, pp. 85–91, 1976. 2, 88

[8] E. Gad, R. Khazaka, M. Nakhla, and R. Griffith, "A circuit reduction technique for finding the steady-state solution of nonlinear circuits," vol. 48, no. 12, pp. 2389–2396, 2000. 2, 61

[9] K. Kundert and A. Sangiovanni-Vincentelli, "Simulation of nonlinear circuits in the frequency domain," vol. 5, pp. 521–535, Oct. 1986. 2

[10] S. R. Glandon, M. Narayanamurthi, and A. Sandu, "Linearly implicit multistep methods for time integration," *SIAM Journal on Scientific Computing*, vol. 44, no. 6, pp. A3437–A3462, 2022. 2

[11] Z. Li and H.-l. Liao, "Stability of variable-step BDF2 and BDF3 methods," *SIAM Journal on Numerical Analysis*, vol. 60, no. 4, pp. 2253–2272, 2022. 2

[12] V. DeCaria, A. Guzel, W. Layton, and Y. Li, "A variable stepsize, variable order family of low complexity," *SIAM Journal on Scientific Computing*, vol. 43, no. 3, pp. A2130–A2160, 2021. 2

[13] T. Buvoli, "Exponential polynomial block methods," *SIAM Journal on Scientific Computing*, vol. 43, no. 3, pp. A1692–A1722, 2021. 2

[14] A. Ostermann, M. Thalhammer, and W. Wright, "A class of explicit exponential general linear methods," *BIT Numerical Mathematics*, vol. 46, pp. 409–431, 2006. 2

[15] F. N. Najm, *Circuit Simulation*. John Wiley & Sons, 2010. 2, 15, 45, 107

[16] G. Dahlquist, "A special stability problem for linear multistep methods," *BIT*, vol. 3, pp. 27–43, 1963. 2, 42, 81

[17] K. Kundert, *The Designer's Guide to SPICE and Spectre*. Norwell, MA.: Kluwer Academic Publishers, 1995. 3

[18] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*. New York, NY, USA: John Wiley & Sons, Inc., 2nd ed., 1993. 3, 12, 33, 34, 39, 57, 84, 107, 112

[19] C. Paul, *Analysis of Multiconductor Transmission Lines*. New York: Wiley, 1994. 3, 112

[20] J. R. Griffith and M. S. Nakhla, "Time-domain analysis of lossy coupled transmission lines," vol. 38, no. 10, pp. 1480–1487, 1990. 3

[21] C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*. N.J.: Prentice-Hall, 1971. 3, 32

[22] A. Iserles and S. P. Nørsett, *Order Stars*. Chapman & Hall, 1991. 3, 87

[23] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*, vol. II. Springer, 1996. 3, 21, 81, 86, 87

[24] Y. Tao, B. Nouri, M. S. Nakhla, M. A. Farhan, and R. Achar, "Variability analysis via parameterized model order reduction and numerical inversion of laplace transform," vol. 7, pp. 678–686, May 2017. 3

[25] L. Lombardi, Y. Tao, B. Nouri, F. Ferranti, G. Antonini, and M. S. Nakhla, "Parameterized model order reduction of delayed PEEC circuits," *IEEE Transactions on Electromagnetic Compatibility*, vol. 62, no. 3, pp. 859–869, 2020. 3

[26] H. Heeb and A. E. Ruehli, "Three-dimensional interconnect analysis using partial element equivalent circuits," vol. 39, no. 11, pp. 974–982, 1992. 3, 44

[27] A. E. Ruehli, G. Antonini, and L. Jiang, *Circuit Oriented Electromagnetic Modeling Using the PEEC Techniques*. Wiley Online Library, 2017. 3, 44

[28] A. E. Ruehli, L. Lombardi, G. Antonini, Y. Tao, M. S. Nakhla, and F. Ferranti, "Impulse response for full wave PEEC models avoiding late time instability," in *2019 IEEE 28th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, pp. 1–3, 2019. 4, 44

[29] R. Griffith and M. S. Nakhla, "Mixed frequency/time domain analysis of nonlinear circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 11, no. 8, pp. 1032–1043, 1992. 4

[30] L. Lu, M. S. Nakhla, and Q.-J. Zhang, "A resetting algorithm for transient analysis of coupled transmission line circuits," *IEEE transactions on microwave theory and techniques*, vol. 42, no. 3, pp. 494–500, 1994. 4

[31] S. L. Manney, M. Nakhla, and Q.-j. Zhang, "Analysis of nonuniform, frequency-dependent high-speed interconnects using numerical inversion of Laplace transform.," vol. 13, pp. 1513–1525, Dec. 1994. 4

[32] B. Bandali, E. Gad, and M. Nakhla, "Fast and stable transient simulation of nonlinear circuits using the numerical inversion of the Laplace transform," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 12, no. 7, pp. 1171–1185, 2022. 5, 108

[33] B. Bandali, E. Gad, and M. Nakhla, "Fast and stable transient simulation of nonlinear circuits using the numerical inversion of the Laplace transform," in *2021 IEEE 25th Workshop on Signal and Power Integrity (SPI)*, pp. 1–4, 2021. 5

[34] Y. Tao, E. Gad, and M. Nakhla, "Fast and stable time-domain simulation based on modified numerical inversion of the Laplace transform," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 11, no. 5, pp. 848–858, 2021. 5, 39, 40, 92, 108, 119

[35] B. Bandali, E. Gad, and M. Nakhla, "High-order stable simulation of nonlinear circuits using modified inversion of the Laplace transform," in *IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization for RF, Microwave, and Terahertz Applications (NEMO2024)*, 2024. 5

[36] B. Bandali, E. Gad, and M. Nakhla, "Modified inversion of the Laplace transform for fast and stable time-domain simulation of nonlinear circuits," *IEEE Transactions on Microwave Theory and Techniques*, pp. 1–12, 2025. 5

[37] C.-W. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Transactions on circuits and systems*, vol. 22, no. 6, pp. 504–509, 1975. 9

[38] J. D. Lambert, *Computational Methods in Ordinary Differential Equations*. Introductory Mathematics for Scientists and Engineers, London Wiley, 1973. 20

[39] D. Martin and L. V. Ahlfors, *Complex Analysis*, vol. 3. McGraw-Hill New York, 1979. 25

[40] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 2016. 29, 81

[41] B. L. Ehle, "High order a-stable methods for the numerical solution of systems of de's," *BIT Numerical Mathematics*, vol. 8, no. 4, pp. 276–278, 1968. 30

[42] J. C. Butcher, "Implicit Runge-Kutta processes," *Mathematics of computation*, vol. 18, no. 85, pp. 50–64, 1964. 30

[43] B. Engquist, *Encyclopedia of Applied and Computational Mathematics*. Springer Heidelberg, 2015. 30

[44] N. Obreshkov, "Sur les quadrature mecanique," *(Bulgarian, French Summary) Spisanie Bulgar. Akad. Nauk*, vol. 65, pp. 191–289, 1942. 31

[45] E. Gad, M. Nakhla, R. Achar, and Y. Zhou, "A-stable and L-stable high-order integration methods for solving stiff differential equations," vol. 28, pp. 1359–1372, Sep. 2009. 32, 51, 70, 79

[46] E. Gad, "Characterizing order of convergence in the Obreshkov method in differential-algebraic equations," *arXiv:2102.02936*, 2021. 32

[47] G. A. Baker and P. Graves-Morris, *Padé Approximants*. Encyclopedia of Mathematics, New York, USA: Cambridge Univ. Press, 1995. 34, 86

[48] G. A. Baker Jr., *Essentials of Padé Approximants*. New York: Academic Press, 1975. 34, 86

[49] K. Singhal, J. Vlach, and M. S. Nakhla, *One-step, explicit, A-stable, high-order method for time domain solution of networks*. University of Waterloo, Department of Electrical Engineering, 1974. 38

[50] K. Singhal, J. Vlach, and M. Nakhla, "Absolutely stable, high order method for time domain solution of networks.," *Archiv fur Elektronik und Uebertragungstechnik (Electronics and Communication)*, vol. Vol. 30, pp. 157–166, 1976. 38

[51] A. Sangiovanni-Vincentelli, L.-K. Chen, and L. Chua, "An efficient heuristic cluster algorithm for tearing large-scale networks," *IEEE Transactions on Circuits and Systems*, vol. 24, no. 12, pp. 709–717, 1977. 52

[52] R. Rohrer, "Circuit partitioning simplified," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 1, pp. 2–5, 1988. 52

[53] G. Kron, *Tensor Analysis of Networks*. General electric series, Wiley, 1949. 52

[54] T.-W. Huang, B. Houshmand, and T. Itoh, "The implementation of time-domain diakoptics in the FDTD method," *IEEE Transactions on Microwave Theory and Techniques*, vol. 42, no. 11, pp. 2149–2155, 1994. 52

[55] A. Newton and A. Sangiovanni-Vincentelli, "Relaxation-based electrical simulation," *IEEE Transactions on Electron Devices*, vol. 30, no. 9, pp. 1184–1207, 1983. 52

[56] E. Lelarasmee, A. Ruehli, and A. Sangiovanni-Vincentelli, "The waveform relaxation method for time-domain analysis of large scale integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 1, no. 3, pp. 131–145, 1982. 52

[57] N. Frohlich, B. Riess, U. Wever, and Q. Zheng, "A new approach for parallel simulation of VLSI circuits on a transistor level," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 45, no. 6, pp. 601–613, 1998. 52

[58] P. Cox, R. Burch, D. Hocevar, P. Yang, and B. Epler, "Direct circuit simulation algorithms for parallel processing (VLSI)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 6, pp. 714–725, 1991. 52

[59] D. Paul, R. Achar, M. S. Nakhla, and N. M. Nakhla, "Addressing partitioning issues in parallel circuit simulation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2713–2723, 2014. 52

[60] I. Hajj, "Sparsity considerations in network solution by tearing," *IEEE Transactions on Circuits and Systems*, vol. 27, no. 5, pp. 357–366, 1980. 52

[61] R. Saleh, K. Gallivan, M.-C. Chang, I. Hajj, D. Smart, and T. Trick, "Parallel circuit simulation on supercomputers," *Proceedings of the IEEE*, vol. 77, no. 12, pp. 1915–1931, 1989. 52

[62] T. Parr, *The Definitive ANTLR 4 Reference*. Pragmatic Bookshelf, 2nd ed., 2013. 62

[63] R. L. Burden, J. D. Faires, and A. C. Reynolds, *Numerical Analysis 2ed.* Prindle, Weber & Schmidt, 1981. 67

[64] F. J. Hickernell and S. Yang, "Explicit Hermite interpolation polynomials via the cycle index with applications," *Int. J. Numer. Anal. Model.*, vol. 5, no. 3, pp. 457–465, 2008. 67, 68

[65] E. Hairer, S. P. Nøsett, and G. Wanner, *Solving Ordinary Differential Equations I Nonstiff Problems*. Berlin Heidelberg: Springer, 3 ed., 2008. 81

[66] K. S. Kundert, J. K. White, and A. Sangiovanni-Vincentelli, *Steady-State Methods for Simulating Analog and Microwave Circuits*. Boston: Kluwer Academic, 1990. 88

[67] E. Gad, R. Khazaka, R. Nakhla, and R. Griffith, "A circuit reduction technique for finding the steady-state solution of nonlinear circuits," *IEEE Transactions on Microwave theory and techniques*, vol. 48, no. 12, pp. 2389–2396, 2000. 88

[68] R. Achar and M. Nakhla, "Simulation of high-speed interconnects," vol. 89, pp. 693–728, May 2001. 96

[69] C. Coe, "The generalized Leibniz formula," *The American Mathematical Monthly*, vol. 57, no. 7P1, pp. 459–466, 1950. 105

[70] Y. Tao, E. Gad, and M. Nakhla, "Low-cost error estimation for fast and stable circuit simulation using modified inversion of the Laplace transform," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 12, no. 7, pp. 1160–1170, 2022. 108

[71] R. Rohrer, "Circuit partitioning simplified," *IEEE transactions on circuits and systems*, vol. 35, no. 1, pp. 2–5, 1988. 110

[72] J. J. Ebers and J. L. Moll, "Large-signal behavior of junction transistors," *Proceedings of the IRE*, vol. 42, no. 12, pp. 1761–1772, 1954. 112

[73] C. C. McAndrew, B. K. Bhattacharyya, and O. Wing, "A single-piece C_∞-continuous MOSFET model including subthreshold conduction," *IEEE Electron Device Letters*, vol. 12, no. 10, pp. 565–567, 1991. 114