**Advanced Framework for Transient Simulation of
High-Speed Circuits and Interconnects using L-Stable
High-Order Integration Methods**

by

Mina Farhan, B.Sc., M.A.Sc

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial
fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Electronics

Faculty of Engineering

Carleton University

Ottawa, Ontario, Canada

© 2015

Mina Farhan

Abstract

The trends toward higher operating frequencies and sharper rise times coupled with faster switching signals have made modeling and simulation of high-speed interconnects an increasingly challenging task. Typically, the circuit models arising from characterizing coupled high-speed interconnects are very large linear circuits with many linear components, making the transient analysis a cumbersome task. Recently, waveform relaxation methods based on transverse partitioning (WR-TP) were proposed to address this issue. It was shown that the complexity of WR-TP grows only linearly with the number of lines. However, as the coupling between the lines becomes stronger, the WR-TP algorithm either fails to converge or the number of iterations required for convergence increases.

To address the issue of convergence in WR-TP, an overlapping partitioning (OP) method is presented in this thesis proposal. Using the OP approach, the coupled interconnects are partitioned such that each subcircuit contains one or more lines that are tightly coupled, while allowing the lines between adjacent subcircuits to overlap. The weak coupling between the subcircuits are represented using voltage/current sources. Next, each subcircuit is simulated independently using a suitable integration method for the whole time of interest. The voltage/current sources representing the

coupling between the subcircuits are then updated. This process is repeated until convergence is obtained.

Typically, during the simulation of each subcircuit, an integration method is used to discretize the time points by taking finite time steps to approximate the waveforms at those time points. Recently, it has been shown that using a high-order and L -stable integration methods based on Obreshkov formula (ObF) can lead to a large reduction in the transient analysis time of general circuits. In this thesis, we show that by using ObF and taking advantage of the special structure of the mathematical formulation of the interconnect circuits, the simulation time of each subcircuit can be significantly reduced. Thus, reducing the overall simulation time.

Finally, the ObF is applied to another class of simulation methods, known as Envelope-Following (EF) methods. The EF method is suitable for transient analysis of highly oscillatory circuits with widely separated time-scales. We show that using the high-order ObF leads to a significant reduction in the computational time compared to low-order EF methods because it allows taking large steps in time while keeping the same accuracy. In addition, the new ObF-based EF method is modified to exploit multicore machines to provide fast simulation.

Acknowledgments

I would like to express my gratitude to my co-supervisors, Prof. Michel Nakhla, Prof. Ram Achar and Prof. Emad Gad, for their guidance, continuous encouragement, friendly motivation, patience and their valuable advise during my research journey. It would not have been possible without their help. One simply could not wish for a better or friendlier supervisors. I feel privileged to have them as my thesis supervisors.

I would like to give my especial thanks to my wife, Rimounda, for her loving support and encouragement. Her smile and love provided me with all the encouragement I needed to finish this thesis. Thank you for supporting me during the difficult times we had together.

I would also like to extend my gratitude to my parents, Adel and Sonia Farhan, and my brother, Michael, for their years of guidance and support. Thank you. I believe your patience finally paid off. I also like to thank all my family members for their continuous encouragement and support.

Finally, I would like to acknowledge my colleagues in the CAD group at Carleton University. Their discussions, comments and ideas made my work an enjoyable experience.

Contents

1	Introduction	1
1.1	Objective of the Thesis	7
1.2	Contributions	7
1.3	Organization of the Thesis	8
2	Overview of Waveform Relaxation and Transverse Partitioning	10
2.1	Mathematical Formulation of WR	11
2.1.1	Convergence of Waveform Relaxation	13
2.2	Simulation of Interconnect Circuits Using Waveform Relaxation and Transverse Partitioning	15
2.2.1	Single-ended Representation	18
2.2.2	Distributed Sources Representation	19
2.3	Summary	21
3	Overview of Integration Methods used in Transient Simulation of High-Speed Circuits	22
3.1	Properties of Numerical Integration Methods	24

3.1.1	Convergence and Order of the Integration Method	25
3.1.2	Stability of the Integration Method	26
3.2	Single Step Methods	28
3.2.1	Forward/Backward Euler	28
3.2.2	Trapezoidal Rule	30
3.3	Linear Multi-Step (LMS) Methods	32
3.3.1	Runge-Kutta Methods	34
3.4	Multi-derivatives Methods	37
3.5	Transient Analysis of Electric Circuits Using Obreshkov-Formula . . .	39
3.5.1	Solution of the Augmented System	43
3.6	Envelope-Following (EF) Method	45
3.7	Summary	49
4	Overlapping Partitioning Techniques for Transient Simulation of	
	Strongly Coupled High-Speed Interconnects	51
4.1	Overview of Multi-Splitting Algorithm for Solving Linear Systems . .	52
4.2	Overlapping Partitioning Algorithm for Tightly Coupled Interconnects	54
4.2.1	Illustrative example:	59
4.3	Practical Considerations	61
4.3.1	Distributed Representation	61
4.3.2	Proper Partitioning for Improved Convergence	62
4.3.3	Computational Steps	65
4.4	Frequency Dependent RLGC Parameters	67

4.5	Numerical Examples	68
4.5.1	Example 1	69
4.5.2	Example 2	70
4.6	Summary	75
5	Accelerated Overlapping Partitioning Techniques using Model Order Reduction	76
5.1	Handling of Distributed Relaxation Sources	77
5.2	Application of MOR	80
5.2.1	Updating the Relaxation Sources	81
5.3	Computational Complexity	83
5.3.1	Reduction of Subcircuits	83
5.3.2	Solving the Complete Network Containing Reduced Subcircuits	84
5.3.3	Updating the Relaxation Sources	85
5.4	Numerical Examples	86
5.4.1	Example 1	86
5.4.2	Example 2	88
5.5	Summary	91
6	Fast Transient Simulation of High-Speed Interconnects using Obreshkov-Formula	92
6.1	Development of the ObF-based Transient Analysis for Interconnects .	93
6.2	Implementation Strategies	99
6.2.1	Mapping ρ_{n+1} to ξ_{n+1}	99

6.2.2	Computing $\tilde{\mathbf{b}}$	100
6.2.3	Computing the right hand-side vector	102
6.3	Numerical Examples	103
6.3.1	Example 1	103
6.3.2	Example 2	105
6.3.3	Example 3	108
6.3.4	Example 4	109
6.4	Summary	110
7	Fast Transient Simulation of High-Speed Interconnects with Non-	
	Linear Terminations Using Obreshkov-Formula	112
7.1	Node Tearing in an Illustrative Circuit	113
7.1.1	Obreshkov-based Discretization	115
7.1.2	Solving the Discretized System	116
7.1.3	Efficient Solution Using Node Tearing	117
7.2	Node Tearing for General Nonlinear Circuits	121
7.3	Numerical Examples	123
7.3.1	Example 1	123
7.3.2	Example 2	126
7.3.3	Example 3	128
7.4	Summary	131
8	Parallel Simulation of Large Linear Circuits with NonLinear Termi-	
	nations	132

8.1	Parallel ObF-based High-Order Simulation for Nonlinear Circuits . . .	137
8.1.1	Formulation of the ObF-based methods using node tearing algorithm	138
8.1.2	Solving the partitioned system	141
8.2	Numerical Examples	148
8.2.1	Example 1	149
8.2.2	Example 2	152
8.2.3	Example 3	154
8.3	Summary	155
9	High Order and A-Stable Envelope Following Method for Transient Simulations of Oscillatory Circuits	156
9.1	Summary and Discussions of ObF-EF	160
9.2	Improved Initial Guess	162
9.3	Derivatives Approximation Using Enhanced Finite-Difference Formula	165
9.4	Stability of the ObF-based Envelope Following	166
9.5	Parallel Implementation of ObF-EF	168
9.6	Numerical Examples	171
9.6.1	Example 1	172
9.6.2	Example 2	174
9.6.3	Example 3	176
9.6.4	Example 4	178
9.7	Summary	179

10 Summary and Future Work	180
10.1 Future Work	181
A Kronecker Product	182
B Illustrative Example for the Node Tearing algorithm	183

List of Figures

2.1	Single-ended representation of the WR-sources [1]	19
2.2	Distributed sources added after each lumped section of the j^{th} line [1]	20
3.1	Using Forward Euler to approximate $x(t_1)$	23
3.2	Stability region of FE	29
3.3	Stability region of BE	30
3.4	Stability region of TR	31
3.5	Ringling of TR when applied to $\dot{x}(t) = -200x(t) + 200\cos(t)$, $x(0) = 0$	31
3.6	Stability region of 2-steps and 3-steps Adams-Moulton LMS method [2]	33
3.7	Butcher tableau for (a)BE, (b)TR	35
3.8	Illustration of the Envelope waveform	47
4.1	Overlapping partitioning of m coupled lines	54
4.2	Equivalent distributed representation for the j^{th} line in subcircuit l .	59
4.3	Three transmission lines with different coupling between the lines . .	65

4.4	Partitioning of the $\mathbf{Y}$ -parameter matrix (a) Partitioning scheme which does not result in a block-diagonal dominant $\mathbf{Y}$ matrix (b) Proper partitioning approach which results in a block-diagonal dominant $\mathbf{Y}$ matrix.	66
4.5	Physical parameters for the lines used in Example 1	69
4.6	Comparison of the time-domain responses at the far-end of victim line 2 (Example 1)	70
4.7	Partitioning of the lines in Example 1	71
4.8	Comparison of the time-domain responses at the far-end of victim line 2 (Example 1)	71
4.9	Physical parameters for the lines used in Example 2	72
4.10	Comparison of the time-domain responses at the far-end of line 2 with $q = 1$ (Example 2)	72
4.11	Comparison of the time-domain responses at the far-end of line 12 with $q = 2$ (Example 2)	73
4.12	Comparison of the time-domain responses at the near-end of line 12 with $q = 3$ (Example 2)	73
4.13	Partitioning of the lines in Example 2 in the case of $q = 3$	74
5.1	Simulating the l^{th} subcircuit with open circuit terminations to find the Thevenin-equivalent sources	77
5.2	The reduced model of the l^{th} subcircuit with the nonlinear devices attached to the lines, where m_l is the number of lines in the l^{th} subcircuit.	77

5.3	Interconnect network (Example 1)	86
5.4	Time-domain response at the far-end of line 2 in module-B (Example 1)	89
5.5	Time-domain response at the far-end of line 1 in module-C (Example 1)	89
5.6	Interconnect network with non-linear terminations (Example 2)	89
5.7	Time-domain response at the near-end of victim line 3 (Example 2) .	90
6.1	Interconnects circuit for Example 1	104
6.2	Waveform comparison at the far-end of victim line 2 (Example 1) . .	104
6.3	Interconnects circuit for Example 2	107
6.4	CPU time comparison of one LU factorization (Example 2)	107
6.5	Waveform comparison at the near-end of aggressor line 12 (Example 2)	107
6.6	Large coupled interconnects network (Example 3)	108
6.7	Waveform comparison at the far-end of victim line 6 in segment 2 (Example 3)	109
6.8	Eight ports linear circuit (Example 4)	110
6.9	Waveform comparison at port 2 (Example 4)	111
7.1	Illustration of the notion of Node Tearing applied to a simple circuit.	114
7.2	Interconnects Circuit (Example 1)	124
7.3	Comparison of transient waveforms at far-end of victim Line 2 (Exam- ple 1)	125
7.4	Comparison of transient waveforms at input node of the inverter at- tached to Line 1 (Example 1)	126

7.5	Comparison of transient waveforms at output node of the inverter attached to Line 1 (Example 1)	126
7.6	Interconnects Circuit (Example 2)	127
7.7	Comparison of transient waveforms at far-end of line 8 (Example 2) .	128
7.8	Eight coupled transmission lines physical structure (Example 3) . . .	129
7.9	Circuit for Example 3	129
7.10	Comparison of transient waveforms at p10 of subnetwork B (Example 3)	130
8.1	The structure of the matrix arising from applying the proposed method compared to the direct application of high-order integration method. .	135
8.2	Flow Chart of the computational steps for a particular NR iteration using the parallel framework.	148
8.3	48-Coupled interconnect circuit used in Example 1	149
8.4	Waveform at the near-end of active line 1 (Example 1)	151
8.5	Waveform at the far-end of victim line 2 (Example 1)	151
8.6	CPU time comparison for finding the solution at $t = 2\text{ns}$ (Example 1)	151
8.7	64-Coupled interconnect circuit with nonlinear loads used in Example 2	152
8.8	Waveform at the far-end of line 9 (Example 2)	153
8.9	Waveform at the far-end of line 10 (Example 2)	154
8.10	Multiple interconnects circuit with nonlinear loads (Example 3)	154
9.1	Block diagram of the circuit used in Example 1	172
9.2	Schematic of the mixer circuit (Example 1)	173
9.3	Output waveform (Example 1)	174

9.4	Detailed simulation using the envelope points (Example 1)	174
9.5	Steady-State response at the output node (Example 1)	175
9.6	Bandpass filter. $R_1 = R_6 = 10k\Omega$, $R_2 = 1k\Omega$, $R_3 = 650k\Omega$, $R_4 = R_5 =$ $15.9k\Omega$, $C_1 = C_2 = 1nF$ (Example 2).	175
9.7	The OpAmp circuit used in Example 2	176
9.8	Tuned amplifier circuit (Example 3)	177
9.9	DC-DC converter circuit.	178
9.10	Sample of the transient response for the inductor current waveform. .	179
B.1	An example used to illustrate the partitioning technique, with $N = 2$ and $P = 2$	183

List of Algorithms

2.1	Waveform Relaxation and Transverse Partitioning	20
3.1	Block-LU factorization	45
3.2	BFS (Block Forward Substitution)	46
3.3	Block Forward/Backward Substitution	46
4.1	Waveform Relaxation with Overlapping Partitioning Algorithm	67
4.2	Partitioning Algorithm for the overlapping method	67
5.1	TL analysis via OP and MOR	84
6.1	Modified Block-LU factorization	97
6.2	Modified BFS (Block Forward Substitution)	97
6.3	Modified Block Forward/Backward Substitution	98

List of Tables

4.1	CPU time comparison for the proposed WR-TP with OP for varying q (Example 2)	75
5.1	CPU-time and performance comparison for Example 1	88
5.2	CPU-time and performance comparison for Example 2	91
6.1	CPU time comparison of Example 1	105
6.2	CPU time comparison of Example 2 for L -stable methods	106
6.3	CPU time comparison of Example 3	109
6.4	CPU time comparison of Example 4	110
7.1	CPU-time comparison for Example 1	125
7.2	CPU-time comparison for Example 2	128
7.3	CPU-time comparison for Example 3	130
8.1	Required number of processors versus the order of the Obreshkov formula.	137
8.2	Comparison of the total CPU time for Example 2	153
8.3	Comparison of the total CPU time for Example 3	155

9.1	CPU-time Comparison (Example 1)	173
9.2	CPU-time Comparison (Example 2)	176
9.3	Effect of Explicit-Predictor on the total number of iterations (Example 3)	177
9.4	CPU-time Comparison for Obf-EF Order 6 with $H = 25T$	179

Abbreviations

p.u.l	Per Unit Length.
TL	Transmission Line.
WR	Waveform Relaxation.
WR-TP	Waveform Relaxation and Transverse Partitioning.
OP	Overlapping Partitioning.
KCL	Kirchhoff's Current Law.
IVP	initial Value Problem.
ODE	Ordinary Differential Equation.
DAE	Differential Algebraic Equation.
NR	Newton-Raphson iterative method.
MNA	Modified Nodal Analysis.
ObF	Obreshkov Formula.
TR	Trapezoidal Rule integration method.

FE	Forward Euler integration method.
BE	Backward Euler integration method.
LMS	Linear Multi-Step integration methods.
RK	Runge-Kutta integration method.
EF	Envelope-Following method.

List of Symbols

$\bar{R}, \bar{L}, \bar{G}, \bar{C}$	p.u.l parameter matrices of a TL.
t	Time.
I	An Identity matrix.
$R(z)$	The stability function of an integration method.
$\otimes$	The Kronecker product operator.
G	A matrix that represents the memory-less elements in the MNA equations.
C	A matrix that represents the memory elements in the MNA equations.
$f(\boldsymbol{x}(t))$	A vector that contains the nonlinear currents flowing in nonlinear resistors and nonlinear charges across nonlinear capacitors in the MNA equations.
$\boldsymbol{b}(t)$	A vector that contains the values of the independent sources in a circuit at a given time t .
$\tilde{G}$	An augmented G Matrix that results from applying ObF to MNA equations.
$\tilde{C}$	An augmented C Matrix that results from applying ObF to MNA equations.

$\tilde{\boldsymbol{\eta}}$	A vector that contains the high-order derivatives w.r.t to MNA equations unknowns, and results from applying ObF to the MNA equations.
$\tilde{\boldsymbol{b}}(t)$	A vector that contains the derivatives of the independent sources in a circuit at a given time t .
$\tilde{\boldsymbol{M}}$	A matrix equals to $\tilde{\boldsymbol{G}} + \tilde{\boldsymbol{C}}$
$\boldsymbol{T}$	A Hessenberg matrix that contains the coefficients of derivatives in the ObF.
$\boldsymbol{\Upsilon}$	A block matrix that contains the eigen-vectors of $\boldsymbol{T}$ on its diagonals ($\boldsymbol{\Upsilon} = \boldsymbol{I} \otimes \boldsymbol{V}$, with $\boldsymbol{V}$ is the eigen-vectors matrix of $\boldsymbol{T}$).
$\tilde{\boldsymbol{\Lambda}}$	A matrix that has the same block sparsity pattern as $\tilde{\boldsymbol{M}}$ but with each block being diagonal. $\tilde{\boldsymbol{\Lambda}} = \boldsymbol{\Upsilon}^{-1} \tilde{\boldsymbol{M}} \boldsymbol{\Upsilon}$
$\boldsymbol{J}$	The Jacobian matrix.

Chapter 1

Introduction

Due to the increasing demand for higher operating frequencies and sharper excitation, accurate representation of interconnects effects; such as crosstalk, delay, distortion, reflections and ringing, becomes a critical task. Neglecting interconnect effects during the design process may cause failure of the device when fabricated. The major difficulties in simulation of high-speed interconnect circuits is that distributed interconnect models do not have a direct representation in the time-domain. As a result, they are generally represented in the frequency-domain. On the other hand, nonlinear devices such as drivers and receivers are best described in the time-domain. These simultaneous formulations are difficult to handle by ordinary differential equation solvers existing in traditional circuit simulators.

Several methods have been proposed in the literature to address the mixed frequency/time representation [3–7]. The common goal of these techniques is to transform the Telegrapher’s equations describing the transmission lines, into a set of coupled ordinary differential equations (ODEs) that can be integrated with circuit simulators. However, when the number of coupled lines increases, the cost of solving the

resulting ODEs increases significantly.

It was shown in [1] that the average cost of simulating an m -coupled line circuit is proportional to m^α , where $2 \leq \alpha \leq 3$. To handle this problem, waveform relaxation (WR) based on transverse partitioning (WR-TP) was proposed as an efficient way for simulating large coupled interconnect circuits [1]. Using WR-TP, the task of simulating a large number of coupled lines is effectively reduced into a set of sub-problems, each consisting of simulating a single line. The coupling effects due to the neighboring lines are represented by voltage/current sources within the subcircuits. A relaxation-based algorithm is used to iterate between the subcircuits until convergence is achieved. It was shown in [1] that the computational complexity of WR-TP increases only linearly with the number of lines (i.e $O(m)$).

However, the convergence rate of WR algorithms depends on the nature of coupling between the partitioned lines. As the coupling between the lines increases, the number of iterations required for WR-TP convergence also increases. As a result, the WR-TP algorithm may require a large number of iterations (or sometimes even fail to converge) in the case of tightly coupled lines. To address this problem, a new WR with overlapping partitioning (OP) method for tightly coupled interconnects is presented in this thesis. The new method exploits the multisplitting partitioning algorithm for solving linear systems [8–10]. Using the proposed OP-based method not only preserves the advantages of the classical WR-TP, but also increases the convergence rate in the case of tightly coupled interconnects.

In the proposed OP, the coupled interconnects are partitioned such that each sub-circuit contains one or more lines that exist in an adjacent subcircuit. In addition,

each subcircuit contains the lines that are strongly coupled, while the weak coupling between the adjoint subcircuits is represented using voltage/current sources attached to each line. The coupling between adjacent subcircuits is represented using voltage/current sources known as relaxation sources. Similar to WR-TP, the proposed OP is also based on the WR iterative technique. The proposed OP starts by assuming an initial guess for relaxation sources. Each subcircuit is simulated independently using an appropriate integration method for the whole time-interval. Using the solution of each subcircuit, the relaxation sources are updated and the next iteration starts.

In order to allow a fast and accurate transient analysis using OP, the simulation of each subcircuit must be performed very fast in order not to affect the overall simulation time. In addition, the final solution must be accurate in order to accurately update the relaxation sources. Typically, the transient simulation of each subcircuit is performed using classical SPICE-like simulators [11]. These subcircuits are modeled by discretizing the lines into finite elements and each element is modeled using lumped components [3]. However, in the case of very strong coupling between the lines, more lines have to be combined in the same subcircuit, which in turn increases the size of each subcircuit. Therefore, the resulting subcircuits, in this case, are large linear circuits that are simulated independently for the whole time-interval using SPICE-like simulators.

The core of the SPICE engine is a differential algebraic equations (DAEs) that models the dynamics of the circuit. The resulting DAEs are solved, using an appropriate DAEs solver, by discretizing the time, and taking finite time steps approximating the circuit waveforms at those time points. However, the DAE solvers used in SPICE-

like simulators are derived from a class of DAEs solvers known as Linear Multi Step (LMS) methods [3]. This class of methods suffer from a conflict between order and numerical stability. The order of the method represents how fast the approximated waveform converges to the exact one. Basically, high-order methods require larger step size (thus fewer time-points) to accurately represent the waveform. The numerical stability refers to the stability of the dynamics of the time-stepping algorithm.

In general, it is desired that the DAE solvers be stable for all types of stable circuits, in addition to having high-order convergence, in order to accurately represent the waveform with small number of time points, thus, saving in the total simulation time. Unfortunately, LMS methods (used in SPICE-like simulators) can not afford high-order without losing stability. This characteristic of LMS methods is known as the Dahlquists barrier [12, 13].

Recently, it was shown in [14] that there exists a different class of methods based on Obreshkov formula (ObF) [14, 15] that is free from the theoretical barriers inherent in LMS methods. The main idea of ObF based methods is to approximate the waveform at finite time-points, similar to LMS methods. However, at each time-point the high-order derivatives are also computed. The order of the integrating method is determined by the number of computed derivatives at two successive time-points, thus high-order methods can be easily derived. It was shown in [14, 16] that, with a proper formulation and implementation, ObF-based methods can surpass the performance of conventional simulators, such as SPICE. The ObF-based methods were successfully applied to general nonlinear circuits showing a significant reduction in the simulation time compared to conventional SPICE-like simulators.

In this thesis, we show that by taking advantage of the special structure of the matrices arising from modeling interconnect structures, ObF-based methods can provide even a greater performance and higher speed-ups. The proposed methods presented in this thesis take advantage of the dominant linear nature of those circuits to reduce the computational efforts and produce faster simulations that surpass previously reported results of ObF based methods.

The final contribution in this thesis targets the transient simulation of another class of circuits known as highly oscillatory circuits, where the behavior of the circuit is characterized by high-frequency components whose amplitudes are slowly varying. Typically, a circuit designer is interested in simulating the steady-state response to determine those figures of merit such as gain or harmonic distortion. However, in other situations the transient response of microwave circuits can be of equal importance. Examples of those situations include, the startup/shutdown of oscillators, the frequency modulation of Voltage-Controlled Oscillators (VCOs) and the injection locking of subharmonically injected oscillators.

Envelope-based methods have been introduced in order to overcome the difficulties of simulating those types of circuits. The main idea in those methods is premised on the fact that the response waveform has widely separated time scales that can be handled independently. The approaches presented under the envelope-based framework may be grouped in two distinct classes: the Envelope-Fourier techniques [17–20] and the Envelope Following (EF) [21–23].

The basic idea in the EF method hinges on the assumption that a trajectory (also known as the “envelope”), which intersects with the original waveform at multiples of

the period of the fast frequency component, satisfies an implicit differential equation. Therefore, classical time-stepping methods, such as BE and TR, may be invoked to trace that trajectory, thereby enabling the simulator to skip many cycles of the highly oscillatory component, enabling significant reduction in simulation time.

Nonetheless, the basic time stepping mechanism used in SPICE-based simulation is of the low-order type which is grounded in the general class of Linear Multistep (LMS) methods. On the one hand, the aspect of low-order approximation is computationally disadvantageous since it forces the simulator to take smaller steps even though the envelope waveform may be slowly varying and could be traced with larger strides. The low-order in this case is simply not a choice, but is rather mandated by the numerical stability considerations without which the results become practically useless. It is a well-known fact that the LMS framework loses the important property of A -stability for orders higher than 2 [3,24]. This situation imposes a particular dilemma, where guaranteeing the numerical stability comes at the expense of computational efficiency.

The purpose of the technique presented in this thesis is to introduce the idea of ObF-based simulation to the problem of EF. As will be shown, the combination of numerical stability with the inherent high-order of the ObF, represents the key to more efficient EF-based simulation compared with the low-order EF techniques.

1.1 Objective of the Thesis

The main objective of this thesis is to advance the state-of-art in simulation techniques focused on high-frequency interconnects circuits and systems. Particularly, transient simulation challenges concerning tightly coupled multi-conductor interconnects as well as large linear circuits terminated with nonlinear devices are targeted.

The thesis also targets developing fast simulation techniques for another class of circuits known as highly oscillatory and fast switching circuits. Particular emphasis is also placed to develop simulation techniques exploiting multicore machines.

1.2 Contributions

The main contributions of this thesis are as follows:

1. A new method based on overlapping partitioning (OP) techniques for fast transient analysis of tightly coupled interconnect circuits, is presented. The new method takes advantage of the special characteristics of the TL circuits to provide a significant reduction of the overall CPU-time (Chapter 4 [25,26]).
 2. A new method that combines the merits of both OP techniques and Model Order Reduction (MOR) methods to provide even faster simulation for tightly coupled interconnect circuits is proposed (Chapter 5 [27]).
-

3. A new method based on high-order and L -stable ObF for fast transient analysis of linear networks arising from modeling of high-speed interconnect circuits is proposed (Chapter 6 [28, 29]).
4. A new method that exploits the node tearing approach and ObF to allow a fast transient analysis of high-speed interconnects terminated with nonlinear devices is proposed (Chapter 7 [30, 31]).
5. A parallel transient analysis method for large linear networks with nonlinear devices attached to the terminals using ObF-based methods is proposed. The new method takes advantage of the multicore processors to provide higher speed ups (Chapter 8 [32, 33]).
6. A new ObF-based Envelope-Following method is developed. The new method combines the merits of both ObF and EF approaches to provide efficient transient simulation for highly-oscillatory and fast switching circuits (Chapter 9 [34–36]).

1.3 Organization of the Thesis

The rest of this thesis is organized as follows. Chapter 2 gives an overview of the waveform relaxation method. Chapter 3 gives an overview of various integration methods used for solving circuits equations, in addition to giving an overview of the high-order and stable ObF. Chapter 4 presents a fast transient analysis method for high-speed interconnect circuits using OP methods. Chapter 5 shows how the MOR method can

be combined with OP methods to provide more speed-up. Chapter 6 presents a fast time-domain method, which is based on ObF, to simulate high-speed interconnects. Chapter 7 shows how the method presented in chapter 6 can be extended to include nonlinear devices attached to large high-speed linear circuits. Chapter 8 shows how the ObF-based methods presented can be modified to exploit multicore machines. Chapter 9 presents a new method that combines both ObF and EF techniques to simulate highly-oscillatory circuits. Finally, Chapter 10 provides the conclusion and future work based on the proposed thesis.

Chapter 2

Overview of Waveform Relaxation and Transverse Partitioning

In this chapter, an overview of transient analysis of high-speed interconnects using waveform relaxation and transverse partitioning (WR-TP) technique is given. WR-TP was proposed as an efficient and fast way to simulate large coupled interconnect circuits [1, 37]. Using WR-TP, the task of simulating a large number of coupled lines is effectively reduced into a set of subproblems, each consisting of simulating a single line. The coupling effects due to the neighboring lines are represented by voltage/current sources within the subcircuits. A relaxation-based algorithm is used to iterate between the subcircuits until convergence is achieved. It was shown in [1] that the computational complexity of WR-TP increases only linearly with the number of lines. In addition, the algorithm is highly suitable for parallel implementation since the simulation of individual subcircuits can be performed simultaneously [38].

We start by presenting a brief mathematical background on waveform relaxation methods. Next, a detailed overview of WR-TP technique is presented.

2.1 Mathematical Formulation of WR

Consider a linear initial value problem defined by the following differential equation

$$\dot{\mathbf{x}}(t) + \mathbf{A}\mathbf{x}(t) = \mathbf{b}(t) \quad , \quad \mathbf{x}(0) = \mathbf{x}_0 \quad (2.1)$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is non-singular, $\mathbf{x}(t) \in \mathbb{R}^n$ is a vector that contains the unknown variables at time t , $\dot{\mathbf{x}}(t)$ is the time derivatives of the unknown vector $\mathbf{x}(t)$, $\mathbf{b}(t)$ is the input vector at time t .

In order to use WR method, the matrix $\mathbf{A}$ is split into $\mathbf{A} = \mathbf{M} - \mathbf{N}$. Next, (2.1) is solved iteratively until convergence is obtained. Typically, the matrix $\mathbf{A}$ is split using one of the two methods: The Gauss-Jacobi or Gauss-Seidel methods. In the case of Gauss-Jacobi, the matrix $\mathbf{M}$ is chosen to be diagonal, while in the case of Gauss-Seidel, $\mathbf{M}$ is chosen to be a lower triangular matrix. One of the main advantages of Gauss-Jacobi over Gauss-Seidel is that it is more suitable for parallel computing, however, it has less convergence rate.

Using the splitting of the matrix $\mathbf{A}$, (2.1) can be written as

$$\dot{\mathbf{x}}(t) + \mathbf{M}\mathbf{x}(t) = \mathbf{N}\mathbf{x}(t) + \mathbf{b}(t) \quad (2.2)$$

In order to apply WR, an iterative technique is used by considering the right hand side of (2.2) as an input that is calculated from the previous iteration as follows

$$\dot{\mathbf{x}}^{(r+1)}(t) + \mathbf{M}\mathbf{x}^{(r+1)}(t) = \mathbf{N}\mathbf{x}^{(r)}(t) + \mathbf{b}(t) \quad (2.3)$$

where r represent the iteration number. Suppose that the Gauss-Jacobi method is used, in this case (2.3) can be written as

$$\begin{aligned} \begin{bmatrix} \dot{x}_1(t) \\ \vdots \\ \dot{x}_n(t) \end{bmatrix}^{(r+1)} + \underbrace{\begin{bmatrix} A_{11} & & \\ & \ddots & \\ & & A_{nn} \end{bmatrix}}_{\mathbf{M}} \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}^{(r+1)} = \\ \underbrace{\begin{bmatrix} 0 & \dots & -A_{1n} \\ \vdots & \ddots & \vdots \\ -A_{n1} & \dots & 0 \end{bmatrix}}_{\mathbf{N}} \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix}^{(r)} + \begin{bmatrix} b_1(t) \\ \vdots \\ b_n(t) \end{bmatrix} \end{aligned} \quad (2.4)$$

where A_{ji} is the element at the j^{th} row and the i^{th} column of the matrix $\mathbf{A}$.

It can be seen that (2.4) consists of n decoupled equations, where $\mathbf{N}$ represents the coupling effect between the decoupled equations. In this case, the j^{th} equation can be written as

$$\dot{x}_j^{(r+1)}(t) + A_{jj}x_j^{(r+1)}(t) = \sum_{\substack{i=1 \\ i \neq j}}^n A_{ji}x_i^{(r)}(t) + b_j(t) \quad (2.5)$$

Typically, the iterations are started by assuming an initial guess $\mathbf{x}^{(0)}(t)$. The system in (2.3) is then solved for the whole time interval of interest. The coupling

$\mathbf{N}\mathbf{x}^{(0)}(t)$ between the decoupled equation is then calculated and used as an input to the decoupled differential equation, and $\mathbf{x}^{(1)}(t)$ is calculated. The iterations are then continued until convergence is obtained.

2.1.1 Convergence of Waveform Relaxation

Convergence is one of the most important characteristic of any iterative technique, such as waveform relaxation. Typically, an iterative technique is required to converge to the right solution as fast as possible, otherwise, the iterative technique would have no practical use. The convergence of WR has been studied in the literature for a long time, see for example [39–42]. The main conclusion of these studies is that WR converges to the correct solution provided that necessary convergence conditions are met. In this section, a brief overview of the WR convergence is presented.

In order to study the convergence of waveform relaxation methods, we start by considering the Laplace transformation of the the exact equation (2.2) given as

$$(\mathbf{M} + s\mathbf{I})\mathbf{X}(s) = \mathbf{N}\mathbf{X}(s) + \mathbf{x}(0) + \mathbf{B}(s) \quad (2.6)$$

where s is the transform variable, $\mathbf{I}$ is the identity matrix, $\mathbf{X}(s)$ and $\mathbf{B}(s)$ are the Laplace transform of $\mathbf{x}(t)$ and $\mathbf{b}(t)$, respectively.

Similarly, the Laplace transformation of the WR iterations (2.3)

$$(\mathbf{M} + s\mathbf{I})\mathbf{X}^{(r+1)}(s) = \mathbf{N}\mathbf{X}^{(r)}(s) + \mathbf{x}(0) + \mathbf{B}(s) \quad (2.7)$$

Subtracting (2.6)-(2.7) results in

$$\boldsymbol{\epsilon}(s) = \mathbf{K}(s)(\mathbf{X}(s) - \mathbf{X}^{(r)}(s)) \quad (2.8)$$

where $\boldsymbol{\epsilon}(s) = \mathbf{X}(s) - \mathbf{X}^{(r+1)}(s)$ is the error in the $(r+1)^{th}$ iteration and

$$\mathbf{K}(s) = (\mathbf{M} + s\mathbf{I})^{-1}\mathbf{N} \quad (2.9)$$

with $\mathbf{K}(s)$ being referred to in the literature as the “iteration matrix“.

Lemma 1. *The WR iterations in (2.7) is convergent (i.e. $\lim_{r \rightarrow \infty} \boldsymbol{\epsilon}(s) = 0$) iff $|\rho(\mathbf{K}(s))| < 1$, where $\rho(\cdot)$ is the spectral radius [39].*

Lemma 1 provides a test for convergence. However, in very large systems, finding the spectral radius of $\mathbf{K}(s)$ for all frequency of interest might be time consuming. The next lemma [40] provides a more practical and easy method for testing for convergence.

Lemma 2. *If $\mathbf{K}(s)$ is diagonally dominant (i.e. $\sum_{j \neq i} |K_{i,j}(s)| < |K_{i,i}(s)|$), then $|\rho(\mathbf{K}(s))| < 1$ [40].*

Lemma 2 provides a faster approach for checking the convergence of the WR

method by checking whether the iteration matrix $\mathbf{K}(s)$ is diagonally dominant or not. However, it must be noted that the condition in Lemma 2 is sufficient but not necessary. In other words, if the matrix is not diagonally dominant, then the spectral radius of the matrix might be less than unity.

2.2 Simulation of Interconnect Circuits Using Waveform Relaxation and Transverse Partitioning

As shown in Section 2.1, using WR the differential equation representing the circuit is first decoupled into independent equations. Alternatively, the partitioning can be applied to the circuit directly to partition it into small subcircuits. The coupling effects between the subcircuits are represented using voltage/current sources, which are referred to in the literature as relaxation sources [42, 43]. An initial guess for these sources are assumed and each subcircuit is then simulated independently. The coupling effect between the subcircuits is then used to update the relaxation sources and the iterations are resumed.

In the case of high-speed interconnects, a longitudinal partitioning can be used to separate the interconnects circuits from the terminal devices attached to it. This partitioning scheme is known in the literature as the Longitudinal Partitioning [44–46]. However, this partitioning does not guarantee the convergence of the WR algorithm because the original circuit must be partitioned such that the iteration matrix $\mathbf{K}(s)$ is diagonally dominant (Lemma 2). Using longitudinal partitioning there is no guar-

antee that the resulting iteration matrix is diagonally dominant, hence, it is difficult to achieve convergence using the longitudinal partitioning approach.

To address the issue of convergence in Longitudinal Partitioning, WR-TP was proposed in [1,37] as an alternative method to the longitudinal partitioning. It was proved that the number of iterations required for convergence in the transverse partitioning is very few compared to the longitudinal partitioning algorithm. Using WR-TP, the coupled TLs are partitioned in the transverse direction into single lines and each line is simulated separately. The key enabling idea of WR-TP is that, in most of the applications, the coupling voltages/currents between the lines are small compared to the actual voltages/currents waveforms traveling through the lines, therefore, in this case, the WR-TP results in a diagonally dominant iteration matrix.

Typically, any interconnect circuit, which consists of m coupled lines, can be represented by a set of partial differential equations known as Telegrapher's equations [47] given as

$$\frac{\partial}{\partial z} \mathbf{v}(z, t) = -\bar{\mathbf{R}} \mathbf{i}(z, t) - \bar{\mathbf{L}} \frac{\partial}{\partial t} \mathbf{i}(z, t) \quad (2.10)$$

$$\frac{\partial}{\partial z} \mathbf{i}(z, t) = -\bar{\mathbf{G}} \mathbf{v}(z, t) - \bar{\mathbf{C}} \frac{\partial}{\partial t} \mathbf{v}(z, t) \quad (2.11)$$

where $\bar{\mathbf{R}} \in \mathbb{R}^{m \times m}$, $\bar{\mathbf{L}} \in \mathbb{R}^{m \times m}$, $\bar{\mathbf{G}} \in \mathbb{R}^{m \times m}$, $\bar{\mathbf{C}} \in \mathbb{R}^{m \times m}$ are the **RLGC** p.u.l parameter matrices of the coupled lines, $\mathbf{v}(z, t)$ and $\mathbf{i}(z, t)$ are the voltages and currents of the lines as a function of distance z and time t .

The equation of the j^{th} line can be written as [1]

$$\frac{\partial}{\partial z} v_j(z, t) = -\bar{R}_{jj} i_j(z, t) - \bar{L}_{jj} \frac{\partial}{\partial t} i_j(z, t) + h_j(z, t) \quad (2.12)$$

$$\frac{\partial}{\partial z} i_j(z, t) = -\bar{G}_{jj} v_j(z, t) - \bar{C}_{jj} \frac{\partial}{\partial t} v_j(z, t) + f_j(z, t) \quad (2.13)$$

where $\bar{R}_{ij}, \bar{L}_{ij}, \bar{G}_{ij}$ and $\bar{C}_{ij}$ are the elements at the i^{th} row and j^{th} column of $\bar{\mathbf{R}}, \bar{\mathbf{L}}, \bar{\mathbf{G}}$ and $\bar{\mathbf{C}}$, respectively, and $h_j(z, t), f_j(z, t)$ are the coupling effects from the neighboring lines and they are defined as

$$h_j(z, t) = - \sum_{\substack{k=1 \\ k \neq j}}^N \left(\bar{R}_{jk} i_k + \bar{L}_{jk} \frac{\partial i_k}{\partial t} \right) \quad (2.14)$$

$$f_j(z, t) = - \sum_{\substack{k=1 \\ k \neq j}}^N \left(\bar{G}_{jk} v_k + \bar{C}_{jk} \frac{\partial v_k}{\partial t} \right) \quad (2.15)$$

where $i_k = i_k(z, t)$ and $v_k = v_k(z, t)$.

In (2.12) and (2.13), the only unknown variables are the voltages and the currents of the j^{th} line (assuming that $h_j(z, t)$ and $f_j(z, t)$ are known). Therefore, the coupled partial differential equations in (2.12) and (2.13) can be decoupled using the waveform relaxation algorithm as follows [1]

$$\frac{\partial}{\partial z} v_j^{(r+1)}(z, t) = -\bar{R}_{jj} i_j^{(r+1)}(z, t) - \bar{L}_{jj} \frac{\partial}{\partial t} i_j^{(r+1)}(z, t) + h_j^{(r)}(z, t) \quad (2.16)$$

$$\frac{\partial}{\partial z} i_j^{(r+1)}(z, t) = -\bar{G}_{jj} v_j^{(r+1)}(z, t) - \bar{C}_{jj} \frac{\partial}{\partial t} v_j^{(r+1)}(z, t) + f_j^{(r)}(z, t) \quad (2.17)$$

where r represents the r^{th} iteration.

The coupling effects between the lines can be represented by voltage and cur-

rent sources attached to each line (Henceforth, referred to as WR-sources). Two approaches have been proposed in [1] for computing the WR-sources. The first approach is the single-ended representation, where the coupling effects are represented by voltage and current sources that are located at the end of each line. The second approach is the distributed representation. In this representation the coupling effects are represented by voltage and current sources that are distributed throughout the decoupled lines and are connected at the end of each lumped segment. Details of both approaches are provided in the following sections.

2.2.1 Single-ended Representation

Single-ended representation has been proposed in [1] to represent the coupling between lines. The coupling effects given in (2.14) and (2.15) can be represented in the frequency-domain at the r^{th} iteration as [1]

$$\begin{bmatrix} h_j^{(r)}(d, s) \\ f_j^{(r)}(d, s) \end{bmatrix} = \int_0^d e^{F_j(d-\lambda)} \Psi_j^{(r)}(\lambda, s) d\lambda \quad (2.18)$$

where

$$F_j = \begin{bmatrix} 0 & -\bar{R}_{jj} - s\bar{L}_{jj} \\ -\bar{G}_{jj} - s\bar{C}_{jj} & 0 \end{bmatrix} \quad (2.19)$$

$$\Psi_j^{(r)}(z, s) = \sum_{\substack{k=1 \\ k \neq j}}^N \begin{bmatrix} 0 & -\bar{R}_{jk} - s\bar{L}_{jk} \\ -\bar{G}_{jk} - s\bar{C}_{jk} & 0 \end{bmatrix} \begin{bmatrix} V_k^{(r)}(z, s) \\ I_k^{(r)}(z, s) \end{bmatrix} \quad (2.20)$$

Equation (2.21) represents the basis for updating the WR-sources at the r^{th} iter-

Figure 2.1: Single-ended representation of the WR-sources [1]

ation. Using (2.18-2.21), the coupling effects can be modeled as voltage and current sources connected to the far end of each line, as depicted in Figure 2.1. It is to be noted that the WR-sources in (2.18) can be calculated analytically using any integration formula, such as trapezoidal rule.

$$\begin{bmatrix} h_j^{(r)}(z + \Delta z, s) \\ f_j^{(r)}(z + \Delta z, s) \end{bmatrix} = e^{F_j \Delta z} \begin{bmatrix} h_j(z, s) \\ f_j(z, s) \end{bmatrix} + \frac{\Delta z}{2} [e^{\Delta z} \Psi_j(z, s) + \Psi_j(z + \Delta z, s)] \quad (2.21)$$

2.2.2 Distributed Sources Representation

In the distributed sources representation [1], the relaxation sources are added after each lumped section of the transmission line macromodel. In this representation, the integral in (2.18) is discretized as follows

Figure 2.2: Distributed sources added after each lumped section of the j^{th} line [1]

$$\tilde{h}_j(z_n, t) = -w_n \Delta z_n \sum_{k=1}^N \left(\bar{R}_{jk} i_k(z_n, t) + \bar{L}_{jk} \frac{\partial}{\partial t} i_k(z_n, t) \right) \quad (2.22)$$

$$\tilde{f}_j(z_n, t) = -w_n \Delta z_n \sum_{k=1}^N \left(\bar{G}_{jk} v_k(z_n, t) + \bar{C}_{jk} \frac{\partial}{\partial t} v_k(z_n, t) \right) \quad (2.23)$$

where w_n is a weighting factor, m is the number of discretization points (i.e. number of sections), z_n is the distance from the near end of the line and $\Delta z_n = z_n - z_{n-1}$.

It was shown in [1] that (2.22) and (2.23) can be represented as voltage and current sources attached at distance z_n . Figure 2.2 shows the distributed source approach used with conventional lumped segmentation macromodel. Algorithm 2.1 details the steps for applying waveform relaxation and transverse partitioning to the coupled interconnect structure [1].

Algorithm 2.1: Waveform Relaxation and Transverse Partitioning

input : $\bar{R}, \bar{L}, \bar{G}, \bar{C}$ parameters of the lines

- 1 Set $iter \leftarrow 0$;
 - 2 **while** *No Convergence* **do**
 - 3 Simulate each line separately. ;
 - 4 Update the WR-sources using (2.21) in case of single-ended representation or (2.22) and (2.23) in case of distributed sources approach. ;
 - 5 $iter \leftarrow iter + 1$
-

2.3 Summary

In this chapter, an overview of the waveform relaxation and transverse partitioning is given. In the transverse partitioning approach, the coupled lines are first partitioned in the transverse direction into single lines. The coupling effects due to the neighboring lines are represented by voltage/current sources. Next, each line is simulated separately and WR-sources are updated using the voltages/currents of each single line. This process is repeated until convergence is obtained.

In the next chapter, an overview of time-domain integration methods and related issues as relevant to this thesis proposal is given.

Chapter 3

Overview of Integration Methods used in Transient Simulation of High-Speed Circuits

Any electric circuit can be described using a set of Differential Algebraic Equations (DAEs), known as the modified nodal analysis (MNA) equations [3]. These equations belong to a set of problems known as Initial Value Problem (IVP). In order to find the time-domain response of the circuit, an integration method (also known as DAE-solver) is used to numerically solve the MNA equations at discrete time-points.

Typically, numerical methods start by the already calculated DC operating point $x(t_0 = 0)$ to find the solution at $t_1 = t_0 + h$ using $x(t_0)$. Once $x(t_1)$ is calculated, the numerical method moves in time and calculate the solution $x(t_2 = t_1 + h)$ ¹. This process is repeated until the final time point defined by the user is reached. In this chapter, we present an overview of different integration methods that are commonly used for solving IVPs. The convergence and stability properties of each method are also discussed.

¹We assume here that a constant step has been used. However, a variable step size can be used to guarantee more accurate results. More details can be found in [3].

In general, any IVP can be written in the following form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t)) \quad (3.1)$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \quad (3.2)$$

where $\mathbf{x}$ is the vector of unknown variables, $\mathbf{x}_0$ is given and t is an independent variable that represents the time.

The approximate solution obtained by the numerical integration method is denoted by x_n at time $t = t_n$, with $n = 0, 1, \dots$. At $n = 0$, the initial condition x_0 is specified and an approximate solution for $\mathbf{x}_1$ is to be calculated. One of the simplest methods to find the approximate solution $\mathbf{x}_1$ is Forward-Euler method. Using Forward-Euler, $\mathbf{x}_1$ can be approximated by assuming a straight line tangent, with slope $\dot{\mathbf{x}}_0$, which connects $\mathbf{x}_0$ with $\mathbf{x}_1$ (depicted in Figure 3.1). Thus, $\mathbf{x}_1$ can be found using the previous known point $\mathbf{x}_0$ as

Figure 3.1: Using Forward Euler to approximate $x(t_1)$

where $h = t_{n+1} - t_n$.

$$\mathbf{x}_1 = \mathbf{x}_0 + h\dot{\mathbf{x}}_0 \quad (3.3)$$

Generalizing (3.3) to find $\mathbf{x}_{n+1}$, given that $\mathbf{x}_n$ is known, results in

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\dot{\mathbf{x}}_n \quad (3.4)$$

A similar formula can also be obtained by assuming that x_1 is obtained from x_0 and the derivative $\dot{x}_1$. This results in the well known Backward-Euler formula, which is given as

$$\mathbf{x}_1 = \mathbf{x}_0 + h\dot{\mathbf{x}}_1 \quad (3.5)$$

Similarly, (3.5) can be generalized for any point $\mathbf{x}_{n+1}$ as

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\dot{\mathbf{x}}_{n+1} \quad (3.6)$$

3.1 Properties of Numerical Integration Methods

The key properties for any numerical method are Convergence, Truncation Error and Stability. Convergence determines how close is the numerical solution $\mathbf{x}_n$ to the analytical solution $\mathbf{x}(t)$. Truncation Error is the difference between the analytical solution and the numerical solution for a specific step size h , while Stability is the ability of the method to damp out the errors. This section gives an overview of these

three key properties.

3.1.1 Convergence and Order of the Integration Method

An integration method is convergent if the approximate solution x_n approaches the exact solution as the step size h decreases. Typically, for any numerical method, it is required that

$$\lim_{h \rightarrow 0} \left(\max_{n=0,1,\dots} \|x_n - x(t_n)\| \right) = 0 \quad (3.7)$$

Therefore, it is expected that for a convergent method, decreasing the step size h results in more accurate results. However, an important question is; how fast is the convergence. The answer for this question is the order of the method.

A method is said to be of order p if

$$\|x_n - x(t_n)\| \leq Ch^{p+1} \quad (3.8)$$

where C is called the error constant.

Typically, the error of a given integration method is defined using the so-called Local Truncation Error (LTE), which is defined as

$$LTE = Ch^{p+1} \frac{d^{p+1}x(t)}{dt^{p+1}} \Big|_{t=t_n} \quad (3.9)$$

The LTE is usually found by assuming that the integration method starts from an exact initial point (i.e. $\mathbf{x}_n = \mathbf{x}(t_n)$) and then calculating

$$\|x_{n+1} - x(t_{n+1})\| = LTE \quad (3.10)$$

3.1.2 Stability of the Integration Method

Another important property of the integration method is stability. In general, the integration method is required to be stable for all stable circuits. The stability is studied by applying the integration method to a scalar test equation (*sometimes referred to in the literature as Dalquist test equation*) given by [48]

$$\dot{x} = \lambda x \quad (3.11)$$

$$x(0) = x_0 \quad (3.12)$$

where λ is a complex constant and $\Re(\lambda) < 0$.

The test equation in (3.11) has an exact solution given by

$$x(t_n) = x(0)e^{\lambda t_n} \quad (3.13)$$

which is an exponentially decaying waveform if $\Re(\lambda) < 0$. Therefore, the numerical solution of the integration method should also be decaying with time, i.e. the numerical solution should be in the form of

$$|x_0| > |x_1| > |x_2| \dots \quad (3.14)$$

Using the test equation in (3.11) leads to the definition of A-stability in the sense of Dalquist

Definition 1. *A numerical method is called A-stable if the numerical solution $x_n \rightarrow 0$ as $n \rightarrow \infty$ when it is applied to the test equation in (3.11).*

Typically, the stability properties of any integration method is determined by examining the stability function $R(z) = \frac{x_{n+1}}{x_n}$, where $z = h\lambda$. If $|R(z)| < 1$, $\forall h, \Re(\lambda) < 0$, then the numerical solution would satisfy the A-stability criteria mentioned in Definition 1.

Next, we define the *stability domain* of the numerical method by using the set

$$\mathbb{S} = \{z \in \mathbb{C}; |R(z)| \leq 1\} \quad (3.15)$$

Using $R(z)$ we can define the A-stability properties using the following definition [49],

Definition 2. *An integration method is A-stable if its region of absolute stability contains the left half plane, i.e.*

$$\mathbb{C}^- \subset \mathbb{S} \quad (3.16)$$

Another important theory in analyzing the stability properties of the numerical method is the order-star [50], which is defined as

Definition 3. *The set $\{z \in \mathbb{C}; |R(z)| > |e^z|\}$ is called the order star of R .*

A numerical method is A-stable if and only if its stability function has no poles in the left-hand plane and its order star contains no purely imaginary numbers [50].

***L*-Stability**

L-stability is one of the most important properties when dealing with ODE integration methods. If the method is not *L*-stable it can result in ringing effects when a large step size is used.

Definition 4. *A method is called *L*-stable if it is *A*-stable and $\lim_{z \rightarrow \infty} R(z) = 0$ [50].*

3.2 Single Step Methods

One of the popular group of integration methods, used in circuit simulation, is the single step methods. In this group of methods, the value $\mathbf{x}_{n+1}$ is calculated by using the information only from the previous point $\mathbf{x}_n$. It is worth highlighting that the single step methods are special case of a general group of methods known as the Linear Multi Step (LMS) methods. This group uses information from not only the previous step, but also from many previous points (i.e. $\mathbf{x}_n, \mathbf{x}_{n-1}, \dots, \mathbf{x}_0$). However, it was proved by Dalquist [51] that *A*-stable LMS methods can not have orders higher than two. In this section, we focus on some of the single-step methods by examining their convergence and stability properties.

3.2.1 Forward/Backward Euler

The Forward Euler (FE) method is given by

$$x_{n+1} = x_n + hx_n^{(1)} \quad (3.17)$$

where $x_n^{(1)} = \frac{d}{dt}x(t)$ at $t = t_n$.

As seen from (3.17), FE is an explicit method (*i.e.* x_{n+1} is approximated using only information from the previous point x_n). Applying FE to the test equation in (3.11) leads to a stability function of FE in the form of $R^{FE}(z) = 1 + z$. Thus, the region of stability of FE is

$$S^{FE} = \{z \in \mathbb{C}; |1 + z| < 1\} \quad (3.18)$$

Figure 3.2 shows the stability region of FE. As seen $\mathbb{C}^- \not\subset S^{FE}$ and therefore FE is not A-stable, neither L-stable. In Fact, it was proved by Dalquist that all LMS-based explicit methods can not be A-stable.

Figure 3.2: Stability region of FE

On the other hand, Backward Euler (BE) method is given by

$$x_{n+1} = x_n + hx_{n+1}^{(1)} \quad (3.19)$$

Similarly, applying BE to the test equation in (3.11) gives a stability function $R^{BE}(z) = \frac{1}{1-z}$. Figure 3.3 shows the stability region of BE. As seen $\mathbb{C}^- \subset S^{BE}$ and therefore BE is an A-stable method.

Figure 3.3: Stability region of BE

Also, BE is an L-stable method because $\lim_{z \rightarrow \infty} R^{BE}(z) = 0$, which satisfies the L-stability condition in Definition 4.

3.2.2 Trapezoidal Rule

The TR method is given as

$$x_{n+1} = x_n + \frac{h}{2}(x_n^{(1)} + x_{n+1}^{(1)}) \quad (3.20)$$

Applying TR to the test equation in (3.11) results in a stability function $R^{TR}(z) = \frac{1 + \frac{z}{2}}{1 - \frac{z}{2}}$. Thus, the region of stability of TR is

$$S^{TR} = \{z \in \mathbb{C}; \frac{1 + \frac{z}{2}}{1 - \frac{z}{2}} < 1\} \quad (3.21)$$

Figure 3.4 shows the stability region of FE. As seen $\mathbb{C}^- \subset S^{TR}$ and therefore TR is an A-stable method.

On the other hand, if we use L-stability in Definition 4 on the TR method, whose stability function $R^{TR}(z)$, we find that $\lim_{z \rightarrow \infty} R^{TR}(z) = -1$. Therefore, the TR might result in a ringing phenomena when a large step-size h is used. In order to demonstrate

Figure 3.4: Stability region of TR

Figure 3.5: Ringing of TR when applied to $\dot{x}(t) = -200x(t) + 200\cos(t)$, $x(0) = 0$

this effect, we consider solving the following differential equation with two methods:

Backward -Euler (BE) and Trapezoidal-Rule (TR), with a step size ($h = 0.1$ sec).

$$\dot{x}(t) = -200x(t) + 200\cos(t), \quad x(0) = 0 \quad (3.22)$$

Figure 3.5 shows the solution obtained by both methods. As seen, there is a ringing in the solution obtained using TR method.

3.3 Linear Multi-Step (LMS) Methods

One of the major classes of integration methods are the LMS methods. In the LMS methods, x_{n+1} is approximated not only by the previous point x_n but also using previous points x_{n-i} , $i = 0, 1, 2, \dots$. In fact, the single step methods presented in the previous section are special cases of the LMS methods when $i = 0$. This section presents a brief overview of the LMS methods.

The general formula of a linear k -step method is given by [2]

$$\sum_{i=0}^k a_i x_{n+k-i} = h \sum_{i=0}^k b_i x_{n+k-i}^{(1)} \quad (3.23)$$

where the coefficients a_i , b_i are specific to each integration method. These coefficients determine the order of the method. For example, if the required order is p and k -steps are used, then a_i and b_i must be chosen to make the first $p+1$ Taylor series coefficients of $\mathcal{M}(x(t)) := \sum_{i=0}^k a_i x_{n+k-i}$ match the first $p+1$ coefficients of $\mathcal{N}(x(t)) := h \sum_{i=0}^k b_i x_{n+k-i}^{(1)}$. For example, if $k = 1$ and $b_0 = 0$, a first order method ($p = 1$) and explicit LMS method can be written using (3.23) as

$$a_0 x_{n+1} + a_1 x_n = h b_1 x_n^{(1)} \quad (3.24)$$

In order to find a_0 , a_1 and b_1 , we set $x_n = x(t_n)$, $x_{n+1} = x(t_{n+1})$ and $\dot{x}_n = \frac{d}{dt}x(t_n)$. Next, $\mathcal{M}(x(t))$ and $\mathcal{N}(x(t))$ are expanded in their Taylor series and the terms up to the p^{th} derivatives are equated. This process yields $a_0 = 1$, $a_1 = -1$ and $b_1 = 1$, which is equivalent to the FE method in (3.17).

Figure 3.6: Stability region of 2-steps and 3-steps Adams-Moulton LMS method [2]

Similarly, a third order 2-steps method, known as implicit Adams-Moulton method [2] can be written as

$$x_{n+1} = x_n + h \left(\frac{5}{12} x_{n+1}^{(1)} + \frac{8}{12} x_n^{(1)} - \frac{1}{12} x_{n-1}^{(1)} \right) \quad (3.25)$$

Although LMS methods can provide very a high-order approximation, they can not provide A -stable methods with orders more than two. It was proved by Dalquist that all explicit LMS methods are not A -stable (*known in the literature as the first Dalquist barrier*). In addition, the highest order and A -stable LMs method is of order 2, which is the TR method (*known in the literature as the second Dalquist barrier*). As an example of the stability region of LMS methods, Figure 3.6 shows the absolute stability of the implicit Adams-Moulton shown in (3.25).

3.3.1 Runge-Kutta Methods

The Runge-Kutta (RK) methods are known in the literature as single-step multi-stage methods. The single step RK does not use information from previous time steps, but uses the information from intermediate stages within the current time step. A Runge-Kutta method that is used to solve ODE in (3.1) can be defined as

$$k_i = x_n + h \sum_{j=1}^s a_{ij} (f(t_{n-1} + c_i h, k_j)), \quad j = 1, \dots, s \quad (3.26)$$

$$x_{n+1} = x_n + h \sum_{i=1}^s b_i k_i \quad (3.27)$$

where s is the number of stages, $a_{i,j}$, c_i and b_i are determined using the Butcher-tableau, defined as [49]

c_1	a_{11}	a_{12}	$\dots$	a_{1s}
c_2	a_{21}	a_{22}	$\dots$	a_{2s}
c_3	a_{31}	a_{32}	$\dots$	a_{3s}
$\vdots$	$\vdots$		$\ddots$	
c_s	a_{s1}	a_{s2}	$\dots$	$a_{s,s-1}$
	b_1	b_2	$\dots$	b_s

A RK method is explicit if $a_{ij} = 0 \quad \forall \quad j > i$. Unfortunately, all explicit RK methods are not A-stable [49], and thus, their usage is limited. Therefore, implicit RK methods are usually used instead of explicit RK methods. Furthermore, implicit RK can achieve higher orders than explicit RK for the same number of stages. It was shown

that the order of implicit RK methods can be equal up to $2s$, where s is the number of stages [49]. However, implicit RK formulas require an iterative method, such as Newton-Raphson (NR), to solve the resulting non-linear equations, whereas explicit RK methods are not challenged with this difficulty.

Using the RK formula, most of the previously discussed popular integration methods can be derived. For example, the BE and TR can be obtained by using the Butcher tableau shown in Figure 3.7

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

(a)

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

(b)

Figure 3.7: Butcher tableau for (a)BE, (b)TR

Another important group of implicit RK methods are the Gauss methods [49]. This group of methods has an order that is equal to $2s$. For example, a two-stage method, with order 4, can be constructed using the following Butcher tableau

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Also, a 3-stage RK method, which provides a 6th order integration method, can be defined using the following Butcher Tableau [49]

$\frac{1}{2} - \frac{\sqrt{15}}{10}$	$\frac{5}{36}$	$\frac{2}{9} - \frac{\sqrt{15}}{15}$	$\frac{5}{36} - \frac{\sqrt{15}}{30}$
$\frac{1}{2}$	$\frac{5}{36} + \frac{\sqrt{15}}{24}$	$\frac{2}{9}$	$\frac{5}{36} - \frac{\sqrt{15}}{24}$
$\frac{1}{2} + \frac{\sqrt{15}}{10}$	$\frac{5}{36} + \frac{\sqrt{15}}{30}$	$\frac{2}{9} + \frac{\sqrt{15}}{15}$	$\frac{5}{36}$
	$\frac{5}{18}$	$\frac{4}{9}$	$\frac{5}{18}$

In order to study the stability properties of implicit RK methods, we first apply it to the test equation in (3.11) and construct the stability function $R(z)$. It was shown that the stability function of the RK methods is given by [49]

$$R(z) = 1 + z\mathbf{b}^T(1 - z\mathbf{A})^{-1}\mathbf{q} \quad (3.28)$$

where $\mathbf{q} = [1, \dots, 1]^T$.

In the case of implicit methods, the stability function can be represented using a rational function as

$$R(z) = \frac{P(z)}{Q(z)} \quad (3.29)$$

It was shown by Ehle [52] that implicit RK methods of order $2s$ are all A-stable (i.e. $|R(z)| < 1$).

Although RK methods can provide high-order convergence without losing stability, they suffer from some drawbacks that limit their usage in circuit simulation. One of the main drawbacks of RK methods is the high cost required to solve the resulting equations, which are strongly coupled. For example, consider a system of N differential equations that is solved using an s -stage RK method. In this case, an iterative Newton-Raphson method is used to solve for the s -stages (k_i). The r^{th} iteration can

be written as

$$\mathbf{A} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_s \end{bmatrix} = - \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \vdots \\ \mathbf{R}_s \end{bmatrix} \quad (3.30)$$

where $\zeta_i = k_i^{r+1} - k_i^r$, $\mathbf{R}_i = k_i^r - x_n - h \sum_{j=1}^s f(k_j^r)$ and $\mathbf{A} \in \mathbb{R}^{sN \times sN}$ is defined as

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} - ha_{11}\mathbf{J}_1 & -ha_{12}\mathbf{J}_2 & \dots & -ha_{1s}\mathbf{J}_s \\ -ha_{21}\mathbf{J}_1 & \mathbf{I} - ha_{22}\mathbf{J}_2 & \dots & -ha_{2s}\mathbf{J}_s \\ \vdots & \dots & \ddots & \vdots \\ -ha_{s1}\mathbf{J}_1 & -ha_{s1}\mathbf{J}_1 & \dots & \mathbf{I} - ha_{ss}\mathbf{J}_1 \end{bmatrix} \quad (3.31)$$

with $J_i = \frac{df}{dk_i}$.

Solving (3.30) is done using LU factorization followed by Forward/Backward (F/B) substitution. Since the matrix $\mathbf{A}$ contains strongly coupled s equations, each of size N , the cost of LU factorization and F/B substitution would be approximately $O(sN^3)$.

3.4 Multi-derivatives Methods

The Multi-derivatives method is a class of a single-step integration methods. In this class of methods, each point is approximated by using the value of the previous time-point as well as its derivatives as follows

$$\sum_{i=0}^m \alpha_i h^i x_n^{(i)} = \sum_{i=0}^k \beta_i h^i x_{n+1}^{(i)} \quad (3.32)$$

where m and k are integers, which determine the order of convergence, α_i and β_i are the coefficients of the ObF and they can be determined by assuming that $x_{n+1}^{(i)}$ are the exact values (i.e. $x_{n+1} = \frac{d^i x(t)}{dt^i}|_{t_{n+1}}$) and then substituting (3.32) in the Taylor series expansion. It can be shown that [14]

$$\alpha_i = (-1)^i \frac{(k-m-i)!m!}{(k+m)!i!(m-i)!} \quad , \quad i = 0, \dots, m \quad (3.33)$$

$$\beta_i = \frac{(k+m-i)!k!}{(k+m)!i!(k-i)!} \quad , \quad i = 0, \dots, k \quad (3.34)$$

(3.32) is also known in the literature as Obreshkov-Formula (ObF). In this thesis proposal, we are going to adopt the ObF formula and show that it can be used to accurately find the time-domain response of high-speed interconnect circuits..

Similar to RK methods, the stability properties of the ObF are examined by examining the stability function $R_{(k,m)}(z)$ of (3.32) corresponding to specific values of m and k . The stability function $R_{(k,m)}(z)$ can be found by finding $\frac{x_{n+1}}{x_n}$ using (3.32).

It was noted by Ehle [52] that $R_{(k,m)}(z)$ is the Padè approximation to the exponential function of order $m+k$, more precisely

$$e^z - R_{(k,m)}(z) = \underbrace{\frac{(-1)^m m! k!}{(m+k)!(m+k+1)!}}_{C_{k,m}} z^{m+k+1} + O(z^{k+m+1}) \quad (3.35)$$

From (3.35), it can be seen that the order of ObF is given by the order of Padè approximation $R_{(k,m)}(z)$, and thus the order of convergence of ObF is $k + m$, more precisely

$$\|x_n - x(t_n)\| \leq C_{k,m} O(h^{k+m+1}) \quad (3.36)$$

Moreover, it was also shown in [52] that the Padè coefficients satisfy the A-stability criteria, i.e. $|R_{(k,k)}(z)| \leq 1, \forall h, \text{Re}(\lambda) \leq 0$. As a result, the ObF is A-stable for all $m = k$. In addition, it was also proved in [14] that the ObF provides an L-stable methods if $k - 2 \leq m \leq k$.

3.5 Transient Analysis of Electric Circuits Using Obreshkov-Formula

This section provides an overview of using the high-order ObF for time-domain analysis of general circuits. Consider a general circuit described in the time-domain using the Modified Nodal Analysis approach (MNA) [3] as

$$\mathbf{G}\mathbf{x}(t) + \mathbf{C}\frac{d\mathbf{x}(t)}{dt} + \mathbf{f}(\mathbf{x}(t)) = \mathbf{b}(t) \quad (3.37)$$

where

- $\mathbf{G}$ and $\mathbf{C} \in \mathbb{R}^{N \times N}$ are matrices describing the memory and memoryless elements in the circuit, respectively,

- $\mathbf{f}(\mathbf{x}(t)) \in \mathbb{R}^N$ is a vector that contains the nonlinear currents flowing in nonlinear resistors and nonlinear charges across nonlinear capacitors,
- $\mathbf{x}(t) \in \mathbb{R}^N$ is the vector of unknown node voltage and current waveforms,
- $\mathbf{b}(t) \in \mathbb{R}^N$ is the vector of independent stimuli to the circuit,
- N is the total number of unknown voltages/currents.

The ObF-based method aims at approximating the solution of $\mathbf{x}(t)$ at $t = t_{n+1}$, which is denoted by $\mathbf{x}_{n+1}$ by using the solution $\mathbf{x}_n$ at the previous time point $t = t_n$. Applying ObF in (3.32) to numerically approximate the transient solution of the MNA equations in (3.37) at $t = t_{n+1}$ results in the following augmented system of equations [16]

$$\tilde{\mathbf{G}}\boldsymbol{\xi}_{n+1} + \tilde{\mathbf{C}}\boldsymbol{\xi}_{n+1} + \tilde{\boldsymbol{\eta}}_{n+1} = \tilde{\mathbf{b}}_{n+1} \quad (3.38)$$

where $\tilde{\mathbf{G}}$ and $\tilde{\mathbf{C}} \in \mathbb{R}^{Nk \times Nk}$ are block structured matrices in which a block at the (i, j) entry, (with $i, j = 1, \dots, N$), is a $k \times k$ matrix whose structure depends on the nature of the i^{th} and j^{th} variable in the MNA formulation. A block in $\tilde{\mathbf{G}}$, denoted by $[\tilde{\mathbf{G}}]_{(i,j)}$, is a diagonal block of form

$$[\tilde{\mathbf{G}}]_{(i,j)} = g_{i,j} \mathbf{I}_k \quad (3.39)$$

where $g_{i,j}$ is the value of the conductance connecting nodes i and j , and $\mathbf{I}_k$ is the

identity matrix of size $k \times k$.

On the other hand, a block in $\tilde{\mathbf{C}}$ is given by

$$[\tilde{\mathbf{C}}]_{(i,j)} = \frac{c_{i,j}}{h} \mathbf{T} \quad (3.40)$$

where h is the size of the time step, i.e. $h = t_{n+1} - t_n$, and $c_{i,j}$ is the value of the capacitance connecting nodes i and j , and $\mathbf{T} \in \mathbb{R}^{k \times k}$ is given by

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 \\ -\frac{\alpha_0}{\alpha_k} & -\frac{\alpha_1}{\alpha_k} & \dots & -\frac{\alpha_{k-1}}{\alpha_k} \end{bmatrix} \quad (3.41)$$

where

$$\alpha_i = (-1)^{(i)} \frac{(m+k-i)!k!}{i!(m+k)!(k-i)!} \quad (3.42)$$

If the i^{th} or j^{th} variables of the MNA formulation represent currents, such as those in inductors or voltage sources, then the above blocks will have the same structure but different values of $g_{i,j}$ and $c_{i,j}$.

The vector $\tilde{\boldsymbol{\eta}}_{n+1}$ results from the discretization of the circuit nonlinearities, and is described by [14]

$$\tilde{\boldsymbol{\eta}}_{n+1} = [\tilde{\boldsymbol{\eta}}_{n+1,1}^\top \quad \dots \quad \tilde{\boldsymbol{\eta}}_{n+1,N}^\top]^\top, \quad (3.43)$$

where

$$\tilde{\boldsymbol{\eta}}_{n+1,i} = [\eta_{n+1,i}^{(0)} \quad \eta_{n+1,i}^{(1)} \quad \dots \quad \eta_{n+1,i}^{(k-1)}]^\top \quad (3.44)$$

and $\eta_{n+1,i}^{(j)} = h^j \frac{d^j}{dt^j} f_i(\mathbf{x}(t))$ computed at t_{n+1} .

The vectors $\boldsymbol{\xi}_{n+1}$ and $\tilde{\mathbf{b}}_{n+1}$ represent expanded versions of the vector of unknowns and independent stimuli, respectively. More precisely, these two vectors are also structured as a sequence of N subvectors, of k components each in the following manner,

$$\boldsymbol{\xi}_{n+1} = [\mathbf{x}_{n+1,1}^T \quad \dots \quad \mathbf{x}_{n+1,N}^T]^T \quad (3.45)$$

and

$$\tilde{\mathbf{b}}_{n+1} = [\tilde{\mathbf{b}}_{n+1,1}^T \quad \dots \quad \tilde{\mathbf{b}}_{n+1,N}^T]^T \quad (3.46)$$

where $\mathbf{x}_{n+1,i} \in \mathbb{R}^k$, $i = 1, \dots, N$ is given by

$$\mathbf{x}_{n+1,i} = [x_{n+1,i}^{(0)} \quad hx_{n+1,i}^{(1)} \quad \dots \quad h^{k-1}x_{n+1,i}^{(k-1)}]^T \quad (3.47)$$

with $x_{n+1,i}^{(j)}$ denoting the numerical approximation to the j^{th} -order derivative of the i^{th} component of the exact solution $\mathbf{x}(t)$ at $t = t_{n+1}$ and $\tilde{\mathbf{b}}_{n+1,i} \in \mathbb{R}^k$, $i = 1, \dots, N$, is given by

$$\tilde{\mathbf{b}}_{n+1,i} = [b_{n+1,i}^{(0)} \quad hb_{n+1,i}^{(1)} \quad \dots \quad h^{k-1}b_{n+1,i}^{(k-1)} - u_{n,i}]^T \quad (3.48)$$

with $b_{n+1,i}^{(j)}$ denoting the numerical approximation to the j^{th} -order derivative of the i^{th} component of the independent stimulus $\mathbf{b}(t)$ at $t = t_{n+1}$ and $u_{n,i}$ being the i^{th} component of the vector $\mathbf{u}_n \in \mathbb{R}^N$, computed from the previous solution obtained at

$t = t_n$ as follows

$$\mathbf{u}_n = \frac{\mathbf{C}}{h\alpha_k} \sum_{j=0}^m \beta_j h^j \mathbf{x}_n^{(j)} \quad (3.49)$$

$$\beta_i = \frac{(m+k-i)!m!}{i!(m+k)!(m-i)!} \quad (3.50)$$

with $\mathbf{x}_n^{(j)}$ is a vector that contains the j^{th} derivatives of all unknown voltages/currents in the MNA equations.

The integers k used in (3.41) and m used in (3.50) determine the order and the stability characteristic of the ObF in (3.32) [16].

3.5.1 Solution of the Augmented System

Solving (8.6) for $\boldsymbol{\xi}_{n+1}$ is done using an iterative Newton-Raphson (NR) approach, starting from an initial guess, $\boldsymbol{\xi}_{n+1}^{(0)}$, and iterating through

$$\boldsymbol{\xi}_{n+1}^{(r+1)} = \boldsymbol{\xi}_{n+1}^{(r)} + \Delta \boldsymbol{\xi}_{n+1}^{(r+1)} \quad (3.51)$$

where the update of the r^{th} iteration $\Delta \boldsymbol{\xi}_{n+1}^{(r+1)}$ is obtained from

$$\tilde{\mathbf{J}} \Delta \boldsymbol{\xi}_{n+1}^{(r+1)} = \boldsymbol{\Phi}^{(r)} \quad (3.52)$$

where $\tilde{\mathbf{J}}$ is the Jacobian matrix and

$$\boldsymbol{\Phi}^{(r)} = \tilde{\mathbf{G}} \boldsymbol{\xi}_{n+1}^{(r)} + \tilde{\mathbf{C}} \boldsymbol{\xi}_{n+1}^{(r)} + \tilde{\boldsymbol{\eta}}_{n+1}^{(r)} - \tilde{\mathbf{b}}_{n+1} \quad (3.53)$$

The Jacobian matrix $\tilde{\mathbf{J}}$ has a block structure form, similar to $\tilde{\mathbf{G}}$ and $\tilde{\mathbf{C}}$. A block

in $\tilde{\mathbf{J}}$ is given by

$$[\tilde{\mathbf{J}}]_{(i,j)} = [\tilde{\mathbf{G}}]_{(i,j)} + [\tilde{\mathbf{C}}]_{(i,j)} + \begin{bmatrix} r_0 & 0 & \dots & 0 \\ hr_1 & r_0 & 0 & \dots & 0 \\ \vdots & \dots & \dots & \ddots & \vdots \\ h^{k-1}(k-1)!r_{k-1} & \dots & \dots & \dots & r_0 \end{bmatrix} \quad (3.54)$$

where r_u ($u = 0, \dots, k-1$) are the Taylor coefficients of $\frac{1}{u!} \frac{\partial f_i}{\partial x_j} \big|_{t=t_{n+1}}$.

At each NR iteration, $\tilde{\mathbf{J}}$ is factorized using a block form of the LU factorization which was implemented using the industry standard KLU algorithm [53]. The block LU factorization can be described as a standard LU technique in which the scalar operations of division, multiplication and subtraction are replaced by their matrix counterparts on the $k \times k$ matrix blocks given by (3.54).

Typically, the block LU factorization is carried out in two steps. The first step is symbolic to determine the non-zero and expected fill-in patterns of the matrix $\mathbf{J} \in \mathbb{R}^{N \times N}$, whose structure mirrors the structure of $\tilde{\mathbf{J}}$, in the sense that an entry (i, j) in $\mathbf{J}$ is non-zero if, and only if, the block entry (i, j) in $\tilde{\mathbf{J}}$ is non-zero. It must be noted that, the matrix $\mathbf{J}$ has the same structure of the matrix arising from applying a conventional low-order method, such as TR.

The second step is numerical but with the exception that only the blocks of $\tilde{\mathbf{J}}$ that have been determined by fill-ins pattern [16], deduced in the first step, are used in the computation. The pseudo-code shown by Algorithm 6.1 illustrates the fundamental

Algorithm 3.1: Block-LU factorization

```

input  :  $\tilde{\mathbf{J}} \in \mathbb{R}^{kN \times kN}$ 
output:  $\tilde{\mathbf{L}}, \tilde{\mathbf{U}} \in \mathbb{R}^{kN \times kN}$ 
1 begin
2    $\mathbf{L} \leftarrow \mathbf{I}_{kN}$ ;
3   for  $p \leftarrow 1$  to  $N$  do
4      $r_1 \leftarrow 1, r_2 \leftarrow kN$ ;
5      $c_1 \leftarrow (p-1)k + 1, c_2 \leftarrow pk$ ;
6      $\mathbf{y} \leftarrow \text{BFS}(\mathbf{L}, \tilde{\mathbf{J}}(r_1:r_2, c_1:c_2))$ ;
7      $r_2 \leftarrow pk$ ;
8      $\mathbf{U}(r_1:r_2, c_1:c_2) \leftarrow \mathbf{y}(r_1:r_2, c_1:c_2)$  ;
9      $\mathbf{H} \leftarrow \text{BPivot}(\mathbf{y})$  ; //Choose the block pivot
10    for  $q \leftarrow p+1$  to  $N$  do
11       $r_1 \leftarrow (q-1)k, r_2 \leftarrow qk$ ;
12       $\mathbf{L}(r_1:r_2, c_1:c_2) \leftarrow \mathbf{y}(r_1:r_2, c_1:c_2) \mathbf{H}^{-1}$ 

```

computational operations of this step (the numerical step), where the BFS procedure at line 6 is illustrated by pseudo-code in Algorithm 3.2. Algorithm 3.3 describes the operation of block Forward/Backward substitution module.

It is worth highlighting that although the CPU-time required for solving the NR iteration in (8.8) is higher than the CPU-time required in the case of conventional low-order methods due to the fact that the scalar operation is extended to the $k \times k$ blocks $\mathbf{J}[i, j] k \times k$, this overhead is significantly offset by the savings in the number of time-points obtained from the high-order methods, because the high-order methods are able to increase the time step without losing accuracy nor stability.

3.6 Envelope-Following (EF) Method

The EF method is proposed in the literature as an efficient way for simulating highly oscillatory circuits. The waveforms of these circuits can be characterized as high-

Algorithm 3.2: BFS (Block Forward Substitution)

```

input  :  $\mathbf{L} \in \mathbb{R}^{kN \times kN}$ ,  $\mathbf{A} \in \mathbb{R}^{kN \times k}$ 
output:  $\mathbf{y} \in \mathbb{R}^{kN \times k}$ 
1 begin
2    $\mathbf{y} \leftarrow \mathbf{A}$ ;
3   for  $p \leftarrow 2$  to  $N$  do
4      $r_1 \leftarrow (p-1)k+1$ ,  $r_2 \leftarrow pk$ ;
5     for  $q \leftarrow 1$  to  $p-1$  do
6        $c_1 \leftarrow (q-1)k+1$ ,  $c_2 \leftarrow qk$ ;
7        $\mathbf{y}(r_1:r_2,:) \leftarrow \mathbf{y}(r_1:r_2,) -$ 
8          $\mathbf{L}(r_1:r_2, c_1:c_2) \mathbf{y}(c_1:c_2,)$ 

```

Algorithm 3.3: Block Forward/Backward Substitution

```

input  :  $\tilde{\mathbf{L}}, \tilde{\mathbf{U}} \in \mathbb{R}^{kN \times kN}$ ,  $\tilde{\mathbf{b}} \in \mathbb{R}^{kN \times k}$ 
output:  $\boldsymbol{\rho} \in \mathbb{R}^{kN \times k}$ 
1 begin
2    $\mathbf{y} \leftarrow \tilde{\mathbf{b}}$ ;
3   for  $p \leftarrow 2$  to  $N$  do
4      $r_1 \leftarrow (p-1)k+1$ ,  $r_2 \leftarrow pk$ ;
5     for  $q \leftarrow 1$  to  $p-1$  do
6        $c_1 \leftarrow (q-1)k+1$ ,  $c_2 \leftarrow qk$ ;
7        $\mathbf{y}(r_1:r_2,:) \leftarrow \mathbf{y}(r_1:r_2,) -$ 
8          $\mathbf{L}(r_1:r_2, c_1:c_2) \mathbf{y}(c_1:c_2,)$ 
9    $\boldsymbol{\rho} \leftarrow \mathbf{y}$ ;
10  for  $p \leftarrow N$  to  $1$  do
11     $r_1 \leftarrow (p-1)k+1$ ,  $r_2 \leftarrow pk$ ;
12    for  $q \leftarrow 1$  to  $p-1$  do
13       $c_1 \leftarrow (q-1)k+1$ ,  $c_2 \leftarrow qk$ ;
14       $\mathbf{D} \leftarrow \mathbf{U}(r_1:r_2, r_1:r_2)$ ;
15       $\boldsymbol{\rho}(r_1:r_2,:) \leftarrow \mathbf{D}^{-1}(\boldsymbol{\rho}(r_1:r_2,) -$ 
16         $\mathbf{U}(r_1:r_2, c_1:c_2) \boldsymbol{\rho}(c_1:c_2,))$ 

```

frequency components whose amplitudes are slowly varying. Example of these circuits are Phase-Locked Loops (PLLs), Voltage-Controlled Oscillators and fast switching circuits such as Switching Power Converters (SPC). In these situations, the transient response becomes equally important as that of the steady-state response. For example,

Figure 3.8: Illustration of the Envelope waveform

the designer might want to verify the behavior of an VCO at the startup/shutdown of the oscillator. The main idea of EF methods is to solve only for the slowly varying components of the signal by sampling the high-frequency signal at a low frequency rate. This sampling process results in a low-frequency waveform, known as the “envelope”. The main property of the envelope waveform is that it intersects with the original waveform at multiples of the period of the high frequency component, these points are referred to as envelope points. Figure 3.8 illustrates the basic idea of EF by depicting a highly oscillatory response and an associated envelope that is typically traced by BE or TR.. Thus, by solving only for these samples of the envelope, a very large step size is used, skipping many cycles of the high-frequency components.

Consider a nonlinear circuit represented using the MNA equations in (3.37). In order to give an overview of the EF method, we need to provide a more precise definition of the “envelope” waveform.

Definition 5. *The “envelope” of a highly-oscillatory waveform is a waveform that results from sampling the original waveform at time $t = mT$, where $m \in \mathbb{N}$ and T is*

the period of the high-frequency component in the original waveform.

The above definition makes it clear that the envelope of a given waveform is not unique, in the sense that different envelopes for the same waveform can be constructed depending on the multiples of T with which this waveform coincides with the original waveform, i.e., $\mathbf{y}(mT) = \mathbf{x}(mT)$, where $\mathbf{y}(t)$ is the envelope waveform.

The basic idea in the EF method is that the envelope of a highly oscillatory waveform satisfies a differential-like equation, whose solution can be approximated numerically using the standard numerical techniques used in the classical SPICE engine, such as the BE and the TR. The starting point of the EF is by assuming that the original circuit has been solved accurately for the m^{th} cycle, i.e., for $[mT, (m+1)T]$. Next, an integration method is used to find another point on the envelope that is several T apart from m^{th} cycle. To illustrate the basic idea, the Forward Euler (FE) method is used here to explicitly predict $\mathbf{y}(nT)$, $n > m$ as follows,

$$\mathbf{y}_n = \mathbf{y}_m + H \mathbf{y}_m^{(1)} \quad (3.55)$$

where $H = (n - m)T$, $\mathbf{y}_n = \mathbf{y}(nT)$, $\mathbf{y}_m = \mathbf{y}(mT)$ and $\mathbf{y}_m^{(1)} = \left. \frac{d\mathbf{y}(t)}{dt} \right|_{t=mT}$.

Since the EF method starts by the assumption that the envelope waveform is much slower than the original waveform, then the derivative $\mathbf{y}_m^{(1)}$ can be approximated using Finite-Difference method as [22, 54]

$$\mathbf{y}_m^{(1)} = \frac{\mathbf{y}_{m+1} - \mathbf{y}_m}{T} \quad (3.56)$$

where $\mathbf{y}_{m+1} = \mathbf{y}((m+1)T)$.

Substituting (3.56) in (3.55) results in a FE-based EF (FE-EF) method given by

$$\text{FE-EF} \Rightarrow \mathbf{y}_n = \mathbf{y}_m + H \left(\frac{\mathbf{y}_{m+1} - \mathbf{y}_m}{T} \right) \quad (3.57)$$

The FE-EF is an explicit method used to compute future envelope-point. In a similar way, an implicit EF method based on the BE or TR formulas can be used to approximate the envelope as follows [22, 54]

$$\text{BE-EF} \Rightarrow \mathbf{y}_n = \mathbf{y}_m + \frac{H}{T} (\mathbf{y}_{n+1} - \mathbf{y}_n) \quad (3.58)$$

$$\text{TR-EF} \Rightarrow \mathbf{y}_n = \mathbf{y}_m + \frac{H}{2} \left(\frac{\mathbf{y}_{n+1} - \mathbf{y}_n}{T} + \frac{\mathbf{y}_{m+1} - \mathbf{y}_m}{T} \right) \quad (3.59)$$

In contrast to the FE-EF in (3.55), in which $\mathbf{y}_n$ is computed explicitly, the EF formula in (3.58) and (3.59) are implicit and they require an iterative scheme, such as Newton-Raphson method [54], to solve for $\mathbf{y}_n$. In addition, it was proven in [22, 23] that BE and TR can provide stable methods when combined with EF techniques.

3.7 Summary

This chapter presented a detailed background on different integration methods that are used as ODE solvers. Stability and accuracy properties are also discussed to show the properties of these methods. Furthermore, properties and details of the ObF have been discussed, in addition to showing how the ObF can be used to solve the

MNA equations. Finally, a brief overview of the EF method, which is used to find the transient response of highly oscillatory circuits, is presented.

Chapter 4

Overlapping Partitioning Techniques for Transient Simulation of Strongly Coupled High-Speed Interconnects

In Chapter 2, transient analysis of high-speed interconnect circuits based on waveform relaxation and transverse partitioning (WR-TP) was provided. In this Chapter, to achieve further speed-up, a waveform relaxation with overlapping partitioning (OP) method for tightly coupled interconnects is presented. The new method exploits the multi-splitting partitioning algorithm for solving linear systems [8]. Using the proposed approach, the coupled interconnects are partitioned such that each subcircuit contains one or more lines that exist in an adjacent subcircuit. In addition, each subcircuit contains the lines that are strongly coupled, while the weak coupling between the adjoint subcircuits is represented using voltage/current sources attached to each line [25, 26].

4.1 Overview of Multi-Splitting Algorithm for Solving Linear Systems

In this section, a brief overview of multi-splitting partitioning is presented. Multi-splitting partitioning was first presented in [8, 55] as an iterative method for solving linear systems of equations. In this case, the original system can be partitioned into several subsystems, where the subsystems are allowed to overlap. In addition, multi-splitting partitioning performance and convergence have been studied in the literature [9, 10]. It has been shown that waveform relaxation techniques based on overlapping blocks yield faster convergence than traditional waveform relaxation [10]. In the rest of this section, a brief overview of the multi-splitting partitioning algorithm is provided.

Consider the linear system described in (2.1)

$$\dot{\mathbf{x}}(t) + \mathbf{A}\mathbf{x}(t) = \mathbf{b}(t), \quad \mathbf{x}(0) = \mathbf{x}_0 \quad (4.1)$$

Next, in order to obtain the system of multi-splitting matrices, we define the following [8, 10]:

Definition 6. *The set of triples $(\mathbf{M}_l, \mathbf{N}_l, \mathbf{E}_l)$ for $l = 1, \dots, S$ is called a multi-splitting of $\mathbf{A}$ if the following two conditions hold:*

- $\mathbf{A} = \mathbf{M}_l - \mathbf{N}_l$ for $l = 1 \dots S$, where S is the number of subsystems
-

- The matrices $\mathbf{E}_l$ (referred to as weighting matrices), are nonnegative diagonal matrices with diagonal elements $0 \leq (E_l)_{j,j} \leq 1$ and

$$\sum_{l=1}^S \mathbf{E}_l = \mathbf{U}_{n \times n} \quad (4.2)$$

where $\mathbf{U}_{n \times n} \in \mathbb{R}^{n \times n}$ is the identity matrix and the subscripts j, k are the element at the j^{th} row and k^{th} column of the corresponding matrix.

Using the multi-splitting matrices $(\mathbf{M}_l, \mathbf{N}_l, \mathbf{E}_l)$ and Definition 1, the linear system in (4.1) can be solved iteratively as follows [8, 56]

$$\begin{aligned} \dot{\mathbf{y}}_l^{(r+1)}(t) + \mathbf{M}_l \mathbf{y}_l^{(r+1)}(t) &= \mathbf{N}_l \mathbf{x}^{(r)}(t) + \mathbf{b}(t), \\ \mathbf{y}_l^{(r+1)}(0) &= \mathbf{x}_0, \quad l = 1, \dots, S \end{aligned} \quad (4.3)$$

where r represents the r^{th} iteration.

The iterative sequence in (4.3) is started by assuming an initial guess $\mathbf{x}^{(0)}(t)$ for the solution. After solving each subsystem for the whole time period of interest $t \in [0, T]$, a new approximation for the solution is calculated as

$$\mathbf{x}^{(r+1)}(t) = \sum_{l=1}^S \mathbf{E}_l \mathbf{y}_l^{(r+1)}(t) \quad (4.4)$$

The next iteration is solved using the updated values of the solution in (4.4). The iterations are repeated until convergence is achieved within an acceptable tolerance.

Figure 4.1: Overlapping partitioning of m coupled lines

4.2 Overlapping Partitioning Algorithm for Tightly Coupled Interconnects

This section presents a novel waveform relaxation method based on overlapping partitioning for tightly coupled interconnects. This method is based on the matrix multi-splitting algorithm presented in Section 4.1. The tightly coupled interconnects are partitioned into S subcircuits, where each subcircuit contains one or more lines. In addition, the subcircuits are allowed to overlap; i.e., each subcircuit contains one or more lines that exist in an adjacent subcircuit. This is depicted in Figure 4.1, for the case of one overlapped line between adjacent subcircuits.

Consider the Telegrapher's equations for m coupled transmission lines presented in (2.10) and (2.11) [47]

$$\frac{\partial}{\partial z} \mathbf{v}(z, t) = -\bar{\mathbf{R}} \mathbf{i}(z, t) - \bar{\mathbf{L}} \frac{\partial}{\partial t} \mathbf{i}(z, t) \quad (4.5)$$

$$\frac{\partial}{\partial z} \mathbf{i}(z, t) = -\bar{\mathbf{G}} \mathbf{v}(z, t) - \bar{\mathbf{C}} \frac{\partial}{\partial t} \mathbf{v}(z, t) \quad (4.6)$$

Using Definition 6, the **RLGC** parameter matrices can be partitioned into S subcircuits as follows [25]

$$\begin{aligned} \bar{\mathbf{R}} &= \bar{\mathbf{R}}_{M,l} - \bar{\mathbf{R}}_{N,l} \quad , \quad \bar{\mathbf{L}} = \bar{\mathbf{L}}_{M,l} - \bar{\mathbf{L}}_{N,l} \\ \bar{\mathbf{C}} &= \bar{\mathbf{C}}_{M,l} - \bar{\mathbf{C}}_{N,l} \quad , \quad \bar{\mathbf{G}} = \bar{\mathbf{G}}_{M,l} - \bar{\mathbf{G}}_{N,l} \end{aligned} \quad (4.7)$$

$$l = 1, \dots, S$$

Using (4.7), the equations of the l^{th} subcircuit can be written at the r^{th} iteration as follows

$$\begin{aligned} \frac{\partial \hat{\mathbf{v}}_l^{(r+1)}(z, t)}{\partial z} &= -\bar{\mathbf{R}}_{M,l} \hat{\mathbf{i}}_l^{(r+1)}(z, t) - \bar{\mathbf{L}}_{M,l} \frac{\partial \hat{\mathbf{i}}_l^{(r+1)}(z, t)}{\partial t} \\ &+ \bar{\mathbf{R}}_{N,l} \mathbf{i}^{(r)}(z, t) + \bar{\mathbf{L}}_{N,l} \frac{\partial \mathbf{i}^{(r)}(z, t)}{\partial t} \end{aligned} \quad (4.8)$$

$$\begin{aligned} \frac{\partial \hat{\mathbf{i}}_l^{(r+1)}(z, t)}{\partial z} &= -\bar{\mathbf{G}}_{M,l} \hat{\mathbf{v}}_l^{(r+1)}(z, t) - \bar{\mathbf{C}}_{M,l} \frac{\partial \hat{\mathbf{v}}_l^{(r+1)}(z, t)}{\partial t} \\ &+ \bar{\mathbf{G}}_{N,l} \mathbf{v}^{(r)}(z, t) + \bar{\mathbf{C}}_{N,l} \frac{\partial \mathbf{v}^{(r)}(z, t)}{\partial t} \end{aligned} \quad (4.9)$$

$$l = 1, \dots, S$$

where $\hat{\mathbf{v}}_l(z, t)$, $\hat{\mathbf{i}}_l(z, t)$ are the auxiliary voltages and currents along the lines in the l^{th} subcircuit.

Equations (4.8) and (4.9) can be re-written as

$$\begin{aligned} \frac{\partial \hat{\mathbf{v}}_l^{(r+1)}(z, t)}{\partial z} &= -\bar{\mathbf{R}}_{M,l} \hat{\mathbf{i}}_l^{(r+1)}(z, t) - \bar{\mathbf{L}}_{M,l} \frac{\partial \hat{\mathbf{i}}_l^{(r+1)}(z, t)}{\partial t} \\ &+ \mathbf{H}_l^{(r)}(z, t) \end{aligned} \quad (4.10)$$

$$\begin{aligned} \frac{\partial \hat{\mathbf{i}}_l^{(r+1)}(z, t)}{\partial z} &= -\bar{\mathbf{G}}_{M,l} \hat{\mathbf{v}}_l^{(r+1)}(z, t) - \bar{\mathbf{C}}_{M,l} \frac{\partial \hat{\mathbf{v}}_l^{(r+1)}(z, t)}{\partial t} \\ &+ \mathbf{F}_l^{(r)}(z, t) \end{aligned} \quad (4.11)$$

where

$$\mathbf{H}_l^{(r)}(z, t) = \bar{\mathbf{R}}_{N,l} \mathbf{i}^{(r)}(z, t) + \bar{\mathbf{L}}_{N,l} \frac{\partial \mathbf{i}^{(r)}(z, t)}{\partial t} \quad (4.12)$$

$$\mathbf{F}_l^{(r)}(z, t) = \bar{\mathbf{G}}_{N,l} \mathbf{v}^{(r)}(z, t) + \bar{\mathbf{C}}_{N,l} \frac{\partial \mathbf{v}^{(r)}(z, t)}{\partial t} \quad (4.13)$$

Once all the subcircuits are simulated, the voltages and currents of the original coupled transmission lines can be updated using the auxiliary voltages and currents as

$$\mathbf{v}^{(r+1)}(z, t) = \sum_{l=1}^S \mathbf{E}_l \hat{\mathbf{v}}_l^{(r+1)}(z, t) \quad (4.14)$$

$$\mathbf{i}^{(r+1)}(z, t) = \sum_{l=1}^S \mathbf{E}_l \hat{\mathbf{i}}_l^{(r+1)}(z, t) \quad (4.15)$$

where $\mathbf{E}_l$ are the weighting matrices defined in (4.2).

In order to allow overlapping partitioning between adjacent subcircuits, the weighting matrices $\mathbf{E}_l$ are selected as [57]

$$E_{l(i,j)} = \begin{cases} 0 & i \neq j \text{ or } i = j \notin \mathbf{T}_l \\ 1 & i = j \in \mathbf{T}_l - (\mathbf{T}_{l+1} \cup \mathbf{T}_{l-1}) \\ \alpha_i & i = j \in \mathbf{T}_l \cap \mathbf{T}_{l+1} \\ 1 - \alpha_i & i = j \in \mathbf{T}_l \cap \mathbf{T}_{l-1} \end{cases} \quad (4.16)$$

where $\mathbf{T}_l$ is the set of lines in the l^{th} subcircuit and $0 \leq \alpha_i \leq 1$.

In most practical cases it is sufficient to choose $\alpha_i = \frac{1}{2}$. However, more intelligent values can be applied depending on the subcircuit of each overlapped line. Further details regarding selection of the weighting matrices can be found in [8].

Next, using (4.10)-(4.13), the equations for the j^{th} line in the l^{th} subcircuit can be written as

$$\frac{\partial \hat{v}_{l,j}^{(r+1)}(z, t)}{\partial z} = -\bar{R}_{jj} \hat{i}_{l,j}^{(r+1)}(z, t) - \bar{L}_{jj} \frac{\partial \hat{i}_{l,j}^{(r+1)}(z, t)}{\partial t} + h_{l,j}^{(r)}(z, t) \quad (4.17)$$

$$\frac{\partial \hat{i}_{l,j}^{(r+1)}(z,t)}{\partial z} = -\bar{G}_{jj}\hat{v}_{l,j}^{(r+1)}(z,t) - \bar{C}_{jj}\frac{\partial \hat{v}_{l,j}^{(r+1)}(z,t)}{\partial t} + f_{l,j}^{(r)}(z,t) \quad (4.18)$$

where

$$h_{l,j}^{(r)}(z,t) = - \sum_{\substack{k=1 \\ k \notin \mathbf{T}_l}}^m \left(\bar{R}_{jk} i_k^{(r)}(z,t) + \bar{L}_{jk} \frac{\partial i_k^{(r)}(z,t)}{\partial t} \right) \quad (4.19)$$

$$f_{l,j}^{(r)}(z,t) = - \sum_{\substack{k=1 \\ k \notin \mathbf{T}_l}}^m \left(\bar{G}_{jk}(v_k^{(r)}(z,t)) + \bar{C}_{jk} \left(\frac{\partial v_k^{(r)}(z,t)}{\partial t} \right) \right)$$

and

- $\hat{v}_{l,j}(z,t)$ and $\hat{i}_{l,j}(z,t)$ are the auxiliary voltages and currents along the j^{th} line in the l^{th} subcircuit,
- $v_j(z,t)$ and $i_j(z,t)$ are the coupled voltages and currents along the j^{th} line, which are computed using (4.14) and (4.15).

Note that $h_{l,j}(z,t)$ and $f_{l,j}(z,t)$ represent the coupling effects on line j in subcircuit l from the neighboring lines outside subcircuit l .

There are two important remarks worth noting:

- When simulating the l^{th} subcircuit, the lines that correspond to a zero diagonal element in $\mathbf{E}_l$ are not simulated. For example, in Figure 4.1, lines 1 and 2, which are tightly coupled, are simulated together in

Figure 4.2: Equivalent distributed representation for the j^{th} line in subcircuit l

one subcircuit decoupled from lines 3 to m . Next, lines 2 and 3 are simulated together, decoupled from other lines, etc.

- The partitioned subcircuits are solved independently, as a result, this method is highly suited for parallel implementation.

4.2.1 Illustrative example:

In order to demonstrate the overlapping partitioning when applied to coupled transmission lines, consider the coupled transmission line structure shown in Figure 4.1 with $m = 3$. If the coupled lines are partitioned into two subcircuits ($S = 2$), the multi-splitting of the **RLGC** matrices for each subcircuit can be written as

$$\bar{\mathbf{R}}_{M,1} = \begin{bmatrix} \bar{R}_{11} & \bar{R}_{12} & 0 \\ \bar{R}_{21} & \bar{R}_{22} & 0 \\ 0 & 0 & \bar{R}_{33} \end{bmatrix}, \bar{\mathbf{R}}_{M,2} = \begin{bmatrix} \bar{R}_{11} & 0 & 0 \\ 0 & \bar{R}_{22} & \bar{R}_{23} \\ 0 & \bar{R}_{32} & \bar{R}_{33} \end{bmatrix} \quad (4.20)$$

$$\bar{\mathbf{R}}_{N,1} = \begin{bmatrix} 0 & 0 & -\bar{R}_{13} \\ 0 & 0 & -\bar{R}_{23} \\ -\bar{R}_{31} & -\bar{R}_{32} & 0 \end{bmatrix}, \bar{\mathbf{R}}_{N,2} = \begin{bmatrix} 0 & -\bar{R}_{12} & -\bar{R}_{13} \\ -\bar{R}_{21} & 0 & 0 \\ -\bar{R}_{31} & 0 & 0 \end{bmatrix} \quad (4.21)$$

$$\mathbf{E}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{E}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 - \alpha_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.22)$$

It can also be shown that the multi-splitting of the $\bar{\mathbf{L}}$, $\bar{\mathbf{G}}$, and $\bar{\mathbf{C}}$ matrices will have a similar structure as the $\bar{\mathbf{R}}$ matrix above. For this example, the coupling effects on line 1 in subsystem 1 are given by

$$h_{1,1}^{(r)}(z, t) = - \left[R_{13} i_3^{(r)}(z, t) + L_{13} \frac{\partial i_3^{(r)}(z, t)}{\partial t} \right] \quad (4.23)$$

$$f_{1,1}^{(r)}(z, t) = - \left[G_{13} (v_1^{(r)}(z, t) - v_3^{(r)}(z, t)) + C_{13} \left(\frac{\partial v_1^{(r)}(z, t)}{\partial t} - \frac{\partial v_3^{(r)}(z, t)}{\partial t} \right) \right] \quad (4.24)$$

The sources above represent the coupling effects on line 1 due to line 3, which is outside of subcircuit 1. Similarly, sources for the remaining lines are calculated in the same manner using (4.19) and (4.20) for $l = 1, 2$ and $j = 1, 2, 3$.

It is to be noted that the overlapping partitioning method can also be easily applied to the case of frequency-dependent (FD) p.u.l parameters. In this case, the p.u.l impedance of each subcircuit is approximated by a positive real function using

vector-fit [58] method followed by a passivity check. The resulting functions are realized using RL/RC circuits. The overlapping partitioning method can then be applied directly to the resulting circuits [37].

4.3 Practical Considerations

4.3.1 Distributed Representation

It was shown in [1] (*also reviewed in Sections 2.2.2 and 2.2.1*) that the coupling effects in (4.19) and (4.20) can be realized using voltage and current sources (referred to as relaxation sources). Two approaches were presented in [1]: the single ended approach and the distributed approach. In the single ended representation, the relaxation sources are located at the end of each line. However, in this case, the source computations require numerous FFT/IFFT operations, which can slow down the simulation process. On the other hand, using the distributed representation, the line is discretized into small sections using lumped segmentation [47]. In this case, the relaxation sources are located at the end of each section, and have a direct time-domain representation, thus avoiding expensive FFT/IFFT operations.

Using the proposed algorithm with the distributed lumped segmentation for the transmission lines, the relaxation sources are defined as [1]

$$\tilde{h}_{l,j}(z_p, t) = \Delta z h_{l,j}(z_p, t) \quad (4.25)$$

$$\tilde{f}_{l,j}(z_p, t) = \Delta z f_{l,j}(z_p, t) \quad (4.26)$$

$$p = 1, \dots, n$$

where $\Delta z = \frac{d}{n}$, d is the length of the line and n is the number of lumped sections.

This realization is depicted in Figure 4.2.

4.3.2 Proper Partitioning for Improved Convergence

We start by considering the frequency-domain representation of the Telegrapher's equations in (4.5)-(4.6). Based on the solution of these equations, the terminal currents and voltages of the interconnect circuit are related by the admittance parameter matrix $\mathbf{Y}$ in the form:

$$\mathbf{Y} \begin{bmatrix} V_1(s) \\ V_2(s) \\ \vdots \\ V_{2m}(s) \end{bmatrix} = \begin{bmatrix} I_1(s) \\ I_2(s) \\ \vdots \\ I_{2m}(s) \end{bmatrix} \quad (4.27)$$

where $V_\eta(s)$, $I_\eta(s)$, $\eta = 1, 2, \dots, 2m$, represent the terminal voltages and currents on all the m lines, respectively.

Let the $\mathbf{Y}$ matrix be partitioned using Definition 1 as

$$\mathbf{Y} = \mathbf{Y}_{M,l} - \mathbf{Y}_{N,l} \quad (4.28)$$

$$l = 1 \dots S$$

Next, we define the following:

Definition 7. *$\mathbf{Y}$ is called a “block-diagonal dominant matrix in regard to the block-partitioning” if [8, 40]*

$$\Gamma = \|\mathbf{U} - \sum_{l=1}^S (\mathbf{E}_l \mathbf{Y}_{M,l}^{-1}) \mathbf{Y}\| < 1 \quad (4.29)$$

where Γ is the convergence factor, $\mathbf{U}$ is the identity matrix and $\|\cdot\|$ is the matrix norm operator.

Equation (4.27) can be solved iteratively using the multi-splitting partitioning algorithm as follows [8]

$$\mathbf{V}^{(r+1)} = \mathbf{Q} \mathbf{V}^{(r)} + \sum_{l=1}^S \mathbf{E}_l \mathbf{Y}_{M,l}^{-1} \mathbf{I} \quad (4.30)$$

where $\mathbf{Q}$ is the iteration matrix and is defined as

$$\mathbf{Q} = \sum_{l=1}^S \mathbf{E}_l \mathbf{Y}_{M,l}^{-1} \mathbf{Y}_{N,l} \quad (4.31)$$

The following theorem presents a sufficient condition for convergence of the iterations in (4.30).

Theorem 1. *If $\mathbf{Y}$ is partitioned in the form of a block-diagonal dominant matrix then the spectral radius of the iteration matrix $\rho(\mathbf{Q}) < 1$ and the multi-splitting partitioning algorithm (4.30) is convergent.*

Proof. If $\|\mathbf{Q}\| < 1$ for any matrix norm, then the spectral radius of the iteration matrix $\rho(\mathbf{Q}) < 1$ [8]

The norm of the iteration matrix $\mathbf{Q}$ can be written as

$$\|\mathbf{Q}\| = \left\| \sum_{l=1}^S \mathbf{E}_l \mathbf{Y}_{M,l}^{-1} \mathbf{Y}_{N,l} \right\| \quad (4.32)$$

$$= \left\| \sum_{l=1}^S \mathbf{E}_l \mathbf{Y}_{M,l}^{-1} (\mathbf{Y}_{M,l} - \mathbf{Y}) \right\| \quad (4.33)$$

$$= \left\| \sum_{l=1}^S \mathbf{E}_l - \sum_{l=1}^S \mathbf{E}_l \mathbf{Y}_{M,l}^{-1} \mathbf{Y} \right\| \quad (4.34)$$

$$= \left\| \mathbf{U} - \sum_{l=1}^S (\mathbf{E}_l \mathbf{Y}_{M,l}^{-1}) \mathbf{Y} \right\| = \Gamma \quad (4.35)$$

Therefore, if the $\mathbf{Y}$ matrix is partitioned in the form of a block-diagonal dominant matrix, then according to Definition 2, this results in $\|\mathbf{Q}\| < 1$, and thus $\rho(\mathbf{Q}) < 1$.

□

It is to be noted that the size of the $\mathbf{Y}$ matrix is very small compared to the size of the matrices resulting from discretizing the coupled Telegrapher's equations. Therefore, the time required to calculate Γ in (4.29) is a very small compared to the simulation time for the large coupled discretized circuit.

Illustrative example:

Consider the three coupled transmission lines shown in Figure 4.3. In this example, lines 1 and 2 are strongly coupled.

First, the coupled lines are partitioned into two subcircuits, where the first sub-

Figure 4.3: Three transmission lines with different coupling between the lines

circuit contains lines 1 and 3, and the second subcircuit contains lines 2 and 3, as shown in Figure 4.4(a). This partitioning scheme resulted in a convergence factor of $\Gamma = 1.25 > 1$, due to the fact that the strongly coupled lines are not bundled within the same subcircuit. In this case, the multi-splitting partitioning algorithm did not converge.

Next, the coupled lines are partitioned such that the strongly coupled lines are bundled within the same subcircuit, ie. the first subcircuit contains lines 1 and 2, and the second subcircuit contains lines 2 and 3 (as shown in Figure 4.4(b)). In this case, the partitioned $\mathbf{Y}$ parameter matrix is block-diagonal dominant, resulting in $\Gamma = 0.72 < 1$ and the multi-splitting partitioning algorithm converged in 5 iterations.

4.3.3 Computational Steps

The computational steps for the OP algorithm can be summarized as shown in Algorithm 4.1. Algorithm 4.2 summarizes the steps to partition the interconnect circuit such that $\Gamma < 1$.

Figure 4.4: Partitioning of the $\mathbf{Y}$ -parameter matrix (a) Partitioning scheme which does not result in a block-diagonal dominant $\mathbf{Y}$ matrix (b) Proper partitioning approach which results in a block-diagonal dominant $\mathbf{Y}$ matrix.

Algorithm 4.1: Waveform Relaxation with Overlapping Partitioning Algorithm

input : Tightly Coupled Interconnects Circuit
output: Transient Simulation

- 1 **while** *No Convergence* **do**
- 2 Calculate the values of the relaxation sources $\tilde{h}_{l,j}(z_p, t)$ and $\tilde{f}_{l,j}(z_p, t)$ in (4.25) and (4.26) for $l = 1, \dots, S$, and $j = 1, \dots, m$;
- 3 Simulate each subcircuit independently to solve for the auxiliary voltages and currents $\hat{v}_l^{(r+1)}(z_p, t)$ and $\hat{i}_l^{(r+1)}(z_p, t)$ in (4.17) and (4.18) .;
- 4 Update the voltages and currents of the original coupled interconnects ($\mathbf{v}^{(r+1)}(z_p, t)$ and $\mathbf{i}^{(r+1)}(z_p, t)$) using (4.14) and (4.15) ;
- 5 $r \leftarrow r+1$;

Algorithm 4.2: Partitioning Algorithm for the overlapping method

input : $\mathbf{Y}$ parameters matrix
output: Partitioned $\mathbf{Y}$ parameters

- 1 Set $\Gamma > 1$;
- 2 $i \leftarrow 0$;
- 3 $q \leftarrow 0$;
- 4 **while** $\Gamma > 1$ **do**
- 5 **if** $q \geq i - 1$ **then**
- 6 $i \leftarrow i + 1$;
- 7 $q \leftarrow 0$;
- 8 **else**
- 9 $q \leftarrow q + 1$;
- 10 Partition the $\mathbf{Y}$ matrix using (4.28), such that each group contains i lines and the number of overlapped lines is q . ;
- 11 Calculate Γ using (4.29). ;
- 12 Partition the **RLGC** parameters using the final values of i and q . ;

4.4 Frequency Dependent RLGC Parameters

At relatively high frequencies, skin, edge and proximity effects [47] must be taken into account when modeling coupled interconnects circuits. Typically, due to these effects,

the p.u.l RLGC parameters describing the lines are frequency-dependent. In this case, the frequency-dependent RLGC parameters are given in the form of tabulated data that can be found from measurements or using electromagnetic simulation.

The handling of the frequency-dependent parameters has been addressed in [37]. The main idea in this approach is to use a fitting algorithm, such as vector-fitting [58], to find a rational function that approximates the transfer function of the lines. Next, the passivity of the resulting rational functions can be checked and enforced using the algorithm described in [59, 60]. Finally, each rational function is synthesized using RL/RC components and used to form an equivalent circuit for the lines. Once the equivalent circuits are obtained, the overlapping partitioning method can be used to find the transient response of the interconnects circuits as described in section 4.2.

4.5 Numerical Examples

In this section, two numerical examples are presented. The examples compare the convergence properties of the proposed waveform relaxation with overlapping partitioning method (WR-TP-OP), versus the original waveform relaxation approach (WR-TP). In addition, Example 2 shows the robustness of the proposed method in the case of a large number of lines.

The **RLGC** parameters of the lines were calculated using field solvers in HSPICE [61], and the CPU time was measured on an Intel Core 2 CPU 2.40 GHz PC.

Figure 4.5: Physical parameters for the lines used in Example 1

4.5.1 Example 1

The interconnect circuit shown in Figure 4.5 is considered. The circuit consists of 9 coupled lines of length $d = 5 \text{ cm}$. The lines were modeled using lumped segmentation ($n = 40$ sections). A voltage source is attached at every fourth line, with rise and fall times of 0.5ns, pulse width of 10ns and 5V in magnitude. The total number of sources in this case is three.

For this example, the classical WR-TP algorithm without overlapping partitioning did not converge. Figure 4.6 shows the output voltage at the far-end of victim line 2 obtained using HSPICE and the WR-TP without overlapping partitioning after 4 iterations, which exhibits considerable error.

Next, the coupled interconnect circuit was simulated using the proposed algorithm. For this purpose, the circuit was partitioned such that each subcircuit contains 3 lines, with one overlapping line between subcircuits ($q = 1$), as shown in Figure 4.7. In this case, the proposed WR-TP with overlapping partitioning converged in 4 iterations as demonstrated in Figure 4.8.

It is to be noted that the case of WR-TP without overlapping partitioning yielded a convergence factor of $\Gamma = 1.23$, while the proposed WR-TP-OP yielded $\Gamma = 0.68$.

Figure 4.6: Comparison of the time-domain responses at the far-end of victim line 2 (Example 1)

4.5.2 Example 2

In this example, we consider 64 coupled lines of length $d = 6cm$ (see Figure 4.9). The lines were modeled using conventional lumped segmentation ($n = 45$ sections). A voltage source is attached at every fourth line, with rise and fall times of 0.5ns, pulse

Figure 4.7: Partitioning of the lines in Example 1

Figure 4.8: Comparison of the time-domain responses at the far-end of victim line 2 (Example 1)

width of 10ns and 5V in magnitude. The total number of the sources used in this example is 22.

The coupled interconnect circuit was partitioned such that each subcircuit contains 4 lines, with one overlapping line between subcircuits ($q = 1$). Figure 4.10 shows the output voltage at the far-end of victim line 2 obtained using HSPICE and the proposed WR-TP-OP algorithm after 5 iterations. In addition, the original WR-

Figure 4.9: Physical parameters for the lines used in Example 2

TP without overlapping partitioning did not converge for this example ($\Gamma = 2.1$ for WR-TP).

Figure 4.10: Comparison of the time-domain responses at the far-end of line 2 with $q = 1$ (Example 2)

Next, the coupled interconnect circuit was partitioned such that each subcircuit contains 4 lines and the number of overlapped lines was varied such that $q = 2, 3$. Figure 4.13 shows the partitioning scheme when $q = 3$. Figure 4.11 shows a sample of the time-domain response at the far-end of line 12 using the proposed method after 5

Figure 4.11: Comparison of the time-domain responses at the far-end of line 12 with $q = 2$ (Example 2)

Figure 4.12: Comparison of the time-domain responses at the near-end of line 12 with $q = 3$ (Example 2)

iterations when $q = 2$. Furthermore, Figure 4.12 shows the time-domain response at the near-end of line 12 using the proposed method after 4 iterations when $q = 3$. As seen, the results are in very good agreement.

Figure 4.13: Partitioning of the lines in Example 2 in the case of $q = 3$

Table 4.1 summarizes the CPU time and the value of Γ for different q . As expected, convergence is achieved faster when the value of Γ is relatively small. It is also to be noted that increasing the number of overlapped lines increases the accuracy for a given number of iterations (or alternatively reduces the required number of iterations for a given accuracy). At the same time, having too many overlapped lines (which in this case may not really account for strong coupling) may lead to an overall increase in the CPU time. It should also be emphasized that varying the number of overlapped lines didn't affect the accuracy of the final converged response.

In addition, the CPU time required for the transient simulation of the full coupled interconnect circuit using HSPICE was 443.84 sec. As seen, even in the case of $q = 3$, the CPU time is less than that of HSPICE. This is due to the fact that simulating four lines in each group requires very small CPU time. Moreover, in the case of $q = 3$, the number of iterations required for convergence decreased to four iterations, thus decreasing the overall CPU time.

Table 4.1: CPU time comparison for the proposed WR-TP with OP for varying q (Example 2)

Simulation of full circuit using HSPICE			443.84 sec.
Iterations #	$q = 1$	$q = 2$	$q = 3$
2	34.78 sec	52.32 sec	113.68 sec
4	74.57 sec	108.52 sec	245.36 sec
5	95.33 sec	136.15 sec	(Converged after 4 iter.)
6	111.95 sec (converged)	(Converged after 5 iter.)	
Γ	0.80	0.71	0.59

4.6 Summary

In this chapter, a new waveform relaxation with overlapping partitioning algorithm is presented for tightly coupled interconnects. The partitioning is applied in the transverse direction in order to exploit the relatively weak coupling between the lines. The lines are partitioned into subcircuits such that each subcircuit contains one or more lines that exist in an adjacent subcircuit. The new method not only preserves the advantages of the original waveform relaxation with transverse partitioning method, but also improves the convergence rate in the case of tightly coupled interconnects. Moreover, the partitioned subcircuits are completely independent, thus opening the door for parallel implementation. Numerical examples were provided, which validate the accuracy and efficiency of the proposed method.

Chapter 5

Accelerated Overlapping Partitioning Techniques using Model Order Reduction

It was shown in Chapter 4 that using the OP method provides a fast convergence in the case of having a strong coupling between the lines. However, as the strong coupling between the lines increases, more lines are grouped in the same subcircuit to overcome the strong coupling. Thus, as the coupling increases, the number of lines per subcircuit increases. In this case, the total number of subcircuits is given by

$$\text{Total number of subcircuits} = \lceil \frac{\text{Total number of lines} - \text{Overlapped lines}}{\text{Lines per subcircuit} - \text{Overlapped lines}} \rceil \quad (5.1)$$

where $\lceil . \rceil$ is the ceiling function.

In order to reduce the simulation time required for each iteration, a reduced order macromodel is first obtained for each partitioned subcircuit using MOR techniques. The obtained macromodels are then combined with nonlinear devices and simulated

Figure 5.1: Simulating the l^{th} subcircuit with open circuit terminations to find the Thevenin-equivalent sources

Figure 5.2: The reduced model of the l^{th} subcircuit with the nonlinear devices attached to the lines, where m_l is the number of lines in the l^{th} subcircuit.

using SPICE-like simulators [27].

In the first step, to facilitate the application of reduction techniques, a method to move the the distributed relaxation sources to the terminals of the lines using the generalized Thevenin theorem is developed. In the next step, MOR is applied to reduce the size of each resulting subcircuit, and the corresponding details are presented. Subsequently, a new method is presented to update the relaxation sources from the solution of the reduced subcircuits.

5.1 Handling of Distributed Relaxation Sources

In order to facilitate the application of MOR techniques for each subcircuit, the distributed relaxation sources (which are located at each lumped segment) must be

moved to the end of the lines. This is achieved by replacing the relaxation sources with their Thevenin equivalent sources at the end of the ports. These equivalent sources are derived using the generalized Thevenin theorem for an active two port network [62,63], thus referred to as Thevenin sources. The Thevenin sources are calculated by finding the open circuit voltages at the two ports of the lines (Figure 5.1). Once the Thevenin sources are calculated, a reduced model for the l^{th} subcircuit is computed and inserted in the full network with the nonlinear devices, as depicted in Figure 5.2. The step of finding the Thevenin sources can be represented using the Modified Nodal Analysis (MNA) [3] equations as follows

$$\mathbf{G}_l \dot{\mathbf{x}}_{oc,l}^{(r+1)}(t) + \mathbf{C}_l \mathbf{x}_{oc,l}^{(r+1)}(t) = \mathbf{D}_l \begin{bmatrix} \boldsymbol{\eta}_l^{(r)}(t) \\ \boldsymbol{\gamma}_l^{(r)}(t) \end{bmatrix} \quad (5.2)$$

$$\mathbf{v}_{th,l}^{(r+1)}(t) = \mathbf{B}_l \mathbf{x}_{oc,l}^{(r+1)} \quad (5.3)$$

where

- r is the waveform relaxation iteration number,
- $\mathbf{x}_{oc,l} \in \mathbb{R}^{M_l}$ is the vector of unknown voltages/currents waveforms, with M_l being the total number of unknown voltages/currents waveforms of the l^{th} subcircuit,
- $\mathbf{G}_l, \mathbf{C}_l \in \mathbb{R}^{M_l \times M_l}$ are constant matrices representing the energy storage elements and conductances of the l^{th} subcircuit, respectively,

- $\mathbf{D}_l$ is a selector matrix that maps the values of the distributed relaxation sources to the MNA variables space,
- $\mathbf{B}_l$ is a selector matrix that selects the voltages at the terminals of the lines,
- $\boldsymbol{\eta}_l^{(r)}(t)$ and $\boldsymbol{\gamma}_l^{(r)}(t)$ are vectors that contain the values of the distributed relaxation sources, which represent the coupling between the subcircuits and are located after each lumped section.

Moving the relaxation sources to the terminals allows representing the current-voltage characteristics at the terminals of the l^{th} subcircuit using the MNA equations as follows

$$\begin{aligned}\mathbf{G}_l \dot{\mathbf{x}}_{\text{th},l}(t) + \mathbf{C}_l \mathbf{x}_{\text{th},l}(t) &= \mathbf{B}_l \mathbf{w}_l(t), \\ \mathbf{u}_l(t) &= \mathbf{B}_l^T \mathbf{x}_{\text{th},l}(t).\end{aligned}\tag{5.4}$$

where

- $\mathbf{x}_{\text{th},l} \in \mathbb{R}^{M_l}$ is the vector of voltages/currents waveforms obtained from the Thevenin-equivalent of the l^{th} subcircuit,
 - $\mathbf{w}_l(t)$ are the currents at the ports of the l^{th} subcircuit,
 - $\mathbf{u}_l(t)$ are the voltages at the ports of the l^{th} subcircuit,
-

5.2 Application of MOR

Next, (5.4) is reduced by mapping the MNA variables $\mathbf{x}_{\text{th},l}$ into the lower dimension space $\hat{\mathbf{x}}_{\text{th},l}$, where $\hat{\mathbf{x}}_{\text{th},l}$ has smaller dimension than $\mathbf{x}_{\text{th},l}$ and is defined as [64]

$$\mathbf{x}_{\text{th},l} = \mathbf{Q}_l \hat{\mathbf{x}}_{\text{th},l} \quad (5.5)$$

where $\mathbf{Q}_l \in \mathbb{R}^{M_l \times q_l}$ is an orthonormal matrix whose columns form bases for the Krylov subspace generated by the matrices $\mathbf{A}_l$ and $\mathbf{R}_l$, given by

$$\text{colsp}(\mathbf{Q}_l) = Kr(\mathbf{A}_l, \mathbf{R}_l, q_l) = \text{colsp}(\mathbf{R}_l, \mathbf{A}_l \mathbf{R}_l, \dots, \mathbf{A}_l^{k_l-1} \mathbf{R}_l) \quad (5.6)$$

where $\mathbf{A}_l = -\mathbf{C}_l^{-1} \mathbf{G}_l$, $\mathbf{R}_l = \mathbf{C}_l^{-1} \mathbf{b}_l$, $k_l = \lfloor \frac{q_l}{2N_l} \rfloor$

The matrix $\mathbf{Q}_l$ is found using the Arnoldi algorithm given in [65]. Next, transforming $\mathbf{x}_{\text{th},l}$ to $\hat{\mathbf{x}}_{\text{th},l}$ results in reduced-order MNA equations of order q in the following form

$$\begin{aligned} \hat{\mathbf{G}}_l \hat{\mathbf{x}}_{\text{th},l}(t) + \hat{\mathbf{C}}_l \dot{\hat{\mathbf{x}}}_{\text{th},l}(t) &= \hat{\mathbf{B}}_l \mathbf{w}_l(t), \\ \mathbf{u}_l(t) &= \hat{\mathbf{B}}_l^T \hat{\mathbf{x}}_{\text{th},l}(t). \end{aligned} \quad (5.7)$$

where $\hat{\mathbf{G}}_l = \mathbf{Q}_l^T \mathbf{G}_l \mathbf{Q}_l$, $\hat{\mathbf{C}}_l = \mathbf{Q}_l^T \mathbf{C}_l \mathbf{Q}_l$ and $\hat{\mathbf{B}}_l = \mathbf{Q}_l^T \mathbf{B}_l$.

It is worth highlighting that the reduced model of each subcircuit is independent

of the OP iterations, thus the reduced models are only found prior to the simulation process. The models are then combined at each iteration by the nonlinear devices and used in the simulation process [66].

5.2.1 Updating the Relaxation Sources

At a given iteration of the OP method, the MNA formulation of each reduced sub-circuit is embedded in the MNA representation of the complete network with the terminal devices, as shown in Figure 5.2. However, in order to update the relaxation sources in (4.19) and (4.20), the auxiliary voltages and currents $\hat{\mathbf{v}}_l(z_p, t)$ and $\hat{\mathbf{i}}_l(z_p, t)$ at distances z_p , $p = 1, \dots, n$ along the lines are required. These auxiliary voltages and currents are not readily available due to simulating each subcircuit using the Thevenin equivalent circuit.

Lemma 3. *If $\mathbf{x}_{\text{oc},l}(t)$ is the vector of voltages/currents obtained by simulating the l^{th} subcircuit with open circuit terminations, then the auxiliary voltages and currents can be written as*

$$\tilde{\mathbf{v}}_l^{(r+1)}(z_p, t) = \mathbf{P}_{p,l} \left[\mathbf{x}_{\text{oc},l}^{(r+1)}(t) + \mathbf{Q}_l \hat{\mathbf{x}}_{\text{th},l}^{(r+1)}(t) \right] \quad (5.8)$$

$$\tilde{\mathbf{i}}_l^{(r+1)}(z_p, t) = \mathbf{W}_{p,l} \left[\mathbf{x}_{\text{oc},l}^{(r+1)}(t) + \mathbf{Q}_l \hat{\mathbf{x}}_{\text{th},l}^{(r+1)}(t) \right] \quad (5.9)$$

where $\mathbf{P}_{p,l}$ and $\mathbf{W}_{p,l} \in [0, 1]$ are selector matrices, which select the voltages and cur-

rents at distance z_p along the lines.

Proof. The auxiliary voltages and currents are calculated by turning on the relaxation sources attached after each lumped section and selecting the voltages/currents at distances z_p along the lines as follows

$$\mathbf{G}_l \dot{\mathbf{x}}_l^{(r+1)}(t) + \mathbf{C}_l \mathbf{x}_l^{(r+1)}(t) = \mathbf{B}_l \mathbf{w}_l^{(r+1)}(t) + \mathbf{D}_l \begin{bmatrix} \boldsymbol{\eta}_l^{(r)}(t) \\ \boldsymbol{\gamma}_l^{(r)}(t) \end{bmatrix}, \quad (5.10)$$

$$\tilde{\mathbf{v}}_l^{(r+1)}(z_p, t) = \mathbf{P}_{p,l} \mathbf{x}_l^{(r+1)}(t), \quad (5.11)$$

$$\tilde{\mathbf{i}}_l^{(r+1)}(z_p, t) = \mathbf{W}_{p,l} \mathbf{x}_l^{(r+1)}(t). \quad (5.12)$$

where r represent the waveform relaxation iteration number.

The relaxation sources are moved to the terminals of the lines by solving each subcircuit with the distributed relaxation sources attached at each lumped section (at z_p) with open circuit terminations (5.2)

Adding (5.2) with (5.4) results in

$$\begin{aligned} \mathbf{G}_l \left(\dot{\mathbf{x}}_{\text{th},l}(t) + \dot{\mathbf{x}}_{\text{oc},l}(t) \right)^{(r+1)} + \mathbf{C}_l \left(\mathbf{x}_{\text{th},l}(t) + \mathbf{x}_{\text{oc},l}(t) \right)^{(r+1)} \\ = \mathbf{B}_l \mathbf{w}_l^{(r)} + \mathbf{D}_l \begin{bmatrix} \boldsymbol{\eta}_l^{(r)}(t) \\ \boldsymbol{\gamma}_l^{(r)}(t) \end{bmatrix} \end{aligned} \quad (5.13)$$

Comparing (5.13) with (5.10) results in

$$\mathbf{x}_l = \mathbf{x}_{\text{th},l}(t) + \mathbf{x}_{\text{oc},l}(t) \quad (5.14)$$

By employing the relation in (5.5), the auxiliary voltages and currents at z_p can then be calculated using (5.8) and (5.9), respectively.

□

5.3 Computational Complexity

Various components of the OP algorithm are summarized in the form of pseudo-code in Algorithm 5.1. The main computational steps of the OP method can be summarized as follows:

1. Reduction of each subcircuit (Step: 3).
2. Solving the complete network containing reduced subcircuits (Step: 9).
3. Updating the relaxation sources (Steps: 10-11).

This section provides the details of the computational complexity for each of these steps.

5.3.1 Reduction of Subcircuits

Finding the reduced models of the partitioned subcircuits requires finding the orthonormal matrices $\mathbf{Q}_l$, $l = 1, \dots, S$. The complexity of finding the orthonormal basis is $O(M_l q^2)$ [67], with M_l being the size of MNA equations of the l^{th} subcircuit. However, the matrices $\mathbf{Q}_l$ do not depend on the iterations and therefore can be found

Algorithm 5.1: TL analysis via OP and MOR

input: The Transmission lines Structure

```

1 begin
2   Partition the lines into subcircuits;
3   Find the reduced model for the subcircuits using (5.7);
4    $iter \leftarrow 1$ ;
5   Assume an initial guess for the relaxation sources ;
6   while No Convergence do
7     for  $j \leftarrow 1$  to  $S$  do
8       Find the Thevenin equivalent sources by finding the open circuit
       voltages at the terminals of each subcircuit using (5.2) and (5.3)
       and store the solution  $\mathbf{x}_{oc,l}$  (Figure 5.1) ;
9       Simulate the reduced models with the Thevenin sources and
       nonlinear terminations using (5.7) and store  $\hat{\mathbf{x}}_{th,l}$  (Figure 5.2);
10      Find the auxiliary voltages/currents ( $\tilde{\mathbf{i}}_l$  and  $\tilde{\mathbf{v}}_l$ ) using
       (5.8) and (5.9);
11      Update the relaxation sources using (4.14)-(4.20);
12       $iter \leftarrow iter + 1$ ;

```

only once prior to the simulation process. It is also worth emphasizing that size of the reduced model is proportional to the number of ports [68]. Thus, applying MOR to each subcircuit results in more compact models than applying MOR to the complete network.

5.3.2 Solving the Complete Network Containing Reduced Subcircuits

This step constitutes the bulk of the CPU effort in the MOR-based OP method. In this step, the reduced models of each subcircuit are simulated for the complete time interval of interest along with the terminal devices attached to the lines. In the case of having nonlinear devices attached to the terminals, Newton-Raphson (NR) iterations are used to solve the nonlinear equations at each time step. For each NR iteration,

LU factorization and Forward/Backward (F/B) substitution are required to solve the resulting system of equations. Applying MOR reduces the size of each subcircuit from $M_l \times M_l$ to $q_l \times q_l$, where $q_l \ll M_l$. In this case, the LU factorization and F/B substitution during each iteration are only applied to the $q_l \times q_l$ subsystems.

5.3.3 Updating the Relaxation Sources

The relaxation sources are updated by first finding the auxiliary voltages/currents using (5.8) and (5.9). This step requires matrix-vector multiplication ($\mathbf{Q}_l \hat{\mathbf{x}}_{th,l}$) and addition, represented by $(\mathbf{x}_{oc,l} + \mathbf{Q}_l \hat{\mathbf{x}}_{th,l})$. Since the size of $\mathbf{Q}_l$ is $M_l \times q_l$, the total complexity of this step is $O(M_l q_l)$, for $l = 1, \dots, S$. It is worth highlighting that multiplying by the selector matrices $\mathbf{P}_{p,l}$ and $\mathbf{W}_{p,l}$ can be done very efficiently by only selecting the values corresponding to the nonzero columns and rows (i.e., no numerical multiplication is required).

The relaxation sources are then updated using (4.19) and (4.20). This step is done on the scalar components of the auxiliary voltages/currents ($\tilde{\mathbf{i}}_l$ and $\tilde{\mathbf{v}}_l$) independently, and thus contributes only a very minor overhead to the total CPU-time. Next, the equivalent sources attached at the end of the lines are found by solving each subcircuit with open circuit terminations, while turning on the distributed relaxation sources along the lines. Using the open circuit at the terminals results in a completely linear circuit representing each subcircuit. Therefore, this step involves solving linear MNA equations and thus, the CPU-time required is insignificant compared to the simulation time for the case of nonlinear terminations.

5.4 Numerical Examples

In this section, we demonstrate the efficiency and accuracy of the MOR-based OP method. The examples were simulated using an in-house simulator developed in C++ and run on a Linux machine. The in-house simulator is used to facilitate the updating of the relaxation sources as opposed to communicating with an external commercial simulator (such as HSPICE [61]) at each iteration.

It is worth highlighting that increasing the number of overlapped lines increases the accuracy for a given number of iterations (or alternatively reduces the required number of iterations for a given accuracy). On the other hand, having too many overlapped lines may lead to an increase in the total CPU time.

5.4.1 Example 1

Figure 5.3: Interconnect network (Example 1)

In this example, the MOR-based OP method is applied to a network with 3 interconnect modules, each of which contains 64 tightly coupled lines, as shown in Figure 8.3. The length of the lines in each module is 6 cm. A 3-V source, with 0.1ns

fall/rise times and 10ns pulse width is attached at the near end of every fourth line of module-A. The uniform lumped segmentation [47] method is used to discretize Telegrapher's equations of each module, with 80 lumped sections.

For comparison purposes, first, the time-domain simulation is performed on the resulting circuit using full simulation without any grouping (*referred to as "SPICE simulation"*) and applying MOR to the entire circuit without any grouping (*referred to as "Conventional MOR"*). Corresponding CPU times are given in Table 5.1.A. As can be seen from the table, the speed-up achieved using conventional MOR is marginal (only 1.35).

It is to be noted that, no convergence is obtained when the circuit is simulated using normal WR-TP (i.e., one line per subcircuit and with no OP: represented as 1/0), which indicates the tight coupling between the lines.

Next, the circuit is simulated using two other approaches, WR-TP with OP but without any reduction of the subcircuits (*referred to as "OP without MOR"*) and the proposed algorithm (i. e., WR-TP with OP and MOR). Corresponding performance is summarized in Tables 5.1.B & 5.1.C, respectively. In order to show the advantages and flexibility of the OP method, different grouping schemes are applied to the circuit. For example, in order to overcome the strong coupling between the lines, the lines are grouped into 3 lines/subcircuit with one overlapped line (represented as 3/1). Another considered grouping is 4 lines/subcircuit with two overlapped lines (represented as 4/2). In these tables, the indicated iterations correspond to a target RMS error of 0.15% (with the norm of the error being calculated at the output of line 2 of module-B).

Figure 5.4 and 5.5 show a sample comparison of time-domain responses using the OP method (dotted line), OP without MOR (dashed line) (both obtained for the case of 4/2, and as can be seen they are indistinguishable) and SPICE simulation (solid line).

As can be seen from Table 5.1, the MOR-based OP method yields superior speed-up and convergence compared to existing approaches.

Table 5.1: CPU-time and performance comparison for Example 1
Table 5.1.A: Performance of SPICE and Conventional MOR based Methods

Method	CPU-time	Speedup w.r.t SPICE sim.
SPICE Simulation	944.06 sec	-
Conventional MOR	697.59 sec	1.35

Table 5.1.B: Performance of the OP without MOR Method

#lines/Overlapped-lines	Iterations	CPU-time	Speedup w.r.t. SPICE
1/0	No Convergence	-	-
3/1	5	282.65 sec	3.34
4/2	3	271.19 sec	3.48

Table 5.1.C: Performance of the MOR-based OP Method

#lines/Overlapped-lines	Iterations	CPU-time	Speedup w.r.t. SPICE
1/0	No Convergence	-	-
3/1	5	122.56 sec	7.70
4/2	3	109.89 sec	8.59

5.4.2 Example 2

In this example, the OP algorithm is applied to a 24 coupled line circuit with nonlinear terminations (Figure 5.6). A CMOS inverter is attached to the near-ends of every fifth line. The network is excited with a 1.8V trapezoidal pulse voltage source, with

Figure 5.4: Time-domain response at the far-end of line 2 in module-B (Example 1)

Figure 5.5: Time-domain response at the far-end of line 1 in module-C (Example 1)

Figure 5.6: Interconnect network with non-linear terminations (Example 2)

Figure 5.7: Time-domain response at the near-end of victim line 3 (Example 2)

0.1ns rise/fall times, and pulse width of 10ns. The lines were modeled with uniform lumped segmentation using 80 sections.

Table 5.2 compares the performances of four different approaches. Here, Table 5.2.A provides a performance comparison of SPICE and Conventional MOR based simulations. Tables 5.2.B and 5.2.C provide the performance details of OP without MOR and the OP method (OP+MOR), respectively. In these experiments, different grouping schemes are considered: the lines are grouped into 3 lines/subcircuit with one overlapped line (represented as 3/1) and 4 lines/subcircuit with two overlapped lines (represented as 4/2). In these tables, the indicated iterations correspond to a target RMS error of 0.1% (with the error being calculated at the near-end of line 2). Figure 5.7 shows a sample comparison of time-domain responses using the OP method (dotted line), OP without MOR (dashed line) (both obtained for the case of 4/2 and are indistinguishable) and SPICE simulation (solid line).

As can be seen from Table 5.2, the MOR-based OP method yields superior speed-up and convergence compared to existing approaches.

Table 5.2: CPU-time and performance comparison for Example 2
Table 5.2.A: Performance of SPICE and Conventional MOR based Methods

Method	CPU-time	Speedup w.r.t SPICE sim.
SPICE Simulation	443.88 sec	-
Conventional MOR	261.10 sec	1.70

Table 5.2.B: Performance of the OP without MOR Method

#lines/Overlapped-lines	Iterations	CPU-time	Speedup w.r.t. SPICE
1/0	No Convergence	-	-
3/1	5	123.18 sec	3.60
4/2	3	111.91 sec	3.96

Table 5.2.C: Performance of the MOR-based OP Method

#lines/Overlapped-lines	Iterations	CPU-time	Speedup w.r.t. SPICE
1/0	No Convergence	-	-
3/1	5	51.90 sec	8.55
4/2	3	48.04 sec	9.23

5.5 Summary

In this chapter, an accelerated overlapping partitioning method is presented. In the case of tightly coupled lines, the interconnects are partitioned such that the tightly coupled lines are grouped in the same subcircuit. The new method exploits the model order reduction techniques to reduce the size of each subcircuit, thus providing a significant speedup in the overall CPU time. Numerical examples were presented, which demonstrate the accuracy and efficiency of the new algorithm.

Chapter 6

Fast Transient Simulation of High-Speed Interconnects using Obreshkov-Formula

As presented in Chapter 3, ObF based methods can provide a high-order convergence without losing stability. However, the system of equations arising from discretizing the MNA equations using ObF is k times larger than the size of MNA equations. Thus, the cost for solving for each time step increases. However, this increase in the per time step cost is offset by the savings in the number of time-points obtained from the high-order methods due to their ability in increasing the time step without losing their accuracy. In this chapter, we show that the cost per time can be reduced significantly for specific types of circuits encountered in high-speed packaging and signal/power integrity applications by exploiting the linear nature of such circuits [28, 29].

A general linear circuit is described in the time-domain using the Modified Nodal

Analysis (MNA), shown in (3.37), as follows

$$\mathbf{G}\mathbf{x}(t) + \mathbf{C}\frac{d\mathbf{x}}{dt} = \mathbf{b}(t) \quad (6.1)$$

As shown in section 3.5, applying Obreshkov-based high-order integration method to numerically approximate the transient solution of the MNA equations in (6.1) at $t = t_{n+1}$ results in the following system of equations

$$(\tilde{\mathbf{G}} + \tilde{\mathbf{C}})\boldsymbol{\xi}_{n+1} = \tilde{\mathbf{b}}_{n+1} \quad (6.2)$$

where $\tilde{\mathbf{G}}$, $\tilde{\mathbf{C}}$, $\boldsymbol{\xi}_{n+1}$ and $\tilde{\mathbf{b}}$ a given (3.38).

Since only linear circuits are considered in this chapter, Newton-Raphson iterations are not required. In this case, the transient simulation of the circuit proceeds by solving the linear system of equations in (6.2) for $n = 1, 2, \dots$ starting from some initial conditions computed at $t = t_0$ [14].

6.1 Development of the ObF-based Transient Analysis for Interconnects

In this section, details of the ObF-based algorithm for efficient analysis of large linear circuits using L -stable and high-order integration methods are given.

Highlighting the main objective of the presented contribution can be best approached by first estimating the computational complexity of the block LU factor-

ization presented in Algorithm 3.1 [16], in order to contrast it with the complexity arising from the proposed algorithm. This task can be easily carried out due to the fact that the matrix $\tilde{\mathbf{M}}$ has the same structure of the typical $N \times N$ MNA matrices, e.g. $\mathbf{G} + 2/h\mathbf{C}$, which arise while using the TR method, except that the entries in the latter are expanded to size k block matrices in $\tilde{\mathbf{M}}$. It is known that the complexity of a single standard LU factorization for a typical matrix with MNA-like sparsity and size N grows in proportion to N^α , where α is a factor that is approximately equal to 1.2, as has been reported recently in [69]. Hence, using a sparse ordering for the block entries of $\tilde{\mathbf{M}}$ and noting that each scalar operation in the standard LU factorization is replaced by a matrix block operation of the same kind, it is reasonable to conclude that the cost of block LU factorization will scale in proportion to $N^\alpha k^3$. This estimate of the computational complexity is typically denoted by $O(N^\alpha k^3)$.

It is important to note that the increase in the computational effort from $O(N^\alpha)$ in a low-order method (such as the TR) compared to $O(N^\alpha k^3)$ in a ObF-based high-order method is part of the computational overhead which is involved naturally in any high-order scheme for numerical solution of DAE. However, it is the saving in the number of time steps, which is made possible by the larger step sizes, that typically offsets this overhead and results in an overall gain in the simulation time.

The central contribution of this chapter aims at reducing the complexity of the overhead from $O(N^\alpha k^3)$ to $O(N^\alpha k)$ for the case of linear circuits. The proposed algorithm takes advantage of the special structure of the blocks in $\tilde{\mathbf{M}}$ to achieve this goal.

Consider the system in (6.2) rewritten in the form

$$\tilde{\mathbf{M}}\boldsymbol{\xi}_{n+1} = \tilde{\mathbf{b}}_{n+1} \quad (6.3)$$

An arbitrary block in $\tilde{\mathbf{M}}$ can be written as

$$[\tilde{\mathbf{M}}]_{(i,j)} = g_{i,j}\mathbf{I} + \frac{c_{i,j}}{h}\mathbf{T} \quad (6.4)$$

Using an eigen-decomposition of $\tilde{\mathbf{M}}_{(i,j)}$, each block can be represented as [28, 29]

$$[\tilde{\mathbf{M}}]_{(i,j)} = \mathbf{V}\boldsymbol{\lambda}_{(i,j)}\mathbf{V}^{-1} \quad (6.5)$$

where $\mathbf{V} \in \mathbb{R}^{k \times k}$ is the matrix of eigen-vectors and $\boldsymbol{\lambda}_{(i,j)}$ is a diagonal matrix having the eigen-values of $\tilde{\mathbf{M}}_{(i,j)}$ as its diagonal elements and can be computed as

$$\boldsymbol{\lambda}_{(i,j)} = g_{i,j}\mathbf{I} + \frac{c_{i,j}}{h}\boldsymbol{\lambda}_T \quad (6.6)$$

where $\boldsymbol{\lambda}_T$ is a diagonal matrix containing the eigen-values of $\mathbf{T}$

The key idea in the proposed algorithm is motivated by the observation that the matrix $\mathbf{V}$ is independent from the block specific entries $g_{i,j}$, $c_{i,j}$ and the step size h . Therefore, the matrix $\mathbf{V}$ is shared by all the blocks in $\tilde{\mathbf{M}}$.

In fact, $\mathbf{V}$ depends only on the value of integer k , and is available in closed-form, or can be computed offline prior to any computation. Indeed, this fact enables rewriting (6.3) as follows.

$$\Upsilon\tilde{\mathbf{A}}\Upsilon^{-1}\boldsymbol{\xi}_{n+1} = \tilde{\mathbf{b}}_{n+1} \quad (6.7)$$

where

$$\Upsilon = \underbrace{\begin{bmatrix} \mathbf{V} & & \\ & \ddots & \\ & & \mathbf{V} \end{bmatrix}}_{Nk \times Nk} \quad (6.8)$$

and $\tilde{\mathbf{A}}$ has the same block-structured form of $\tilde{\mathbf{M}}$, but with the exception that each block is now a diagonal block, whose diagonal entries are given by $g_{i,j} + \frac{c_{i,j}}{h}\lambda_p$, with λ_p being the p^{th} eigen-value of $\mathbf{T}$ and $p = 1, \dots, k$.

To take advantage of the formulation in (6.7), we use the following change of variables

$$\boldsymbol{\rho}_{n+1} = \Upsilon^{-1} \boldsymbol{\xi}_{n+1} \quad (6.9)$$

Then, the system of equations in (6.7) can be written as

$$\tilde{\mathbf{A}} \boldsymbol{\rho}_{n+1} = \Upsilon^{-1} \tilde{\mathbf{b}}_{n+1} \quad (6.10)$$

Solving the system in (6.10) for $\boldsymbol{\rho}_{n+1}$ can be carried out using the block LU factorization. However, in contrast to the previous formulation in (6.3), the blocks in $\tilde{\mathbf{A}}$ of (6.10) are all diagonal, therefore requiring only at most k computations during any block operation such as inversion, multiplication or addition. This fact implies that the overall computational complexity will be $O(N^\alpha k)$ as opposed to the $O(N^\alpha k^3)$ that is incurred in the previous approaches [16].

In order to emphasize the advantage of this new formulation, the pseudo-code pre-

sented in Algorithm 3.1 is modified to use $\tilde{\mathbf{A}}$ instead of $\tilde{\mathbf{J}}$, as shown in Algorithm 6.1.

Algorithm 6.1: Modified Block-LU factorization

```

input  :  $\tilde{\mathbf{A}} \in \mathbb{R}^{kN \times kN}$ 
output:  $\tilde{\mathbf{L}}, \tilde{\mathbf{U}} \in \mathbb{R}^{kN \times kN}$ 
1 begin
2    $\mathbf{L} \leftarrow \mathbf{I}_{kN}$ ;
3   for  $p \leftarrow 1$  to  $N$  do
4      $r_1 \leftarrow 1, r_2 \leftarrow kN$ ;
5      $c_1 \leftarrow (p-1)k+1, c_2 \leftarrow pk$ ;
6      $\mathbf{y} \leftarrow \text{BFS}(\mathbf{L}, \tilde{\mathbf{A}}(r_1:r_2, c_1:c_2))$ ;
7      $r_2 \leftarrow pk$ ;
8      $\mathbf{U}(r_1:r_2, c_1:c_2) \leftarrow \mathbf{y}(r_1:r_2, c_1:c_2)$ ;
9      $\mathbf{H} \leftarrow \text{BPivot}(\mathbf{y})$ ; //Choose the block pivot
10    for  $q \leftarrow p+1$  to  $N$  do
11       $r_1 \leftarrow (q-1)k, r_2 \leftarrow qk$ ;
12       $\mathbf{L}(r_1:r_2, c_1:c_2) \leftarrow \mathbf{y}(r_1:r_2, c_1:c_2) \mathbf{H}^{-1}$ 

```

Algorithm 6.2: Modified BFS (Block Forward Substitution)

```

input  :  $\mathbf{L} \in \mathbb{R}^{kN \times kN}, \mathbf{A} \in \mathbb{R}^{kN \times k}$ 
output:  $\mathbf{y} \in \mathbb{R}^{kN \times k}$ 
1 begin
2    $\mathbf{y} \leftarrow \mathbf{A}$ ;
3   for  $p \leftarrow 2$  to  $N$  do
4      $r_1 \leftarrow (p-1)k+1, r_2 \leftarrow pk$ ;
5     for  $q \leftarrow 1$  to  $p-1$  do
6        $c_1 \leftarrow (q-1)k+1, c_2 \leftarrow qk$ ;
7        $\mathbf{y}(r_1:r_2, :) \leftarrow \mathbf{y}(r_1:r_2, :) -$ 
8          $\mathbf{L}(r_1:r_2, c_1:c_2) \mathbf{y}(c_1:c_2, :)$ 

```

The pseudo-code illustrations make it clear that the basic computational effort in the Block LU (BLU) factorization involves matrix-matrix multiplication (step 7 in Algorithm 6.2 and in Algorithm 6.3), matrix inversion (step 12 in Algorithm 6.1 and step 15 in Algorithm 6.3) and matrix-vector multiplications for the matrices of

Algorithm 6.3: Modified Block Forward/Backward Substitution

```

input  :  $\tilde{L}, \tilde{U} \in \mathbb{R}^{kN \times kN}, \tilde{b} \in \mathbb{R}^{kN \times k}$ 
output:  $\rho \in \mathbb{R}^{kN \times k}$ 
1 begin
2    $y \leftarrow \tilde{b};$ 
3   for  $p \leftarrow 2$  to  $N$  do
4      $r_1 \leftarrow (p-1)k+1, r_2 \leftarrow pk;$ 
5     for  $q \leftarrow 1$  to  $p-1$  do
6        $c_1 \leftarrow (q-1)k+1, c_2 \leftarrow qk;$ 
7        $y(r_1:r_2, :) \leftarrow y(r_1:r_2, :) -$ 
8          $L(r_1:r_2, c_1:c_2) y(c_1:c_2, :)$ 
9    $\rho \leftarrow y;$ 
10  for  $p \leftarrow N$  to  $1$  do
11     $r_1 \leftarrow (p-1)k+1, r_2 \leftarrow pk;$ 
12    for  $q \leftarrow 1$  to  $p-1$  do
13       $c_1 \leftarrow (q-1)k+1, c_2 \leftarrow qk;$ 
14       $D \leftarrow U(r_1:r_2, r_1:r_2);$ 
15       $\rho(r_1:r_2, :) \leftarrow D^{-1}(\rho(r_1:r_2, :) -$ 
16         $U(r_1:r_2, c_1:c_2) \rho(c_1:c_2, :))$ 

```

the individual blocks which are diagonal with size k . Therefore, each one of those computations can be carried out with complexity that scales as k per block.

The memory required to save the matrix resulting from applying the ObF-based method would be k times larger than conventional TR method. However, it is to be noted that, in the case of the proposed method, it is only required to store the diagonal blocks. Therefore, a smaller memory is required to factorize the matrix than if the entries are treated as full blocks. The LU factorization of the matrix using the proposed method is $O(kN^\alpha)$, where k is the order of integration and N is the size of unknown variables. This minor increase in the memory and factorization costs are far outweighed by the significant CPU-time savings using the proposed method due to its ability to take larger time-steps.

6.2 Implementation Strategies

It is important to stress that solving for $\boldsymbol{\rho}_{n+1}$ in (6.10) through the block LU factorization of $\tilde{\mathbf{A}}$ is only one component of the transient simulation. However, in order to maintain the same complexity of $O(N^{\alpha}k)$ in the other computational components involved, several implementation issues have to be carefully considered. This section discusses those issues.

In addition to solving for $\boldsymbol{\rho}_{n+1}$, the basic time stepping mechanism involves several other steps. More precisely, those steps are

1. The mapping from $\boldsymbol{\rho}_{n+1}$ to $\boldsymbol{\xi}_{n+1}$,
2. Computing the new value for the source vector $\tilde{\mathbf{b}}_{n+1}$, and
3. Computing the right hand-side vector $\boldsymbol{\Upsilon}^{-1}\tilde{\mathbf{b}}_{n+1}$ in (6.10).

The following subsections address handling the above computational issues efficiently.

6.2.1 Mapping $\boldsymbol{\rho}_{n+1}$ to $\boldsymbol{\xi}_{n+1}$

It is obvious that $\boldsymbol{\xi}_{n+1}$ represent the physical variables, in which a designer is typically interested, whereas $\boldsymbol{\rho}_{n+1}$ represents a transformation of those variables. Clearly mapping from $\boldsymbol{\rho}_{n+1}$ to $\boldsymbol{\xi}_{n+1}$ requires multiplying the former by $\boldsymbol{\Upsilon}$ to obtain the latter. This process, however, will scale as $O(Nk^2)$ and could, therefore, make the complexity of the overall algorithm proportional to Nk^2 .

However, it is typically the case that the designer is interested in viewing the response at a small set of selected variables, e.g, the voltage at the receiving end of a victim line in a huge interconnect network. The knowledge of those voltages and currents can be utilized to make sure that only those variables in the $\boldsymbol{\rho}_{n+1}$ space are mapped using $\mathbf{V}$ to the $\boldsymbol{\xi}_{n+1}$ space. This fact can be used to limit the transformation from $\boldsymbol{\rho}_{n+1}$ to $\boldsymbol{\xi}_{n+1}$ to only those variables indicated by the designer.

6.2.2 Computing $\tilde{\mathbf{b}}$

As shown in (3.46), computing $\tilde{\mathbf{b}}$ requires having access to all the components in $\boldsymbol{\xi}_{n+1}$. Unlike the previous issue, computing the full vectors of $\mathbf{x}_n^{(i)}$ is mandatory to form $\tilde{\mathbf{b}}_{n+1}$, thereby, requiring an $O(Nk^2)$ computation. In order to address this issue, next, we show that by adopting a careful ordering strategy, the cost of this computational step can be reduced to just $O(Nk)$ [28].

To this end, we note that the relation between $\boldsymbol{\rho}_{n+1}$ and $\boldsymbol{\xi}_{n+1}$ is through the transformation

$$\boldsymbol{\xi}_n = \Upsilon \boldsymbol{\rho}_n \quad (6.11)$$

Using the fact that $\boldsymbol{\xi}_n$ is partitioned into N vectors, with k components each, i.e.,

$$\boldsymbol{\xi}_n = [x_{n,1}^{(0)} \dots h^{k-1} x_{n,1}^{(k-1)} \dots \dots x_{n,N}^{(0)} \dots h^{k-1} x_{n,N}^{(k-1)}]^T \quad (6.12)$$

and adopting a similar partitioning to $\boldsymbol{\rho}_n$,

$$\boldsymbol{\rho}_n = [\rho_{n,1}^{(0)} \dots h^{k-1} \rho_{n,1}^{(k-1)} \dots \rho_{n,N}^{(0)} \dots h^{k-1} \rho_{n,N}^{(k-1)}]^T \quad (6.13)$$

shows that

$$h^j x_{n,p}^{(j)} = \sum_{i=0}^{k-1} V_{j,i} h^i \rho_{n,p}^{(i)} \quad (6.14)$$

where $V_{j,i}$ is component at the j^{th} row and i^{th} column in the matrix $\mathbf{V}$.

Now, consider the vector $\sum_{i=0}^m \beta_i h^i \mathbf{x}_n^{(i)}$ needed to compute $\tilde{\mathbf{b}}_{n+1}$. This vector can be formulated in the following form

$$\sum_{i=0}^m \beta_i h^i \mathbf{x}_n^{(i)} = \begin{bmatrix} \sum_{i=0}^m \beta_i h^i x_{n,1}^{(i)} \\ \vdots \\ \sum_{i=0}^m \beta_i h^i x_{n,N}^{(i)} \end{bmatrix} \quad (6.15)$$

Substituting from (6.14) into (6.15) shows that

$$\sum_{i=0}^m \beta_i h^i \mathbf{x}_n^{(i)} = \begin{bmatrix} \sum_{l=0}^m \beta_l \sum_{i=0}^{k-1} V_{l,i} h^i \rho_{n,1}^{(i)} \\ \vdots \\ \sum_{l=0}^m \beta_l \sum_{i=0}^{k-1} V_{l,i} h^i \rho_{n,N}^{(i)} \end{bmatrix} \quad (6.16)$$

Interchanging the double summation yields

$$\sum_{i=0}^m \beta_i h^i \mathbf{x}_n^{(i)} = \begin{bmatrix} \overbrace{\sum_{i=0}^{k-1} \sum_{l=0}^m \beta_l V_{l,i} h^i \rho_{n,1}^{(i)}}^{\Psi_i} \\ \vdots \\ \underbrace{\sum_{i=0}^{k-1} \sum_{l=0}^m \beta_l V_{l,i} h^i \rho_{n,N}^{(i)}}_{\Psi_i} \end{bmatrix} \quad (6.17)$$

where Ψ_i is defined as

$$\Psi_i = \sum_{l=0}^m \beta_l V_{l,i} \quad (6.18)$$

and needs to be evaluated only once before the beginning of the transient simulation.

Equation (6.17) shows that the vector $\sum_{i=0}^m \beta_i h^i \mathbf{x}_n^{(i)}$ can be computed using Nk multiplication and addition, thereby making the overall complexity of this step of $O(Nk)$.

6.2.3 Computing the right hand-side vector

In order to solve for $\boldsymbol{\rho}_{n+1}$ in (6.10), the vector $\tilde{\mathbf{b}}_{n+1}$ is multiplied by $\mathbf{\Upsilon}^{-1}$ to form the right hand-side. This multiplication scales to $O(N^\alpha k^2)$ and may increase the overall complexity of the algorithm.

However, the vector $\tilde{\mathbf{b}}_{n+1}$ represents the independent sources in the circuit and in most practical circuits, the number of sources is very small compared to the number of nodes. Utilizing this fact, the multiplication $\mathbf{\Upsilon}^{-1} \tilde{\mathbf{b}}$ can be computed in almost $O(N^\alpha k)$ by mapping the few entries in $\tilde{\mathbf{b}}$ due to the independent stimuli and multiplying only the last component in $\tilde{\mathbf{b}}_{n+1,i}$ by the last column in $\mathbf{V}$ for all $i = 1, \dots, N$. This strategy

keeps the computational complexity very close to $O(N^\alpha k)$

6.3 Numerical Examples

This section presents the results of the numerical simulations using the ObF-based algorithm presented in this chapter. We compare the CPU time taken by the ObF-based algorithm versus the time taken by the low-order methods used in traditional SPICE-like simulators. For the latter, we use the TR which represents the most accurate A -stable method in the whole class of LMS method [70]. It should be noted that the ObF-based algorithm reduces to the TR when executed with $k = m = 1$, in which case the order becomes equal to 2. We also employ the notation of (i/j) to denote running the ObF-based algorithm using $m = i$ and $k = j$, thereby making the order equal to $i + j$.

6.3.1 Example 1

The circuit considered in this example is a coupled two-conductor TL whose physical cross-section is shown in Figure 6.1 and its length is 8cm. The p.u.l parameters of the coupled lines are extracted using field solvers in HSPICE [61]. The lines are then modeled using a uniform lumped segmentation approach [47] with 80 lumped sections for each line.

Table 6.1 summarizes the number of time points and CPU time taken by the ObF-based technique at different orders, starting with order 2 which represents the TR. The last column shows the speedup achieved by the high-order method relative

Figure 6.1: Interconnects circuit for Example 1

to the low-order of the TR.

Figure 6.2 shows a sample time-domain waveform at the far-end of the victim line (second line) using different orders. As seen from Table 6.1, the number of time points required decreases significantly at higher orders.

Figure 6.2: Waveform comparison at the far-end of victim line 2 (Example 1)

Table 6.1: CPU time comparison of Example 1

Order	Number of time steps	ObF-based method (sec)	Speedup w.r.t TR
2(TR)	480	0.18	1.00
4	170	0.07	2.57
6	140	0.04	4.5
8	105	0.03	6.00
10	85	0.028	6.43
12	65	0.021	8.57

6.3.2 Example 2

In this example, we consider a coupled interconnects circuit shown in Figure 6.3. The length of the lines is 6cm. The coupled lines are modeled using lumped segmentation approach [47] (40 sections). The lines are excited with a voltage source attached to each fourth line, with 0.5ns fall/rise time, 10ns pulse width and 3.5V in magnitude. The size of the resulting matrix is $Nk \times Nk$, with $N = 8000$ and k representing the order of the method being used.

To further illustrate the computational advantage of the ObF-based method, Figure 6.4 shows the CPU time of one LU matrix factorization (versus k) using the ObF-based algorithm, and contrasts it with the regular block LU factorization used previously [16]. For $k > 1$, this figure basically depicts the major part of the additional computations incurred from the high-order scheme, when compared with low-order ($k = 1$). It should be noted, however, that this computational growth is a part of an overhead needed to implement the high-order scheme. However, it is usually the savings in the number of time points, made possible by the larger step sizes, that

significantly offsets this overhead and results in an overall saving of computational time.

What Figure 4 clearly demonstrates is that the ObF-based algorithm makes the computational overhead grow linearly with the order of the method k , which agrees with the analysis presented in the previous section as regards the computational complexity. This should be contrasted against the quadratic growth in computational overhead that results from applying a regular Block LU factorization [16] without taking advantage of the special structure of the underlying matrix blocks.

Table 6.2 summarizes the results obtained when the ObF-based algorithm is executed in its L -stable mode ($k - 2 \leq m < k$). It also shows the achieved speedup relative to the low-order L -stable method Backward Euler (BE), which is obtained as the special case at $m = 0, k = 1$.

Figure 6.5 shows a sample time-domain waveform at the near-end of the aggressor line 12 using different integration orders. As can be observed, the results obtained from the high-order can approximate the waveforms at a much lower number of time points.

Table 6.2: CPU time comparison of Example 2 for L -stable methods

Order	Number of time steps	ObF-based method (sec)	Speedup w.r.t BE
0/1(BE)	550	215.41	1.00
1/3	285	99.99	2.15
2/4	105	57.71	3.73
3/5	67	46.56	4.63
4/6	35	30.55	7.05

Figure 6.3: Interconnects circuit for Example 2

Figure 6.4: CPU time comparison of one LU factorization (Example 2)

Figure 6.5: Waveform comparison at the near-end of aggressor line 12 (Example 2)

6.3.3 Example 3

The interconnect network shown in Figure 6.6 is considered for this example. The length of the lines is 4 cm. The coupled lines are modeled using lumped segmentation approach [47] (40 sections). The circuit is excited with a current source attached to each fourth line in segment 1, with 0.5ns fall/rise time, 10ns pulse width and 0.06A in magnitude.

Figure 6.6: Large coupled interconnects network (Example 3)

Figure 6.7 shows a sample comparison of time domain responses at far-end of line 6 in segment 2 using the presented ObF-based method with different integration orders.

Table 9.4 shows the total CPU time required to perform the transient simulation with different orders. Column three shows the speedup obtained using the presented ObF-based method compared to the conventional TR method.

Figure 6.7: Waveform comparison at the far-end of victim line 6 in segment 2 (Example 3)

Table 6.3: CPU time comparison of Example 3

Order	Number of time steps	ObF-based method (sec)	Speedup w.r.t TR
2(TR)	205	261.09	1.00
4	102	78.33	3.33
6	81	57.86	4.51
8	67	40.35	6.47
10	51	31.28	8.34
12	33	26.84	9.72

6.3.4 Example 4

In this example, we consider an interconnect circuit characterized by tabulated data of the S-parameters at different frequency points. The data is first fitted using a vector-fitting algorithm [58], then the results are synthesized using linear components. The input ports are excited with voltage sources of 0.1ns rise/fall time, 10ns pulse width and 3V of magnitude. The output ports are terminated with linear capacitors of 1pF, as shown in Figure 6.8. The CPU time comparison is shown in Table 6.4. In addition,

Figure 6.9 shows the time-domain comparison of the waveforms at port 2 using the TR method, order 4 and order 6.

Figure 6.8: Eight ports linear circuit (Example 4)

Table 6.4: CPU time comparison of Example 4

Order	Number of time steps	ObF-based method (sec)	Speedup w.r.t TR
2(TR)	652	2.35	1
4	306	1.12	2.09
6	204	0.92	2.55
8	101	0.60	3.91
10	62	0.32	7.34
12	35	0.22	10.68

6.4 Summary

In this chapter, an efficient algorithm that accelerates the performance of the Obreshkov-based high-order methods, for simulation of large linear circuits that typically arise from high-speed interconnects and packages. The core of the presented technique is a new algorithm to perform the block factorization of the system matrix. This factorization takes advantage of the special matrix structure that arises from the linearity of the underlying circuit. It was shown, analytically and through numerical

Figure 6.9: Waveform comparison at port 2 (Example 4)

simulations, that the presented technique in this chapter reduces the factorization complexity from $O(N^\alpha k^3)$ to only $O(N^\alpha k)$, where N is the size of the circuit MNA formulation and k is an integer that corresponds to the order used.

Chapter 7

Fast Transient Simulation of High-Speed Interconnects with Non-Linear Terminations Using Obreshkov-Formula

The previous chapter showed how the linearity of the circuit can be exploited to offer an additional computational advantage for the Obreshkov-based circuit simulation. This chapter shows how this advantage can be maintained when the circuit is terminated by nonlinear devices, such as buffers or digital circuitry [30,31].

Typically, the presence of nonlinear devices in the circuit is accommodated in the general MNA formulation via adding the term $\mathbf{f}(\mathbf{x}(t))$ to the left-side of (6.1), i.e.,

$$\mathbf{G}\mathbf{x}(t) + \mathbf{C}\frac{d\mathbf{x}(t)}{dt} + \mathbf{f}(\mathbf{x}(t)) = \mathbf{b}(t) \quad (7.1)$$

Here $\mathbf{f}(\mathbf{x}(t))$ is a vector of nonlinear functions representing the currents supplied from or drawn into the nonlinear devices. Representing dynamic nonlinearities such as nonlinear capacitors or inductors is carried out using the Charge-Oriented formu-

lation, which incorporates the charge or flux as an additional variable, while adding the nonlinearity describing their relation to the terminal voltage as an extra entry to $\mathbf{f}(\mathbf{x}(t))$ [71].

In order to handle the new formulation while treating the linear part using the approach presented in the previous section, we use the notion of “node tearing” [72] to divide the circuit into several partitions, with the objective of isolating the nonlinear devices in independent partitions, while keeping the linear subcircuit in a separate partition. For the sake of clarity, this idea is first explained by considering a simple circuit.

7.1 Node Tearing in an Illustrative Circuit

Consider the circuit shown in Figure 7.1, which represents a transmission line modeled using an RLC tree and terminated by a CMOS inverter. As shown in the lower part of this figure, the concept of node tearing is applied to this circuit by splitting the node at the interface between the linear subcircuit and the nonlinear inverter into two nodes and adding a short-circuit link between these two nodes. This process results in adding two additional variables to the MNA formulation in (7.1); the first one is the voltage of the extra node (p'), while the other one is the current flowing through the short-circuit link, γ . In addition, it endows the MNA formulation with a special structure that is key to efficient analysis.

To visualize this structure, we assume, without loss of generality, that the MNA variables pertaining to the nonlinear CMOS inverter have been ordered so that they

Figure 7.1: Illustration of the notion of Node Tearing applied to a simple circuit.

come at the end of the MNA variables of the entire circuit, followed by the current in the short-circuit link γ [71]. This ordering entails an MNA in the following form

$$\begin{bmatrix} \mathbf{G}_{\text{lin}} & \mathbf{0} & \mathbf{\Xi}_1 \\ \mathbf{0} & \mathbf{G}_{\text{inv}} & \mathbf{\Xi}_2 \\ \mathbf{\Xi}_1^\top & \mathbf{\Xi}_2^\top & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\text{lin}} \\ \mathbf{x}_{\text{inv}} \\ \gamma \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{\text{lin}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{\text{inv}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}}_{\text{lin}} \\ \dot{\mathbf{x}}_{\text{inv}} \\ \dot{\gamma} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{f}(\mathbf{x}_{\text{inv}}(t))_{\text{inv}} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\text{lin}} \\ \mathbf{b}_{\text{inv}} \\ 0 \end{bmatrix} \quad (7.2)$$

where the subscripts “lin” and “inv” are added to denote whether a structure belongs to the linear or inverter partition, respectively, $\mathbf{\Xi}_1$ and $\mathbf{\Xi}_2$ are vectors with zeros except at the entry corresponding to the node p (p'), where it has +1 (respectively, -1) and $\top$ denotes the matrix transposition operator. To facilitate the rest of the mathematical development, we assume that $\mathbf{G}_{\text{inv}}, \mathbf{C}_{\text{inv}} \in \mathbb{R}^{P \times P}$ and $\mathbf{G}_{\text{lin}}, \mathbf{C}_{\text{lin}} \in \mathbb{R}^{N \times N}$. Appendix B presents a simple example to further illustrate the above partitioning

technique for a simple circuit having $N = 2, P = 2$.

7.1.1 Obreshkov-based Discretization

Next, consider applying the discretization rule prescribed by the Obreshkov formula in (3.32) to the nonlinear MNA (7.1), exploiting the structure revealed by (7.2). As shown in [14], this process transforms the differential system into an algebraic set of nonlinear augmented system of equations,

$$\begin{bmatrix} \tilde{\mathbf{M}}_{\text{lin}} & \mathbf{0} & \tilde{\mathbf{\Xi}}_1 \\ \mathbf{0} & \tilde{\mathbf{M}}_{\text{inv}} & \tilde{\mathbf{\Xi}}_2 \\ \tilde{\mathbf{\Xi}}_1^\top & \tilde{\mathbf{\Xi}}_2^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}_{\text{lin}n+1} \\ \boldsymbol{\xi}_{\text{inv}n+1} \\ \tilde{\boldsymbol{\gamma}}_{n+1} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \tilde{\boldsymbol{\eta}}_{\text{inv}n+1} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{b}}_{\text{lin}n+1} \\ \tilde{\mathbf{b}}_{\text{inv}n+1} \\ \mathbf{0} \end{bmatrix}, \quad (7.3)$$

where $\tilde{\mathbf{M}}_{\text{lin}} = \tilde{\mathbf{G}}_{\text{lin}} + \tilde{\mathbf{C}}_{\text{lin}}$, $\tilde{\mathbf{M}}_{\text{inv}} = \tilde{\mathbf{G}}_{\text{inv}} + \tilde{\mathbf{C}}_{\text{inv}}$, with the matrices $\tilde{\mathbf{G}}_{\text{lin}}$, $\tilde{\mathbf{C}}_{\text{lin}}$, $\tilde{\mathbf{G}}_{\text{inv}}$ and $\tilde{\mathbf{C}}_{\text{inv}}$ are augmented matrices that are obtained using the similar process as outlined in Section 3.5. It is to be noted that both $\tilde{\mathbf{M}}_{\text{lin}}$ and $\tilde{\mathbf{M}}_{\text{inv}}$ are of size k times that of $\mathbf{G}_{\text{lin}}$ and $\mathbf{G}_{\text{inv}}$, i.e. $kN \times kN$. In addition, $\tilde{\mathbf{\Xi}}_1$ and $\tilde{\mathbf{\Xi}}_2$ are matrices with k columns that are obtained using the similar process as outlined in Appendix B, and may be expressed in closed-form using the Kronecker product as follows, $\tilde{\mathbf{\Xi}}_{1,2} = \mathbf{\Xi}_{1,2} \otimes \mathbf{I}_k$.

Also, the nonlinear term $\tilde{\boldsymbol{\eta}}_{\text{inv}n+1}$ results from the discretization of the inverter circuit nonlinearities, and is described by [14]

$$\tilde{\boldsymbol{\eta}}_{\text{inv}n+1} = [\tilde{\boldsymbol{\eta}}_{n+1,1}^\top \quad \cdots \quad \tilde{\boldsymbol{\eta}}_{n+1,P}^\top]^\top, \quad (7.4)$$

where

$$\tilde{\boldsymbol{\eta}}_{n+1,i} = [\eta_{n+1,i}^{(0)} \quad \eta_{n+1,i}^{(1)} \quad \cdots \quad \eta_{n+1,i}^{(k-1)}]^\top \quad (7.5)$$

and $\eta_{n+1,i}^{(j)} = h^j \frac{d^j}{dt^j} f_i(\mathbf{x}(t))$ computed at t_{n+1} .

7.1.2 Solving the Discretized System

Denote the set of unknowns in (8.6) by $\boldsymbol{\zeta}_{n+1}$. Solving (8.6) for $\boldsymbol{\zeta}_{n+1}$ is done using an iterative Newton-Raphson approach, starting from an initial guess, $\boldsymbol{\zeta}_{n+1}^{(0)}$, and iterating through

$$\boldsymbol{\zeta}_{n+1}^{(r+1)} = \boldsymbol{\zeta}_{n+1}^{(r)} + \Delta \boldsymbol{\zeta}_{n+1}^{(r+1)} \quad (7.6)$$

where the update of the r^{th} iteration $\Delta \boldsymbol{\zeta}_{n+1}^{(r+1)}$ is obtained from

$$\tilde{\mathbf{J}} \Delta \boldsymbol{\zeta}_{n+1}^{(r+1)} = \boldsymbol{\Phi}^{(r)} \quad (7.7)$$

using $\boldsymbol{\Phi}^{(r)}$ (the difference between the right- and left-side of (8.6) arising from the r^{th} iteration), and the Jacobian matrix.

$$\tilde{\mathbf{J}} = \begin{bmatrix} \tilde{\mathbf{M}}_{\text{lin}} & \mathbf{0} & \tilde{\boldsymbol{\Xi}}_1 \\ \mathbf{0} & \tilde{\mathbf{M}}_{\text{inv}} + \tilde{\mathbf{J}}_{\text{inv}} & \tilde{\boldsymbol{\Xi}}_2 \\ \tilde{\boldsymbol{\Xi}}_1^\top & \tilde{\boldsymbol{\Xi}}_2^\top & \mathbf{0} \end{bmatrix} \quad (7.8)$$

where $\tilde{\mathbf{J}}_{\text{inv}}$ can be calculated as presented in Section 3.5 [16].

It must be stressed that $\tilde{\mathbf{M}}_{\text{lin}}$ exhibits the same structure of the matrix $\tilde{\mathbf{M}}$ of linear circuits, which has been analyzed in Chapter 6. As a consequence, factorization of

$\tilde{\mathbf{M}}_{\text{lin}}$ may be carried out using the same efficient technique that takes advantage of this structure, as illustrated by Chapter 6. Nonetheless, one should keep in mind that it is the factorization of $\tilde{\mathbf{J}}$ and not $\tilde{\mathbf{M}}_{\text{lin}}$ that we seek in this situation. The challenge then becomes how to factorize $\tilde{\mathbf{J}}$ without losing the computational advantage facilitated by the structure of $\tilde{\mathbf{M}}_{\text{lin}}$. This is indeed important in situations where the circuit is dominated by a large linear part, such as in microwave and packaging applications.

7.1.3 Efficient Solution Using Node Tearing

Similar to the steps in Chapter 6, $\tilde{\mathbf{M}}_{\text{lin}}$ can be written as the decomposition $\mathbf{\Upsilon} \tilde{\mathbf{\Lambda}}_{\text{lin}} \mathbf{\Upsilon}^{-1}$, hence a change of variables similar to (6.9) can be performed to transform the system (8.8) into

$$\begin{bmatrix} \tilde{\mathbf{\Lambda}}_{\text{lin}} & \mathbf{0} & \mathbf{\Gamma}_1 \\ \mathbf{0} & \tilde{\mathbf{K}}_{\text{inv}} & \tilde{\mathbf{\Xi}}_2 \\ \hat{\mathbf{\Gamma}}_1 & \tilde{\mathbf{\Xi}}_2^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{\rho}_{n+1}^{(r+1)} \\ \Delta \boldsymbol{\xi}_{\text{inv}n+1}^{(r+1)} \\ \Delta \tilde{\boldsymbol{\gamma}}_{n+1}^{(r+1)} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{\Phi}}_1^{(r)} \\ \boldsymbol{\Phi}_2^{(r)} \\ \mathbf{0} \end{bmatrix} \quad (7.9)$$

where $\tilde{\mathbf{K}}_{\text{inv}} = \tilde{\mathbf{M}}_{\text{inv}} + \tilde{\mathbf{J}}_{\text{inv}}$, $\Delta \boldsymbol{\rho}_{n+1}^{(r+1)} = \mathbf{\Upsilon}^{-1} \Delta \boldsymbol{\xi}_{\text{lin}n+1}^{(r+1)}$, $\mathbf{\Gamma}_1 = \mathbf{\Upsilon}^{-1} \tilde{\mathbf{\Xi}}_1$, $\hat{\mathbf{\Gamma}}_1 = \tilde{\mathbf{\Xi}}_1^\top \mathbf{\Upsilon}$, and $\hat{\boldsymbol{\Phi}}_1^{(r)} = \mathbf{\Upsilon}^{-1} \boldsymbol{\Phi}_1^{(r)}$.

The main idea in the node tearing approach utilizes the fact that (7.9) represents the linearization of the circuit at the instant $t = t_{n+1}$, and hence all partitions in the circuit can be viewed as linear subcircuits connected by the links created through the process of node tearing. This fact facilitates decoupling the solution of (7.9) into several stages that can be summarized as follows.

1. First, the voltages and currents (and their h -scaled derivatives) internal to each subcircuit are found, assuming zero currents flowing in the links $\Delta\tilde{\gamma} = \mathbf{0}$. The voltages and currents computed under this assumption are denoted by $\hat{\rho}_{n+1}^{(r+1)}$ and $\hat{\xi}_{\text{inv}n+1}^{(r+1)}$ and follow from the solution of the following systems,

$$\tilde{\Lambda}_{\text{lin}} \hat{\rho}_{n+1}^{(r+1)} = \hat{\Phi}_1^{(r)} \quad (7.10)$$

$$\tilde{K}_{\text{inv}} \hat{\xi}_{\text{inv}n+1}^{(r+1)} = \Phi_2^{(r)} \quad (7.11)$$

The solutions obtained from solving the above systems need to be adjusted for the currents in the links. This is done in the following three steps.

2. The impedance matrix of the multiport network whose ports are the link nodes is computed. This matrix is obtained by applying unity current sources having unity h -scaled derivatives at the nodes of the link (p and p') while deactivating independent sources in each subcircuit. The obtained terminal voltages represent the impedance matrix, which provides the relation between the input currents, and its h -scaled derivatives, to the output voltages and its h -scaled derivatives. In the case of the simple circuit considered by Figure 7.1, such a matrix will be a $k \times k$ matrix with two components contributed from the two subcircuits. The first component is the one obtained from

the inverter subcircuit and is given by $\tilde{\mathbf{\Xi}}_2^\top \tilde{\mathbf{K}}_{\text{inv}}^{-1} \tilde{\mathbf{\Xi}}_2$, while the other component arises from the linear circuit to the left of the link and is given by $\tilde{\mathbf{\Xi}}_1^\top \tilde{\mathbf{M}}_{\text{lin}}^{-1} \tilde{\mathbf{\Xi}}_1$. Thus, using the relation between $\tilde{\mathbf{\Xi}}_1$ and $\mathbf{\Upsilon}$, this matrix can be expressed as

$$\tilde{\mathbf{Z}} = \left(\hat{\mathbf{\Gamma}}_1 \tilde{\mathbf{\Lambda}}_{\text{lin}}^{-1} \mathbf{\Gamma}_1 + \tilde{\mathbf{\Xi}}_2^\top \tilde{\mathbf{K}}_{\text{inv}}^{-1} \tilde{\mathbf{\Xi}}_2 \right) \quad (7.12)$$

$\tilde{\mathbf{Z}}$ may be viewed as the Thévenin equivalent impedance matrix of the two subcircuits that relates not only the terminal currents to the voltages but also their derivatives scaled by powers of h .

3. The impedance matrix obtained above can be used to compute the actual current, and its high-order h -scaled derivatives, flowing in the link using the voltages (and its h -scaled derivatives) obtained from the first step. The voltages at the node p on the inverter side of the link are given by $\tilde{\mathbf{\Xi}}_2^\top \hat{\mathbf{\xi}}_{\text{inv}n+1}^{(r+1)}$, while those obtained at the node p' on the left side of the link are given by $\tilde{\mathbf{\Xi}}_1^\top \hat{\mathbf{\xi}}_{\text{lin}n+1}^{(r+1)} = \hat{\mathbf{\Gamma}}_1 \hat{\boldsymbol{\rho}}_{n+1}^{(r+1)}$. Thus the update to the link currents is obtained from

$$\Delta \tilde{\boldsymbol{\gamma}} = \tilde{\mathbf{Z}}^{-1} \left(\hat{\mathbf{\Gamma}}_1 \hat{\boldsymbol{\rho}}_{n+1}^{(r+1)} + \tilde{\mathbf{\Xi}}_2^\top \hat{\mathbf{\xi}}_{\text{inv}n+1}^{(r+1)} \right) \quad (7.13)$$

4. Finally the update to the link current in (8.24) are used to adjust the solution obtained in the first step and compute the desired solution

of (7.9), as follows

$$\tilde{\mathbf{A}}_{\text{inv}} \Delta \boldsymbol{\rho}_{n+1}^{(r+1)} = \hat{\boldsymbol{\Phi}}_1^{(r)} - \boldsymbol{\Gamma}_1 \Delta \tilde{\boldsymbol{\gamma}} \quad (7.14)$$

$$\tilde{\mathbf{K}}_{\text{inv}} \Delta \boldsymbol{\xi}_{\text{inv}n+1}^{(r+1)} = \boldsymbol{\Phi}_2^{(r)} - \tilde{\boldsymbol{\Xi}}_2 \Delta \tilde{\boldsymbol{\gamma}} \quad (7.15)$$

The following lemma establishes the relation between the solution obtained from the above four steps and the solution of the system (7.9)

Lemma 4. *The solution obtained from the above steps is identical to the solution of the system (7.9).*

Proof. Define, the matrices $\mathbf{A}_{11}$, $\mathbf{A}_{12}$, $\mathbf{A}_{21}$ and $\mathbf{B}$ as follows,

$$\mathbf{A}_{11} = \begin{bmatrix} \tilde{\mathbf{A}}_{\text{lin}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{K}}_{\text{inv}} \end{bmatrix} \quad (7.16)$$

$$\mathbf{A}_{12} = \begin{bmatrix} \boldsymbol{\Gamma}_1^\top & \tilde{\boldsymbol{\Xi}}_2^\top \end{bmatrix}^\top \quad (7.17)$$

$$\mathbf{A}_{21} = \begin{bmatrix} \hat{\boldsymbol{\Gamma}}_1 & \tilde{\boldsymbol{\Xi}}_2 \end{bmatrix} \quad (7.18)$$

$$\mathbf{B} = \begin{bmatrix} \hat{\boldsymbol{\Phi}}_1^{(r)\top} & \boldsymbol{\Phi}_2^{(r)\top} \end{bmatrix}^\top \quad (7.19)$$

It can be seen that $\tilde{\mathbf{Z}} = \mathbf{A}_{21} \mathbf{A}_{11}^{-1} \mathbf{A}_{12}$. The solution obtained from the above four steps can be reconstructed by the following matrix-vector product,

$$\begin{bmatrix} \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1} \mathbf{A}_{12} \tilde{\mathbf{Z}}^{-1} \mathbf{A}_{21} \mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1} \mathbf{A}_{12} \tilde{\mathbf{Z}}^{-1} \\ -\tilde{\mathbf{Z}}^{-1} \mathbf{A}_{21} \mathbf{A}_{11}^{-1} & \tilde{\mathbf{Z}}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix}$$

It can be shown that the inverse of the above matrix is given by

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & 0 \end{bmatrix} \quad (7.20)$$

which is equal to the matrix on the right-side of (7.9), thereby completing the proof.

□

7.2 Node Tearing for General Nonlinear Circuits

The above steps can be extended to a more general setting. For example, assume that the circuit can be partitioned to p_{lin} linear subcircuits and p_{nonlin} nonlinear subcircuits. Further, assume that node tearing has been applied to separate those blocks in the circuit, thereby resulting in H short-circuit links. In this situation, the binary selector matrices Ξ_j will have H columns having at most ± 1 entries in each column, and zero otherwise, with $j = 1, \dots, p_{\text{lin}} + p_{\text{nonlin}}$. In general, the m^{th} column of Ξ_j will have a ± 1 entry only if the m^{th} link is connected to the j^{th} partition.

The above solution stages will have to be modified as follows.

1. Solve for the voltage and currents (and their h -scaled derivatives) in each subcircuit (assuming open-circuit links) using

$$\tilde{\mathbf{A}}_{\text{lin}j} \hat{\boldsymbol{\rho}}_{n+1,j}^{(r+1)} = \hat{\boldsymbol{\Phi}}_j^{(r)}, \quad j = 1, \dots, p_{\text{lin}} \quad (7.21)$$

$$\tilde{\mathbf{K}}_{\text{nonlin}j} \hat{\boldsymbol{\xi}}_{n+1,j}^{(r+1)} = \boldsymbol{\Phi}_j^{(r)}, \quad j = 1, \dots, p_{\text{nonlin}} \quad (7.22)$$

Note here that although any of the subcircuits may not have external independent sources attached to it, the entries in the vectors $\hat{\Phi}_j^{(r)}$ act as independent sources and maybe viewed as the stimuli that generates responses in each subcircuit under the condition of open-circuited terminals $\Delta\tilde{\gamma} = \mathbf{0}$.

2. Compute the impedance matrix at the ports created by the links:

$$\tilde{\mathbf{Z}} = - \left(\sum_{j=1}^{p_{\text{lin}}} \hat{\Gamma}_j (\tilde{\Lambda}_{\text{lin}j})^{-1} \Gamma_j + \sum_{j=1}^{p_{\text{nonlin}}} \tilde{\Xi}_j^{\top} (\tilde{\mathbf{K}}_{\text{nonlin}j})^{-1} \tilde{\Xi}_j \right) \quad (7.23)$$

where $\Gamma_j = \Upsilon^{-1} \tilde{\Xi}_j$, $\hat{\Gamma}_j = \tilde{\Xi}_j^{\top} \Upsilon$, and $\tilde{\Xi}_j$ is a matrix with kH columns obtained from $\Xi_j \otimes \mathbf{I}_k$. Using the properties of the Kronecker matrix operator, $\hat{\Gamma}_j$ can be simplified, as shown next

$$\Gamma_j = \Upsilon^{-1} \tilde{\Xi}_j = (\mathbf{I}_{N_j} \otimes \mathbf{V}^{-1}) (\Xi_j \otimes \mathbf{I}_k) = \Xi_j \otimes \mathbf{V}^{-1}$$

where N_j is the size of the j^{th} linear subcircuit. By a similar argument, we can show that $\hat{\Gamma}_j = \Xi_j^{\top} \otimes \mathbf{V}$.

3. The correction needed for link currents $\Delta\tilde{\gamma}$ will follow from

$$\Delta\tilde{\gamma} = -\tilde{\mathbf{Z}}^{-1} \left(\sum_{j=1}^{p_{\text{lin}}} \hat{\Gamma}_j \hat{\rho}_{n+1,j}^{(r+1)} + \sum_{j=1}^{p_{\text{nonlin}}} \tilde{\Xi}_j^{\top} \hat{\xi}_{n+1,j}^{(r+1)} \right) \quad (7.24)$$

4. Finally, we obtain the correction needed for the voltages and currents

of the subcircuits

$$\tilde{\mathbf{A}}_{\text{lin}j} \Delta \boldsymbol{\rho}_{n+1,j}^{(r+1)} = \hat{\boldsymbol{\Phi}}_j^{(r)} - \boldsymbol{\Gamma}_j \Delta \tilde{\boldsymbol{\gamma}}, \quad j = 1, \dots, p_{\text{lin}} \quad (7.25)$$

$$\tilde{\mathbf{K}}_{\text{nonlin}j} \Delta \boldsymbol{\xi}_{n+1,j}^{(r+1)} = \boldsymbol{\Phi}_j^{(r)} - \tilde{\boldsymbol{\Xi}}_j \Delta \tilde{\boldsymbol{\gamma}}, \quad j = 1, \dots, p_{\text{nonlin}} \quad (7.26)$$

7.3 Numerical Examples

This section presents the results of the numerical simulations using the ObF for various practical examples. Throughout the examples, the performance of the presented ObF-based method is compared to the performance of the trapezoidal rule method (TR), which is of order 2 and is used in commercial simulators such as HSPICE [61]. Our choice for this method in carrying out the comparison is also motivated by the fact that it represents the highest order A -stable method in the LMS suite. Therefore, it is not affected by the inaccuracies resulting from the lack of stability, and hence, can be used for fair comparison, since the ObF-based method is also A -stable. It should be noted that our implementation of the TR is based on using the developed technique for $k = m = 1$, which can be shown to be equivalent to the TR [50].

7.3.1 Example 1

In this example, we consider a coupled interconnects circuit shown in Figure 7.2. The length of the lines is 6cm. The p.u.l parameters of the coupled lines are extracted using field solvers in HSPICE [61]. The lines are then modeled using 50 lumped

Figure 7.2: Interconnects Circuit (Example 1)

sections [47]. The lines are excited with a voltage source attached to the near-end of each fourth line, with 0.2ns fall/rise time, 5ns pulse width and 2V in magnitude. A CMOS inverter is attached at the far-end of each fourth line (*with a total of 6 inverters*). The Node Tearing algorithm is applied to partition the circuit such that the first subcircuit contains the linear RLC components representing the transmission lines, while the other subcircuits contain the nonlinear terminations at the far-ends of the lines. In this example, $N = 3776$ and $p_{\text{nonlin}} = 6$.

Table 7.1 shows the CPU-time comparison for the transient simulation in the case of different integration orders. The second column shows the number of time points required to simulate the circuit, while the third column shows the maximum error for each order at the far-end node of the victim line 2, estimated using the techniques in [14]. The CPU-time required for the transient simulation using TR method for this example was 328.95 sec and the number of time points required is 2405. Note here that the decrease in the number of time points with the increase in the order

offsets the additional computational cost of the high-order and results in an overall saving in computation time.

Table 7.1: CPU-time comparison for Example 1

(m/k)	# Time Pts	ObF-based method (sec)	Speedup w.r.t TR
(2/2)	325	43.02	7.65
(3/3)	165	28.62	11.49
(4/4)	122	22.11	14.88

Figure 7.3 shows a sample time-domain waveform at the far-end of the victim line (second line) using different orders, Figure 7.4 shows a sample time-domain waveform at the input node of the inverter attached to the far-end of the first line and Figure 7.5 shows the time-domain waveform at the output node of the inverter.

Figure 7.3: Comparison of transient waveforms at far-end of victim Line 2 (Example 1)

Figure 7.4: Comparison of transient waveforms at input node of the inverter attached to Line 1 (Example 1)

Figure 7.5: Comparison of transient waveforms at output node of the inverter attached to Line 1 (Example 1)

7.3.2 Example 2

A 32 coupled interconnects circuit, of length 5 cm, shown in Figure 7.6, is considered.

The active lines are excited with voltage sources with 0.2ns fall/rise time, 5ns pulse

Figure 7.6: Interconnects Circuit (Example 2)

width and 1.2V in magnitude. The lines are terminated with the CMOS-inverter used in Example 1, with a total number of 11 inverters. The lines are modeled using the lumped segmentation approach (*40-sections*). In this example, the size of the MNA equations is $N = 4094$ and the number of nonlinear partitions is $p_{\text{nonlin}} = 8$.

Table 7.2 shows the CPU-time comparison for the transient simulation in the case of different integration orders. The second column shows the number of time points required to simulate the circuit, while the third column shows the maximum error obtained for each order at the far-end node of the victim line 14. Again, all tested orders result in the same accuracy, while the number of points decreases for high orders. The CPU-time required for the transient simulation using TR method for this example was 743.59 sec and the number of time points required is 2350. Figure 7.7 shows a sample time-domain waveform at the far-end of line 8 using different orders.

Table 7.2: CPU-time comparison for Example 2

(m/k)	# Time Pts	ObF-based method (sec)	Speedup w.r.t TR
(2/2)	315	81.91 sec	9.08
(3/3)	174	53.22 sec	13.97
(4/4)	134	42.12 sec	17.65

Figure 7.7: Comparison of transient waveforms at far-end of line 8 (Example 2)

7.3.3 Example 3

In this example, an interconnect structure consisting of two 16-port interconnects structure, as shown in Figure 7.9, is considered. Each 16-port module has the same physical structure as shown in Figure 7.8. The S-parameters are first calculated using a 2-D field solver in Hspice [61]. The data is then fitted using a vector-fitting algorithm [58], and the results are synthesized using linear elements. The input ports

Figure 7.8: Eight coupled transmission lines physical structure (Example 3)

Figure 7.9: Circuit for Example 3

are excited with voltage sources of 0.1ns rise/fall time, 10ns pulse width and 3V of magnitude. The output ports are terminated with a CMOS inverter. In this example, $N = 25708$ and $p_{\text{nonlin}} = 4$.

Table 7.3 compares the CPU-time required for the transient simulation for orders 4, 6 and 8, with column two showing the number of time points required for the transient simulation and column three showing the maximum error obtained at a sample node p10 of subnetwork (B). As seen, the magnitude of the error of different integration orders are almost the same, however, the number of time points required decreases as the order increases. The CPU-time required for the TR method (order

2) for this example is 141.11 sec. Column 3 shows the speedup of the ObF-based method compared to the TR method. Figure 7.10 shows a sample of the waveforms at port 10 of subnetwork (B) for the case of order 6 using the ObF-based method.

Table 7.3: CPU-time comparison for Example 3

(m/k)	#	Time Pts	Max. Err.	ObF-based method (sec)	Speedup w.r.t TR
(2/2)	310		0.0010	46.54	3.03
(3/3)	190		0.0009	28.12	5.02
(4/4)	130		0.0011	20.10	7.02

Figure 7.10: Comparison of transient waveforms at p10 of subnetwork B (Example 3)

7.4 Summary

In this chapter, a fast transient simulation is presented that uses the Obreshkov-based high order methods, for simulation of nonlinear circuits dominated with large linear blocks. These kind of circuits typically arise from high-speed interconnects and packages. The core of the presented technique is to partition the circuit into linear and non-linear subcircuits using Node Tearing methods in order to exploit the special matrix structure that arises from the linear subcircuits. It was shown through numerical simulations that the presented ObF-based method provides significant speedup compared to conventional TR methods.

Chapter 8

Parallel Simulation of Large Linear Circuits with NonLinear Terminations

In this chapter, details of the ObF-based parallel simulation framework for linear circuits is given. Recently, using modern multi-core architectures to perform parallel time-domain transient simulation that scales efficiently with the number of available cores has been presented recently in [73, 74]. However, the approaches presented in those works usually reach their peak efficiency when the circuit matrices are those that arise from using classical low-order methods in solving the circuit MNA. On the other hand, the matrices that arise from the high-order ObF methods are different in structure and larger in size. Therefore, combining the advantages of the scalable parallel multicore with the ObF high-order methods is not straightforward [32, 33].

This difficulty is addressed in the presented approach using a problem reformulation that produces several matrices whose structures are identical to the matrix produced by the low-order methods. This fact is then used to allow a straightforward application of the multicore low-order paradigms to process the resulting matrices in

the same scalable manner.

For the rest of this chapter, the main development will be focused on the problem reformulation. The main objective is to demonstrate how this problem reformulation also enables a parallel multicore architecture to make the per-time-step computational cost of the high-order methods comparable to the cost of the low-order ones, while maintaining much smaller number of time points of the high-order methods. More specifically, it will be shown that the parallel algorithms decrease the computational complexity per-time-step of the high-order from $O(N^\sigma k)$, arising from the Obreshkov formula (3.32), whose order $k + m$, to only $O(N^\sigma)$, which is the typical cost of the low-order method used SPICE-based simulators while maintaining a larger time step than the low-order methods.

For this purpose, the set of equations in (6.10) is transformed into an alternative problem wherein the matrix resulting from the high-order integration is decoupled into independent matrices of smaller size; thus enabling the solution of (6.10) via factorizing the independent matrices in parallel. This transformation is accomplished by first permuting the unknown vector $\boldsymbol{\xi}_{n+1}$ into $\boldsymbol{\xi}_{p,n+1}$ so that the components corresponding to the i^{th} -order derivative of all the N components are all grouped together, as

$$\boldsymbol{\xi}_{p,n+1} = \mathbf{P}\boldsymbol{\xi}_{n+1} = [\mathbf{x}_{n+1}^{(0)T} \quad h\mathbf{x}_{n+1}^{(1)T} \quad \dots \quad h^{k-1}\mathbf{x}_{n+1}^{(k-1)T}] \quad (8.1)$$

where $\mathbf{x}_{n+1}^{(i)} \in \mathbb{R}^N$ contains the i^{th} derivatives of the unknown voltages/currents of the circuit and $\mathbf{P}$ is a permutation matrix whose entries $\in [0, 1]$.

Next, applying a change of variables similar to (6.7) and (6.10) results in

$$\tilde{\Lambda}(\boldsymbol{\rho}_{n+1}) = \Upsilon^{-1}\tilde{\mathbf{b}}_{n+1} \quad (8.2)$$

$$\mathbf{P}\tilde{\Lambda}\mathbf{P}^\top(\mathbf{P}\boldsymbol{\rho}_{n+1}) = \mathbf{P}\Upsilon^{-1}\tilde{\mathbf{b}}_{n+1} \quad (8.3)$$

$$\tilde{\Lambda}_p\boldsymbol{\rho}_{p,n+1} = \mathbf{P}\Upsilon^{-1}\tilde{\mathbf{b}}_{n+1} \quad (8.4)$$

where $\boldsymbol{\rho}_{p,n+1} = \mathbf{P}\boldsymbol{\rho}_{n+1}$ and $\tilde{\Lambda}_p = \mathbf{P}\tilde{\Lambda}\mathbf{P}^\top$.

Fig. 8.1 illustrates the effect of applying the transformation using $\mathbf{P}$ on the structure of the matrix $\tilde{\Lambda}$. Here, Fig. 8.1(a) shows a random matrix (whose entries are represented by the black dots) and Fig. 8.1(b) shows the matrix with diagonal blocks that is obtained as a result of applying the transformation.

The system of equations in (8.4) is solved using LU factorization followed by Forward/Backward (F/B) substitution. However, as Fig. 8.1 clearly shows, the LU factorization and the F/B substitution can be done in parallel by exploiting some important characteristics of the matrix $\tilde{\Lambda}_p$:

- The matrix $\tilde{\Lambda}_p$ is a block diagonal matrix with k blocks. Each block is a $N \times N$ complex matrix as depicted in Fig. 8.1. Therefore, each block can be factorized independently on a separate processor.
- The size and sparsity pattern of each block is identical to the size and sparsity pattern of the matrix that arises in the conventional low-order methods, used in current SPICE-based simulators, such as

Figure 8.1: The structure of the matrix arising from applying the proposed method compared to the direct application of high-order integration method.

Backward-Euler method. Thus, only one sparse ordering is required for all the $N \times N$ blocks.

- The above feature also implies that the existing code used to parallelize the simulation of low-order methods can be linked to process each block and produce further performance gain that scales with the available cores on the system.
- Each block matrix is constructed using one of the eigen-values of the matrix $\mathbf{T}$. For example, a block in $\tilde{\mathbf{A}}_p$ can be written as $\mathbf{G} + \frac{\lambda_p}{h}\mathbf{C}$ using one of the eigenvalues of $\mathbf{T}$ (λ_p). Given the fact that those eigenvalues always appear in complex conjugate pairs, the diagonal blocks of $\tilde{\mathbf{A}}_p$ will also appear in complex conjugate pairs. This can be exploited to reduce the computational effort, by factorizing only

one of the conjugate pair matrices, and using the corresponding solution to construct, via conjugation, to obtain the solution due to its conjugate. This feature allows saving one processor per complex-pair.

To illustrate this idea further, Table 8.1 lists the possible various orders of the Obreshkov method indicating the type (real or complex) of the blocks on the diagonal. This table shows only the combinations of k and m that guarantee the stability of the method, i.e. $k - 2 \leq m \leq k$, whilst the order of the method is $k + m$ [14]. It should be noted that the number of diagonal blocks is equal to k , which is the number of high-order derivatives at the current time step. However, the types of blocks (real or complex-conjugate) depend on the actual value of k .

Table 8.1 also shows the number of processors that can be assigned to factorize those blocks, with the objective of making the per-time-step cost of the ObF comparable to that of low-order conventional methods. As shown, Table 8.1 lists the number of processors required for this purpose for possible values of k . For example, the case of $k = 4, m = 3$ (order 7) results in 4 blocks (2 complex-conjugate pairs) that can be assigned to two independent processors. Similarly, $k = 5, m = 4$ (order 9) results in 5 blocks (1 real and 2-complex-conjugate pair) on the diagonal.

It should be noted that a processor assigned to factorize a complex block will have to use the complex version of the matrix library, and may therefore consume slightly more time than to factorize a real block.

It is to be highlighted that the permutation in (8.1) is done by changing the rows and not by multiplying by the matrix $\mathbf{P}$. Thus, this step does not affect the over all

Table 8.1: Required number of processors versus the order of the Obreshkov formula.

Total Order*		Number of diagonal blocks		Required number of processors
k	m	Real	Complex	
2	0	0	2	1
	1	0	2	1
	2	0	2	1
3	1	1	2	2
	2	1	2	2
	3	1	2	2
4	2	0	4	2
	3	0	4	2
	4	0	4	2
5	3	1	4	3
	4	1	4	3
	5	1	4	3

* The order of the ObF is $k + m$.

CPU time.

8.1 Parallel ObF-based High-Order Simulation for Nonlinear Circuits

In this section, details of the parallel simulation framework for the case of large linear networks with nonlinear devices loads. The main objective in this section is to show how the high-order ObF-based method can be used to simulate those types of circuits, with per-time-step computational complexity of $O(N^\sigma)$, which is roughly the same computational complexity of low-order methods used in current SPICE-based

simulators. Similar to the presentation in the previous section, this computational complexity does not depend on the value of k used to raise the order of integration, and therefore allows the benefits of using high-order but without the overhead cost.

8.1.1 Formulation of the ObF-based methods using node tearing algorithm

Consider the MNA equations for nonlinear circuits

$$\mathbf{G}\mathbf{x}(t) + \mathbf{C}\frac{d\mathbf{x}}{dt} + \mathbf{f}(\mathbf{x}(t)) = \mathbf{b}(t) \quad (8.5)$$

The goal of this section is to summarize the net effect of carrying out the following two processes:

- Applying the concept of node tearing on the circuit, and
- Discretizing the MNA representing the circuit resulting from the node tearing process using the Obreshkov formula on the system (8.5).

First, the process of node tearing is applied to the circuit with the objective of isolating the nonlinear loads into independent partitions, while grouping the linear modules in separate partitions. Node tearing typically proceeds by inserting short-circuit links between the different partitions, and adding the currents in those links as additional variables in the MNA formulations. Without loss of generality, and to simplify the equations, we assume that the circuit is partitioned, using node tearing, into one large linear subcircuit and s nonlinear subcircuits representing the terminal

devices, and let the required number of short-circuit links be H . Next, the Obreshkov formula is applied to the resulting MNA using the same approaches presented in [14].

The net effect of performing the above two processes on (8.5) is to produce the following system of nonlinear algebraic equations,

$$\begin{bmatrix} \boxed{\begin{matrix} \tilde{\Lambda} \\ \text{large linear block} \end{matrix}} & \mathbf{0} & \dots & \dots & \tilde{\Xi}_{\text{lin}} \\ \mathbf{0} & \tilde{M}_1 & \dots & \dots & \tilde{\Xi}_1 \\ \vdots & \mathbf{0} & \dots & \dots & \vdots \\ \mathbf{0} & \dots 0 & \dots & \tilde{M}_s & \tilde{\Xi}_s \\ \tilde{\Omega} & \tilde{\Xi}_1^\top & \dots & \tilde{\Xi}_s^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\rho}_{n+1} \\ \boldsymbol{\xi}_{1,n+1} \\ \vdots \\ \boldsymbol{\xi}_{s,n+1} \\ \boldsymbol{\gamma}_{n+1} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \tilde{\boldsymbol{\eta}}_{1,n+1} \\ \vdots \\ \tilde{\boldsymbol{\eta}}_{s,n+1} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \Upsilon^{-1} \tilde{\mathbf{b}}_{n+1} \\ \tilde{\mathbf{b}}_{1,n+1} \\ \vdots \\ \tilde{\mathbf{b}}_{s,n+1} \\ \mathbf{0} \end{bmatrix} \quad (8.6)$$

The system of equations (8.6), being nonlinear, requires an iterative scheme such as the Newton-Raphson (NR) [75] method to find its solution. The r^{th} NR iteration is typically written as

$$\boldsymbol{\zeta}_{n+1}^{(r+1)} = \boldsymbol{\zeta}_{n+1}^{(r)} + \Delta \boldsymbol{\zeta}_{n+1}^{(r+1)} \quad (8.7)$$

where $\Delta \boldsymbol{\zeta}_{n+1}^{(r+1)}$ is the update of the r^{th} iteration, which is obtained via solving the following system of equations

$$\tilde{\mathbf{J}} \Delta \boldsymbol{\zeta}_{n+1}^{(r+1)} = \boldsymbol{\Phi}^{(r)} \quad (8.8)$$

In (8.8), $\boldsymbol{\Phi}^{(r)}$ is residual error, or the difference between the right- and left-side of (8.6) arising from the r^{th} iteration trial vector $\boldsymbol{\zeta}_{n+1}^{(r+1)}$, and $\tilde{\mathbf{J}}$ is the Jacobian matrix

(or the matrix of partial derivatives) given by

$$\tilde{\mathbf{J}} = \begin{bmatrix} \boxed{\begin{matrix} \tilde{\mathbf{\Lambda}} \\ \text{large linear block} \end{matrix}} & \mathbf{0} & \dots & \dots & \tilde{\mathbf{\Xi}}_{\text{lin}} \\ \mathbf{0} & \tilde{\mathbf{K}}_1 & \dots & \dots & \mathbf{\Xi}_1 \\ \vdots & \mathbf{0} & \dots & \dots & \vdots \\ \mathbf{0} & \dots 0 & \dots & \tilde{\mathbf{K}}_s & \mathbf{\Xi}_s \\ \tilde{\mathbf{\Omega}} & \mathbf{\Xi}_1^\top & \dots & \mathbf{\Xi}_s^\top & \mathbf{0} \end{bmatrix} \quad (8.9)$$

with $\tilde{\mathbf{K}}_j = \tilde{\mathbf{M}}_j + \tilde{\mathbf{J}}_j$ and $\tilde{\mathbf{J}}_j$ is the Jacobian matrix of the j^{th} nonlinear subcircuit resulting from applying ObF to the MNA equations in (8.5).

The description of the Jacobian matrix given in (8.9) reveals the basic structure of the matrix that needs to be factorized at each NR iteration. Indeed, for the types of circuits addressed in this chapter (large linear networks with small nonlinear devices), the Jacobian matrix produced by the node tearing algorithm will have a large block corresponding to the linear network, represented by $\tilde{\mathbf{\Lambda}}$, and several much smaller blocks $\tilde{\mathbf{K}}_i, i = 1, \dots, s$ corresponding to the nonlinear device loads, in addition to the border matrices that arise in the normal course of applying the node tearing.

The main purpose in the remainder of this section is to present a parallel algorithm to solve the system of equations in (8.8). It must be noted that the regular application of the node tearing method would proceed to parallelize this solution by simply factorizing the Jacobian matrix corresponding to each subcircuit on a separate processor. However, such an approach does not provide good speedup. This is mainly because the circuit is partitioned such that the large linear subcircuit appears in a

separate partition. Therefore, when executed on a parallel environment, the CPUs responsible for factorizing the Jacobian matrix pertaining to the linear subnetworks may take more time to finish while the CPUs responsible for the nonlinear devices remain idle.

The remainder of this section will therefore follow a different approach to address this issue. It will first start by outlining the computational steps used to solve (8.8) and then followed by the parallel implementation strategy for each step.

8.1.2 Solving the partitioned system

The basic steps that need to be followed to solve the system in (8.8), without having to factorize the entire Jacobian matrix $\tilde{\mathbf{J}}$ in (8.9) can be summarized by the following steps.

Step 1: Solving each subcircuit independently

In the first step, solve for the voltages and currents (and their h -scaled derivatives) in each subcircuit (assuming zero current flowing through the inserted links, i.e. $\Delta\boldsymbol{\gamma}_{n+1} = \mathbf{0}$).

$$\tilde{\mathbf{\Lambda}}\boldsymbol{\kappa} = \boldsymbol{\Phi}^{(r)}, \quad (8.10)$$

$$\tilde{\mathbf{K}}_j\boldsymbol{\kappa}_j = \boldsymbol{\Phi}_j^{(r)}, \quad j = 1, \dots, s \quad (8.11)$$

where $\boldsymbol{\kappa}_p$ and $\boldsymbol{\kappa}_j$ represent the voltages/currents of the linear partition and the j^{th} nonlinear subcircuit, respectively..

As is evident, it is required to solve each subcircuit to find $\boldsymbol{\kappa}$ and $\boldsymbol{\kappa}_j$, $j = 1, \dots, s$, in (8.10) and (8.11). This step involves LU factorization of the diagonal blocks ($\tilde{\mathbf{\Lambda}}$, and $\tilde{\mathbf{K}}_i, i = 1, \dots, s$), followed by Forward/Backward substitution.

Similar to Chapter 6, the matrix $\tilde{\mathbf{\Lambda}}$ is first permuted to a block diagonal structure by applying a permutation using the matrix $\mathbf{P}$ as follows

$$\tilde{\mathbf{\Lambda}}_p \boldsymbol{\kappa}_p = \mathbf{P} \boldsymbol{\Phi}^{(r)} \quad (8.12)$$

The vector $\boldsymbol{\kappa}_p$ is found by first LU factorizing the diagonal blocks of $\tilde{\mathbf{\Lambda}}_p$ in parallel assigning a processor to each diagonal block. Next, a Forward/Backward substitution is performed in parallel by also taking advantage of its structure. Once any processor is available, a nonlinear subcircuit can be assigned to it to solve for its $\boldsymbol{\kappa}_j$.

Step 2: Computing the impedance matrix

This step requires evaluating the impedance matrix ($\tilde{\mathbf{Z}}$) at the ports created by the inserted short-circuit links, which is given by

$$\tilde{\mathbf{Z}} = - \left(\tilde{\mathbf{\Omega}} \tilde{\mathbf{\Lambda}}^{-1} \tilde{\mathbf{\Xi}}_{\text{lin}} + \sum_{j=1}^s \tilde{\mathbf{\Xi}}_j^{\top} \tilde{\mathbf{K}}_j^{-1} \tilde{\mathbf{\Xi}}_j \right) \quad (8.13)$$

The matrix $\tilde{\mathbf{Z}}$ is an $kH \times kH$ matrix and is solved in two steps. The $\tilde{\mathbf{Z}}$ matrix is computed by first applying a permutation to $\tilde{\mathbf{\Lambda}}$, similar to Step 1, then solving the

systems,

$$\tilde{\mathbf{\Lambda}}_p \boldsymbol{\psi}_p = \mathbf{P} \tilde{\boldsymbol{\Xi}}_{\text{lin}}, \quad (8.14)$$

$$\tilde{\mathbf{K}}_j \boldsymbol{\psi}_j = \tilde{\boldsymbol{\Xi}}_j, \quad j = 1, \dots, s \quad (8.15)$$

The matrix $\tilde{\mathbf{Z}}$ is found as

$$\tilde{\mathbf{Z}} = - \left(\tilde{\boldsymbol{\Omega}} \boldsymbol{\psi}_p + \sum_{j=1}^s \tilde{\boldsymbol{\Xi}}_j^\top \boldsymbol{\psi}_j \right) \quad (8.16)$$

It is apparent that solving (8.14) can be done in parallel by taking advantage of the block diagonal structure of $\tilde{\mathbf{\Lambda}}_p$, similar to Step 1. Also note that, the L and U factors obtained in Step 1 can be reused to directly run the Forward/Backward substitution to obtain $\boldsymbol{\psi}_p$.

However, this step could consume a significant portion of the overall computation time, as well as affecting the CPU-scalability, due to the fact that the number of Forward/Backward substitution required in (8.14) is kH . Fortunately, we can identify several structural properties for $\tilde{\mathbf{\Lambda}}_p$ and $\tilde{\boldsymbol{\Xi}}_p$, which can be exploited for an efficient parallel implementation, as follows.

Given that $\tilde{\mathbf{\Lambda}}_p$ is a block diagonal matrix, we can write the equation of each block separately as

$$\tilde{\mathbf{\Lambda}}_{p,i} \boldsymbol{\psi}_{p,i} = \tilde{\boldsymbol{\Xi}}_{\text{lin},i}, \quad i = 1, \dots, k \quad (8.17)$$

where $\tilde{\mathbf{\Lambda}}_{p,i} \in \mathbb{R}^{N \times N}$ is the i^{th} block of the matrix $\tilde{\mathbf{\Lambda}}_p$, $\boldsymbol{\psi}_{p,i} \in \mathbb{R}^{N \times kH}$ and $\tilde{\boldsymbol{\Xi}}_{\text{lin},i} \in \mathbb{R}^{N \times kH}$

are the portions of $\boldsymbol{\psi}_p$ and $P\tilde{\boldsymbol{\Xi}}_{\text{lin},i}$, respectively, that correspond to the i^{th} block.

Next, $\boldsymbol{\psi}_{p,i}$ and $\tilde{\boldsymbol{\Xi}}_{\text{lin},i}$ are partitioned into H rectangular matrices each of k columns as follows

$$\boldsymbol{\psi}_{p,i} = [\hat{\boldsymbol{\psi}}_1 \quad \dots \quad \hat{\boldsymbol{\psi}}_H] \quad (8.18)$$

$$\tilde{\boldsymbol{\Xi}}_{\text{lin},i} = [\hat{\boldsymbol{\Xi}}_1 \quad \dots \quad \hat{\boldsymbol{\Xi}}_H] \quad (8.19)$$

Next, we define the following Lemma, which is used in developing the parallelization scheme.

Lemma 5. *Define the set of scalars $R_{p,q}$, by $R_{p,q} = V_{p,q}/V_{p,1}$, $p = 1, \dots, k$, where $V_{p,q}$ is the element at (p,q) entry of the eigen-vectors matrix $\mathbf{V}$ used in (6.7), and let the r^{th} column in $\hat{\boldsymbol{\psi}}_j$, $j = 1, \dots, H$, be denoted by $\hat{\boldsymbol{\psi}}_{j,r}$, where $r = 1, \dots, k$, then*

$$\hat{\boldsymbol{\psi}}_{j,r} = R_{j,r} \hat{\boldsymbol{\psi}}_{j,1} \quad (8.20)$$

Proof. To prove the lemma, note that

$$\tilde{\Lambda}_{p,i} \hat{\boldsymbol{\psi}}_{j,r} = \hat{\boldsymbol{\Xi}}_{j,r} = R_{j,r} \hat{\boldsymbol{\Xi}}_{j,1}$$

Multiplying both sides by $(1/R_{j,r})$

$$\tilde{\Lambda}_{p,i} R_{j,r}^{-1} \hat{\boldsymbol{\psi}}_{j,r} = \hat{\boldsymbol{\Xi}}_{j,1} \quad (8.21)$$

Nonetheless, we have by definition

$$\tilde{\Lambda}_{p,i} \hat{\psi}_{j,1} = \hat{\Xi}_{j,1}, \quad (8.22)$$

Therefore, it follows from comparing (8.22) with (8.21), that

$$R_{j,r}^{-1} \hat{\psi}_{j,r} = \hat{\psi}_{j,1}, \quad (8.23)$$

which proves the lemma. $\square$

Notice that the scalars $R_{p,q}$ are a set of predefined scalars that are pre-determined for all circuits prior to simulation. As a consequence of this fact and as a result of the above lemma, one does not need to run k Forward/Backward substitutions to find all k columns of $\hat{\psi}_j$. Instead, those k solution vectors can be easily obtained through performing only one Forward/Backward substitution (using the first vector in $\hat{\Xi}_j$) to solve for the first column in $\hat{\psi}_j$; the remaining $k - 1$ columns are simply obtained through scaling with the scalars $R_{j,r}, r = 2, \dots, k$. It should be noted that this process is amenable to efficient parallel implementation.

Another important characteristic to note here is that for large circuits dominated by large linear subnetworks with relatively few nonlinear devices, the vector $\hat{\Xi}_{j,r}$ has at most one non-zero component in each column. This structure can be exploited in the Forward/Backward substitution to save crucial computational efforts. For example, the Forward substitution can start from the location of the non-zero entry instead from the first entry.

Step 3: Calculating the link currents

In this step, the NR update to the link currents ($\Delta\gamma_{n+1}$) are calculated using the value of the impedance matrix found in the previous step and the voltages and currents obtained from Step 1 as

$$\Delta\tilde{\gamma} = \tilde{\mathbf{Z}}^{-1} \left(\tilde{\mathbf{\Omega}}\boldsymbol{\kappa} + \sum_{j=1}^s \tilde{\mathbf{\Xi}}_j^\top \boldsymbol{\kappa}_j \right) \quad (8.24)$$

As can be seen, this step is a simple summation of matrix-vector products. The products of $\tilde{\mathbf{\Omega}}\boldsymbol{\kappa}$ and $\tilde{\mathbf{\Xi}}_j^\top \boldsymbol{\kappa}_j$ are first calculated in parallel on separate processors and then the summation is found using the master processor. However, it is important to highlight that, in the case of the nonlinear subcircuits, the matrices $\tilde{\mathbf{\Xi}}_j$ have only ± 1 entries. Therefore, the product can be found by only selecting the entries in $\boldsymbol{\kappa}_{j,n+1}$ that correspond to ± 1 in $\tilde{\mathbf{\Xi}}_j$.

Step 4: Finding the voltages/currents of the complete circuit

In the final step, we obtain the NR update for the rest of the variables in (8.6) by solving the following system of equations,

$$\tilde{\mathbf{\Lambda}}\Delta\boldsymbol{\rho}_{n+1}^{(r+1)} = \boldsymbol{\Phi}^{(r)} - \tilde{\mathbf{\Xi}}_{\text{lin}}\Delta\boldsymbol{\gamma}, \quad (8.25)$$

$$\tilde{\mathbf{K}}_j\Delta\boldsymbol{\xi}_{n+1,j}^{(r+1)} = \boldsymbol{\Phi}_j^{(r)} - \tilde{\mathbf{\Xi}}_j\Delta\boldsymbol{\gamma}, \quad j = 1, \dots, s \quad (8.26)$$

Note that this step is similar to Step 1 and hence can be performed in parallel using a similar approach. However, the LU factors of the matrices in this step need not be recalculated but can be reused from Step 1. Thus, only Forward/Backward substitution is required to find $\Delta \boldsymbol{\rho}_{n+1}$ and $\Delta \boldsymbol{\xi}_j$. A flow-chart of the above computational steps is given in Fig. 8.2.

To this extent, it is worth highlighting two important notes:

- Since the circuit is partitioned into one large linear subcircuit and small nonlinear subcircuits, then the LU factors of the linear subcircuit can be found only once for each time point and reused during Newton-Raphson iterations. Once convergence is obtained, the time is advanced and new LU factors are calculated.
- Increasing the nonlinear terminations (and thus increasing the number of links H) has a very small effect on the total CPU-time. This is mainly because increasing the number of links results only in increasing the number of columns in the matrices $\tilde{\mathbf{E}}_{lin}$ and $\tilde{\mathbf{E}}_j$, $j = 1, \dots, S$. However, the columns of these matrices are usually very sparse or even zero if there is no link between two subcircuits. Therefore, the Forward/Backward substitution in step (2a) and the matrix-vector multiplications in steps (2b-4a) can be done very efficiently given the sparse structure of these matrices.

Figure 8.2: Flow Chart of the computational steps for a particular NR iteration using the parallel framework.

8.2 Numerical Examples

This section provides three numerical examples to show the accuracy and CPU-scalability of the presented method when executed on a parallel platform. Simulations were performed on an eight-core machine (Intel(R) Core(TM)2) with a linux operating system. Throughout the examples, we show that the time required to find

the solution at a particular time point can be minimized in spite of increasing the order of the Obreshkov integration method. Also, it is to be emphasized that as the order of the Obreshkov integration method increases, the total number of time points required for the same accuracy decreases significantly (compared to 2nd-order methods), leading to higher speed-up [14, 16].

The results obtained by the parallel algorithm are compared with the conventional SPICE simulation Trapezoidal-Rule (TR) method, since it is the only A -stable low-order method used by current SPICE simulators. In fact, it can be shown that the TR is equivalent to the ObF-method if we set $k = 1$ and $m = 1$ [14] .

8.2.1 Example 1

Figure 8.3: 48-Coupled interconnect circuit used in Example 1

In this example, we consider an interconnects circuit shown in Fig. 8.3. The lines are modeled using lumped RLC sections [47] (40 sections). A voltage source is attached at the input end of each fourth line, with 0.5ns rise/fall times, amplitude of 3V and 5ns pulse width.

Fig. 8.4 and Fig. 8.5 show a set of sample waveforms obtained using the proposed

method as compared to the low-order TR method (order 2, used in conventional simulators). As seen, the results are in a good agreement.

Further, for performance comparison purposes, transient analysis results are compared using three different approaches: the direct-method without node tearing, eigen-decomposition of the blocks (EDB) with node tearing but without parallel processing, and the proposed method (with node tearing and parallel simulation). Fig. 8.6 compares the increase in CPU-time required to solve for one time-step (at $t = 2ns$) using the above three approaches versus the order k of integration used in the Obreshkov-formula. As shown, the CPU-time required for the solution per-time-step, using the direct approach presented in section 3.5, grows with order $O(k^3)$, while it increases linearly in the case of applying EDB presented in Chapter 6 and is almost constant in the case of the proposed parallel method. This observation therefore indicates that the per-time-step computational cost of the high-order method is kept to almost at the similar cost of the low-order methods.

Another worthy observation here is related to the performance of the algorithm at orders 6 (3/3) and 8 (4/4) provided in Fig. 8.6. According to the analysis in Table 8.1, the ideal number of processors to handle these orders is 2, in order to achieve full parallelization in factorizing the diagonal blocks. Therefore, no tangible difference in the performance between using 2 and 3 processors should be expected, as indicated by Fig. 8.6. On the other hand, for orders 10 (5/5) and 12 (6/6) (as Table 8.1 shows) the ideal number of processors to use is 3. Thus, using 3 processors for these orders results in tangible improvements compared with using 2 processors, as reflected in Fig. 8.6.

Figure 8.4: Waveform at the near-end of active line 1 (Example 1)

Figure 8.5: Waveform at the far-end of victim line 2 (Example 1)

Figure 8.6: CPU time comparison for finding the solution at $t = 2\text{ns}$ (Example 1)

8.2.2 Example 2

Figure 8.7: 64-Coupled interconnect circuit with nonlinear loads used in Example 2

In this example, we consider a large coupled interconnects circuit terminated with a CMOS inverter at the far-end of the active lines, as shown in Fig. 8.7. A voltage source, attached at the near-end of each fourth line, with 0.5ns rise/fall times, amplitude of 2V and 5ns pulse width, is used to excite the active lines.

Table 9.4 reports the CPU simulation time using the ObF-based transient simulation, with various orders shown in the first column. The second column shows the number of time points taken during the transient simulations. The third, fourth, and fifth column show, respectively, the CPU time taken by the direct approach in section 3.5, the approach based on node tearing (but without parallel processing), or EDB, presented in Chapter 7, and the approach presented in this Chapter (indicating the number of processors used in parallel to factorize the independent diagonal blocks). The last column shows the speedup ratio between the proposed algorithm compared to the classical SPICE-TR.

The results in the first row correspond to $k = m = 1$, which, as stated earlier, are

Table 8.2: Comparison of the total CPU time for Example 2

Order	CPU (seconds)			Speed up w.r.t TR
	Without Node Tearing	With Node Tearing	With Node Tearing & Parallel	
2 ($k = 1, m = 1$)	228.10	175.46	175.46 (one processor) *	1.3
8 ($k = 4, m = 4$)	64.81	22.82	11.63 (two processors) *	19.6
10 ($k = 5, m = 5$)	58.81	20.91	8.17 (three processors) *	27.9

* The number of processors is based on the criteria described by Table 8.1.

equivalent to the TR used in the classical SPICE simulators, referred to as SPICE-TR. In this particular case, the matrix $\mathbf{T}$ is a scalar, thereby resulting in only one diagonal block that can be factorized using only one processor, as explained in Table 8.1. This row also demonstrates the impact of the node tearing on the TR, indicating about 30% improvement over the classical TR when executed without the node tearing.

Fig. 8.8 and Fig. 8.9 show a set of sample waveforms obtained using the proposed method when run on the parallel platform and are compared with the TR method.

Figure 8.8: Waveform at the far-end of line 9 (Example 2)

Figure 8.9: Waveform at the far-end of line 10 (Example 2)

8.2.3 Example 3

Figure 8.10: Multiple interconnects circuit with nonlinear loads (Example 3)

The circuit for this example consists of three interconnect networks (shown in Fig. 8.10). An inverter is attached to the far-ends of each forth line. The network is excited with a 2V voltage source of 0.5ns rise/fall times and 5ns pulse width.

Following the same structure of Table 9.4 for the previous example, Table 8.3 summarizes the CPU-time results obtained using different integration orders for this example.

Table 8.3: Comparison of the total CPU time for Example 3

Order	CPU (seconds)			Speed up w.r.t TR
	Without Node Tearing	With Node Tearing	With Node Tearing & Parallel	
2 ($k = 1, m = 1$)	431.75	318.39	318.39 (one processor) *	1.4
6 ($k = 3, m = 3$)	90.92	36.27	24.18 (two processors) *	17.8
8 ($k = 4, m = 4$)	65.64	30.974	16.10 (two processors) *	26.8

* The number of processors is based on the criteria described by Table 8.1.

8.3 Summary

This chapter presented an efficient parallel framework to perform Obreshkov formula and high-order integration based transient simulations. The new algorithm is targeted towards circuits with large linear modules and nonlinear terminations. Special structural properties of circuit matrices from such networks are exploited for while developing the parallel algorithm. The parallel framework significantly minimizes the CPU time required (which otherwise tends to increase) while using higher-orders for ObF, by adopting appropriate parallelization schemes.

Chapter 9

High Order and A-Stable Envelope Following Method for Transient Simulations of Oscillatory Circuits

As discussed in section 3.6, the main advantage of the EF method stems from the fact that it enables skipping a large number of cycles of the highly oscillatory waveform. This advantage results in saving significant computational efforts that could have been spent otherwise on solving the system of differential equations at very small step sizes. On the other hand, if the designer is interested in capturing the original waveform with higher details at a particular time interval, then the envelope points generated by the EF method can be utilized to initiate a detailed SPICE analysis with a fine step size to recover the exact waveform in the interval of interest.

Nonetheless, this advantage can not be maintained unless the particular numerical method used to follow the envelope is numerically stable for all physically stable circuits. This is a property known in the mathematical literature as the *A*-stability

[24, 76]. It has been shown in [23] that FE-EF is not A -stable, whereas both BE-EF and TR-EF are A -stable.

Although A -stability is a necessary condition for guaranteeing the success of the EF method, its order [70] is crucial to the efficiency with which the envelope points are generated. To illustrate, the basic concept behind the order of a given method, assume that the *exact* envelope point at $t = nT$ is denoted by $\bar{\mathbf{y}}_n$, and that the past envelope point, i.e., when $t = mT$, is identical to the exact one, that is $\mathbf{y}_m = \bar{\mathbf{x}}_m$. A method is said to be order q if

$$\bar{\mathbf{y}}_n = \mathbf{y}_n + O(H^{q+1}) \quad (9.1)$$

where $O(H^{q+1})$ denoted terms with powers of H that are higher than or equal to $q+1$. Typically a method with high-order enables using a large value for the envelope step size H , thereby reducing number of computed envelope points and the overall simulation time.

It has been shown that the orders of the BE-EF and TR-EF in (3.58) and (3.59), respectively, are 1 and 2, respectively [23]. In this chapter, a new EF method that provides high-order, while keeping the A -stability feature, is presented [34–36]. The new method is based on utilizing the Obreshkov-formula (ObF), which was proposed in [14, 15] for circuit simulation (*discussed in section 3.4 in details*).

Consider the ObF in the following form

$$\sum_{i=0}^k \alpha_i H^i \mathbf{y}_n^{(i)} = \sum_{i=0}^l \beta_i H^i \mathbf{y}_m^{(i)} \quad (9.2)$$

where $\mathbf{y}_n = \mathbf{y}(nT)$ is the envelope waveform, with T being the period of the high frequency waveform, and

$$\alpha_i = (-1)^i \frac{(l+k-i)!k!}{i!(l+k)!(k-i)!} \quad (9.3)$$

$$\beta_i = \frac{(l+k-i)!l!}{i!(l+k)!(l-i)!} \quad (9.4)$$

Adapting the ObF for EF requires a suitable method to approximate the high-order derivatives of the envelope at $t = mT$ and $t = nT$, i.e., $\mathbf{y}_m^{(i)}$ and $\mathbf{y}_n^{(i)}$, respectively. One way to do that is to generalize the finite-difference method (3.56) for approximating high-order derivatives. The generalization of (3.56) to the j^{th} derivative can be written using the difference approximation method as [77]

$$\mathbf{y}_n^{(j)} \approx \frac{1}{T^j} \sum_{i=0}^j \binom{j}{i} (-1)^i \mathbf{y}_{n+(j-i)} \quad (9.5)$$

where $\binom{j}{i}$ are the binomial coefficients.

Using the approximation of (9.5) for the purpose of the Obreshkov-based EF (ObF-EF) results in,

$$\sum_{i=0}^k \sum_{j=i}^k (-1)^{(j-i)} \alpha_j \frac{H^j}{T^j} \binom{j}{j-i} \mathbf{y}_{n+i} = \sum_{i=0}^l \sum_{j=i}^l (-1)^{(j-i)} \beta_j \frac{H^j}{T^j} \binom{j}{j-i} \mathbf{y}_{m+i} \quad (9.6)$$

The resulting system in (9.6) is easily seen as a system of N nonlinear equations that need to be solved for $\mathbf{y}_{n+i}$ using the iterative scheme of the Newton Raphson (NR)

method. The approach used for that purpose, is to relate $\mathbf{y}_{n+i}, i \geq 1$ to $\mathbf{y}_n$ through the state-transition function. Starting from an initial guess for $\tilde{\mathbf{y}}_n^{(0)}$, the NR method updates that guess through

$$\tilde{\mathbf{y}}_n^{(r+1)} = \tilde{\mathbf{y}}_n^{(r)} - \Delta \tilde{\mathbf{y}}_n^{(r+1)} \quad (9.7)$$

$$\Delta \tilde{\mathbf{y}}_n^{(r+1)} = \mathbf{J}^{-1} \Psi(\tilde{\mathbf{y}}_n^{(r)}) \quad (9.8)$$

where r is the iteration number, $\Psi(\tilde{\mathbf{y}}_n^{(r)})$ is the difference between the right and left-hand sides of (9.6) and $\mathbf{J}$ is the Jacobian matrix. Alternatively, for a better convergence, a damped NR method can be used [78].

The Jacobian matrix in (9.8) is defined as

$$\mathbf{J} = \sum_{i=0}^k \sum_{j=i}^k (-1)^{(j-i)} \alpha_j \frac{H^j}{T^j} \binom{j}{j-i} \frac{d\mathbf{y}_{n+i}}{d\mathbf{y}_n} \quad (9.9)$$

where the matrices $d\mathbf{y}_{n+i}/d\mathbf{y}_n \in \mathbb{R}^{N \times N}$ ($i = 1, \dots, k$) are known as the sensitivity matrices, which in this context represent the first-order variation of the solution at $t = (n+i)T$ w.r.t. the solution at $t = nT$.

In general, a sensitivity matrix, e.g. $d\mathbf{y}_{n+1}/d\mathbf{y}_n \in \mathbb{R}^{N \times N}$ at $t = (n+1)T$ is computed using standard techniques proposed in the literature [54], through initiating a classical SPICE-based detailed analysis at the initial point $t = nT$ and terminating it at $t = (n+1)T$. Using the results of this simulation, the sensitivity matrix of the terminal solution w.r.t. the initial condition can be computed. The details of efficiently

computing such a matrix can be found in [23, 79, 80].

It is worth highlighting that the ObF-EF method can handle circuits including distributed elements. The computation of $\mathbf{y}_{m+i}$ as well as the solution of the boundary value problem in (9.6) can be determined using any suitable stable integration method.

9.1 Summary and Discussions of ObF-EF

The above presentation suggests that the implementation of an Obreshkov-based single envelope step can be summarized by the following steps.

1. Assume that $\mathbf{y}_m, \mathbf{y}_{m+1}, \dots, \mathbf{y}_{m+l}$ envelope points have been already computed.
 2. Assign an initial guess for the envelope point at $t = nT$, and denote this guess by $\tilde{\mathbf{y}}_n^{(r)}$, where $r = 0$.
 3. Using classical (non-envelope) SPICE-based analysis perform transient time-domain simulation for the system (3.37), starting at $t = nT$, (assuming that $\mathbf{y}(nT) = \tilde{\mathbf{y}}_n^{(r)}$) and until $t = (n+k)T$.
 4. Assign the values obtained through the analysis in the previous step to the envelope points, i.e., let $\tilde{\mathbf{y}}_{n+i}^{(r)} = \mathbf{y}((n+i)T)$, $i = 1, \dots, k$.
 5. Substitute the values of $\tilde{\mathbf{y}}_{n+i}^{(r)}$ for $\mathbf{y}_{n+i}$ in the left side of (9.6), and mark the resulting difference between the right and left side of (9.6) by $\Psi(\tilde{\mathbf{y}}_n^{(r)})$.
-

6. If $\|\Psi(\mathbf{y}_n^{(r)})\| < \epsilon$, where $\|\cdot\|$ is an appropriate vector norm, then STOP:
the next envelope points (i.e., at $t = (n+i)T$) $\mathbf{y}_{n+i}$ are given by $\tilde{\mathbf{y}}_{n+i}^{(r)}$,
for $i = 0, \dots, k$.
7. Use the points generated in Step (3) to compute the sensitivity matrices $d\tilde{\mathbf{y}}_{n+i}/d\tilde{\mathbf{y}}_n$, $i = 1, \dots, k$ and use those matrices to compute the Jacobian matrix, $\mathbf{J}$ in (9.9).
8. Using the update (9.7), compute $\tilde{\mathbf{y}}_n^{(r+1)}$.
9. Let $r \leftarrow r + 1$, then go to Step (3).

The key advantage that the ObF-EF offers, compared with other EF methods, is the ability of using high-order approximation of the envelope waveform without losing the A -stability property. In addition, an adaptive step size control can be used to control the local truncation error (LTE) below a certain pre-specified tolerance. More details can be found in [14].

Nevertheless, transforming this potential into concrete computational gains requires efficient implementation strategies. The rest of the chapter is dedicated to describing those strategies.

More particularly, Section 9.2 describes an efficient technique to compute the initial guess, $\tilde{\mathbf{y}}_n^{(0)}$, required in Step (2). As will be shown in the numerical simulations of Section 9.6, this technique reduces the overall NR iterations needed to reach convergence. Section 9.3 explores utilizing alternative approaches to the finite difference formula in (9.5) for the approximation of the high-order derivatives

9.2 Improved Initial Guess

It should be obvious from the previous section that the proposed ObF-EF is of the implicit type, which requires the utilization of the iterative NR framework to calculate the next point on the envelope waveform. Furthermore, each iteration requires the computation of the sensitivity matrix over several periods of the fast cycle T . Therefore, to ensure that the proposed scheme is executed efficiently, the number of iterations should be minimized through starting the NR with an initial guess that is sufficiently close to the final solution.

In the classical low-order simulation methods that are used by SPICE, this issue is easily resolved by taking the initial guess as the past point (which is already calculated). The rationale behind this strategy is that low-order methods naturally force the simulator to take a small step size. Hence, the past point is likely to be close enough to the future point on the waveform, thereby enabling fast convergence within few iterations.

However, this rationale can no longer be held for the high-order ObF-EF scheme in the proposed approach, since the main objective here is to take advantage of the large step size afforded by the high-order. With a large envelope step size H , the envelope waveform, more likely than not, will have taken a value that could be significantly different from the past point. Therefore, adopting the strategy of using the past value (in this case $\mathbf{y}_m$) for the initial guess $\tilde{\mathbf{y}}_n^{(0)}$ might result in a number of iterations that is not within the maximum allowable pre-specified number of iterations.

This section proposes a new strategy aiming at providing an initial guess that is suitable for the proposed approach. The proposed scheme is based on the work presented in [81] and can be illustrated as follows.

Consider a set of N waveforms, denoted by $\hat{y}_i(t)$, $i = 1, \dots, N$, described by a hybrid waveform of a k -degree polynomial, where k determines the order of ObF in (9.2), and an exponential function, i.e.,

$$\hat{y}_i(t) = \sum_{j=0}^{k-2} a_{j,i}(t - mT)^j + b_{1,i} \exp(b_{2,i}(t - mT)) \quad (9.10)$$

where the coefficients $a_{j,i}$, $b_{1,i}$ and $b_{2,i}$ are undetermined coefficients. Those coefficients are computed by requiring that the first k terms of the Taylor expansion of the $\hat{y}_i(t)$ match those k terms of the i^{th} component of the envelope waveform $\mathbf{y}(t)$, when both are expanded around $t = mT$ and evaluated at $t = nT$. This process would then produce the following results for those coefficients

$$b_{2,i} = \frac{y_{i,m}^{(k-1)}}{y_{i,m}^{(k-2)}}, \quad (9.11)$$

$$b_{1,i} = \frac{y_{i,m}^{(k-1)}}{b_{2,i}} \quad (9.12)$$

$$a_{j,i} = \frac{y_{i,m}^{(j)}}{j!} - b_{1,i} \frac{b_{2,i}^j}{j!} \quad (9.13)$$

The i^{th} component in the initial guess vector $\tilde{\mathbf{y}}_n(0)$, used in starting the NR iteration, is then obtained by substituting from (9.11)-(9.13) into (9.10) for $t = nT$.

In other words, the i^{th} component in the initial guess vector is taken as follows,

$$\tilde{y}_{i,n}^{(0)} = \sum_{j=0}^{k-2} \frac{x_{i,m}^{(j)}}{j!} H^j + \frac{y_{i,m}^{(k-1)}}{b_{2,i}^{k-1}} \left[\exp(b_{2,i} H) - \sum_{j=0}^{k-2} \frac{b_{2,i}^j}{j!} \right] \quad (9.14)$$

To this extent, it is worth highlighting some important points:

- The formula in (9.14) provides an approximation to $\tilde{x}_{i,n}^{(0)}$ with an error of $O(H^{k+1})$. More precisely the LTE is given by [81]

$$\begin{aligned} LTE &= |\tilde{y}_{i,n}^{(0)} - y_i(nT)| \\ &= \left| \frac{1}{(k+1)!} \left[y_{i,n}^{(k+1)}(nT) - \frac{(y_{i,n}^{(k)}(nT))^2}{(y_{i,n}^{(k-1)}(nT))} \right] H^{k+1} \right| + O(H^{k+2}) \end{aligned}$$

- The coefficients required to construct the initial guess in (9.14) are readily available. Indeed, these coefficients are simply the individual components in $\mathbf{y}_m^{(i)}$, the past point in the envelope already calculated, and used on the right-side of (9.2) and evaluated via the finite difference formula in (9.6). Thus computing those coefficients does not represent additional computational efforts,
- The above process of constructing the initial guess may breakdown if either, or both of, $x_{i,m}^{(k-1)}$ or $x_{i,m}^{(k-2)}$ happens to be equal to 0, or if $x_{i,m}^{(k-1)} \cdot x_{i,m}^{(k-2)} = 0$. That situation may be encountered, for example, if the i^{th} component of the MNA state variables is a voltage for a node connected to an independent DC biasing voltage source, in which case the high-order derivatives are naturally zeros. Handling such a

situation is typically done by adopting a vector-based [82] form of the above procedure to circumvent this difficulty.

9.3 Derivatives Approximation Using Enhanced Finite-Difference Formula

As mentioned earlier, the utilization of the ObF formula (9.6) in EF requires a suitable method to approximate the high-order derivatives. The general finite difference formula that was used in (9.5) provided an illustrative example for that purpose. This section provides advanced method that produces more accurate estimation for the high-order derivatives. The presented method takes advantage of all the envelope points that are readily available from the ObF-EF to provide more accurate approximation of the derivatives.

An alternative approach to approximating high-order derivatives is based on the Taylor series expansion [83]. In this approach, the j^{th} order derivative is approximated as follows

$$\mathbf{y}_n^{(j)} \approx \frac{1}{T^j} \sum_{i=0}^k g_{i,j,k} \mathbf{y}_{n+i} \quad (9.15)$$

where the coefficients $g_{i,j,k}$ are available in a (problem-independent) tabular form that can be found in [84]. It can be shown [84] that the above formula yields an approximation with an error of order T^{k+1} to the actual envelope waveform.

To illustrate the advantage of this alternative formula, consider its application in the ObF-EF of order-6 to approximate $\mathbf{y}_m^{m+i}$. In this case, $\mathbf{y}_{m+i}$, $i = 0, \dots, 3$ are

envelope points that are available from the previous time-step. Using these points , $\mathbf{y}_m^{(i)}$ can be now approximated as follows [84]

$$\mathbf{y}_m^{(1)} \approx \frac{2\mathbf{y}_{m+3} - 9\mathbf{y}_{m+2} + 18\mathbf{y}_{m+1} - 11\mathbf{y}_m}{6T} \quad (9.16)$$

$$\mathbf{y}_m^{(2)} \approx \frac{-\mathbf{y}_{m+3} + 4\mathbf{y}_{m+2} - 5\mathbf{y}_{m+1} + 2\mathbf{y}_m}{T^2} \quad (9.17)$$

$$\mathbf{y}_m^{(3)} \approx \frac{\mathbf{y}_{m+3} - 3\mathbf{y}_{m+2} + 3\mathbf{y}_{m+1} - \mathbf{y}_m}{T^3} \quad (9.18)$$

One should note that the above approximations yields an error of order T^4 . By contrast, using the finite difference formula (9.5) to approximate $\mathbf{y}_m^1$ and $\mathbf{y}_m^{(2)}$ yields an error of orders T^2 and T^3 , respectively.

Using (9.15), the proposed ObF-EF can now be written as

$$\sum_{i=0}^k \sum_{j=0}^k g_{i,j,k} \alpha_j \frac{H^j}{T^j} \mathbf{y}_{n+i} = \sum_{i=0}^l \sum_{j=0}^l g_{i,j,l} \beta_j \frac{H^j}{T^j} \mathbf{y}_{m+i} \quad (9.19)$$

where $g_{0,0,k} = g_{0,0,l} = 1$ and $g_{i,0,k} = g_{i,0,l} = 0$.

9.4 Stability of the ObF-based Envelope Following

Stability is one of the most important properties when dealing with integration methods. Typically, any numerical method is required to be stable for any stable circuits, otherwise, it may cause the solution to grow without bounds in time, thus rendering the results of limited value. In this section, we prove that using the ObF along with EF techniques leads to A -stable EF methods.

The A -stability criteria of any numerical method can be examined by studying the behavior of the method when applied to the scalar equation $x^{(1)} = \lambda x$. This is known in the literature as the Dalquist test equation.

Definition 8. *An integration method is said to be A -stable if the numerical solution of the scalar test equation $x^{(1)} = \lambda x$, where $\mathcal{R}e(\lambda) < 0$, approaches zero as $t \rightarrow \infty$ for any step size and any number of steps.*

We start by assuming that the integration with a fine step sizes starting from x_m and x_n does not introduce any error. Then, x_{n+i} and x_{m+i} can be written as

$$x_{n+i} = e^{i\lambda T} x_n, \quad i = 1, \dots, k \quad (9.20)$$

$$x_{m+i} = e^{i\lambda T} x_m, \quad i = 1, \dots, l \quad (9.21)$$

Theorem 2. *The ObF-based EF method is A -stable iff $k - 2 \leq l \leq k$.*

Proof. Applying the proposed ObF-EF in (9.6) method to the Dalquist test equation yields

$$\frac{x_n}{x_m} = \left[\frac{\beta_0 + \beta_1 \frac{H}{T} (e^{\lambda T} - 1) + \dots + \beta_l \frac{H^l}{T^l} (e^{\lambda T} - 1)^l}{\alpha_0 + \alpha_1 \frac{H}{T} (e^{\lambda T} - 1) + \dots + \alpha_k \frac{H^k}{T^k} (e^{\lambda T} - 1)^k} \right] \quad (9.22)$$

Setting $z = \frac{H}{T} (e^{\lambda T} - 1)$ results in

$$x_n = \mathcal{Q}_{(l,k)}(z)x_m \quad (9.23)$$

where

$$\mathcal{Q}_{(l,k)}(z) = \left[\frac{\beta_0 + \beta_1 z + \beta_2 z^2 + \cdots + \beta_l z^l}{\alpha_0 + \alpha_1 z + \alpha_2 z^2 + \cdots + \alpha_k z^k} \right] \quad (9.24)$$

In order to have the numerical solution approaches zero (i.e., $x_m > x_n$) for any step size, it is required that the absolute value $|\mathcal{Q}_{(l,k)}(z)| \leq 1$ for any step size H .

It was shown in [85] that $\mathcal{Q}_{(l,k)}(z)$ is the Padé approximation to the exponential function. In addition, it is proved in [50, 86] that for all $k - 2 \leq l \leq k$, the Padé table satisfies the A-stability criteria,

$$|\mathcal{Q}_{(l,k)}(z)| \leq 1, \quad \text{if } \mathcal{R}e(z) \leq 0. \quad (9.25)$$

Since $z = \frac{H}{T}(e^{\lambda T} - 1)$ and $\mathcal{R}e(\lambda) \leq 0$, then $\mathcal{R}e(z) \leq 0$ is always true for any value of H . This in turn results in $|\mathcal{Q}_{(l,k)}(z)| \leq 1$ according to (9.25).

□

9.5 Parallel Implementation of ObF-EF

In this section, we present a parallel EF framework that exploits the rapid advances in hardware platforms, mainly the emergence of the multi-core and distributed com-

puters. The new ObF-EF method is based on modifying the high-order EF technique presented in this chapter to make it suitable for parallel implementation.

Consider the r^{th} NR iteration in (9.8), the sensitivity matrices are calculated by initiating a SPICE-based simulation starting from $t = nT$ and terminating it at $t = (n + k)T$ with an initial point $\tilde{\mathbf{y}}_n^{(r)}$. Using the results of this simulation, the sensitivity matrices can be calculated [22, 54]. The main idea of the parallel ObF-EF method is to partition the SPICE simulation time $[t_n : t_{n+k}]$ into k independent sub-intervals; i.e. $[t_n : t_{n+1}]$, $[t_{n+1} : t_{n+2}]$, $\dots$, $[t_{n+k-1} : t_{n+k}]$. By dividing the time interval, the SPICE based simulation, which is required to find the sensitivity matrices, can be executed in parallel for each interval.

We start by changing the unknown vector of the r^{th} NR iteration to be the vector $\mathbf{s}_n$ defined as

$$\tilde{\mathbf{s}}_n^{(r)} = [\tilde{\mathbf{y}}_n^{(r)} \quad \tilde{\mathbf{y}}_{n+1}^{(r)} \quad \dots \quad \tilde{\mathbf{y}}_{n+k}^{(r)}]^\top \quad (9.26)$$

Each vector-entry of $\tilde{\mathbf{s}}_n^{(r)}$ can also be represented using the state-transition matrix as

$$\tilde{\mathbf{y}}_{n+i}^{(r)} = \phi(\mathbf{y}_{n+i-1}^{(r)}), \quad i = 1, \dots, k \quad (9.27)$$

Using (9.27) and the ObF-EF in (9.6) as the governing relation of the envelope waveform, results in the following nonlinear set of equations

$$\mathbf{F}(\mathbf{s}_n) = \begin{bmatrix} \phi(\tilde{\mathbf{y}}_n^{(r)}) - \tilde{\mathbf{y}}_{n+1}^{(r)} \\ \phi(\tilde{\mathbf{y}}_{n+1}^{(r)}) - \tilde{\mathbf{y}}_{n+2}^{(r)} \\ \vdots \\ \phi(\tilde{\mathbf{y}}_{n+k-1}^{(r)}) - \tilde{\mathbf{y}}_{n+k}^{(r)} \\ \Psi(\tilde{\mathbf{s}}_n^{(r)}) \end{bmatrix} = 0 \quad (9.28)$$

where $\Psi(\tilde{\mathbf{s}}_n^{(r)})$ is the difference between the LHS and RHS of the ObF-EF in (9.6).

The set of nonlinear equations in (9.28) can be solved using NR iterative technique using

$$\tilde{\mathbf{s}}_n^{(r+1)} = \tilde{\mathbf{s}}_n^{(r)} - \bar{\mathbf{J}}^{-1} \mathbf{F}(\tilde{\mathbf{s}}_n^{(r)}) \quad (9.29)$$

where $\bar{\mathbf{J}}$ is the Jacobian matrix and is given by

$$\bar{\mathbf{J}} = \begin{bmatrix} \frac{d\phi(\tilde{\mathbf{y}}_n^{(r)})}{d\tilde{\mathbf{y}}_n^{(r)}} & -\mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \frac{d\phi(\tilde{\mathbf{y}}_{n+1}^{(r)})}{d\tilde{\mathbf{y}}_{n+1}^{(r)}} & -\mathbf{I} & \ddots & \mathbf{0} \\ \vdots & \ddots & \ddots & \dots & \vdots \\ \mathbf{0} & \dots & \frac{d\phi(\tilde{\mathbf{y}}_{n+k-1}^{(r)})}{d\tilde{\mathbf{y}}_{n+k-1}^{(r)}} & -\mathbf{I} \\ \frac{d\Psi(\tilde{\mathbf{s}}_n^{(r)})}{d\tilde{\mathbf{y}}_n^{(r)}} & \frac{d\Psi(\tilde{\mathbf{s}}_n^{(r)})}{d\tilde{\mathbf{y}}_{n+1}^{(r)}} & \dots & \frac{d\Psi(\tilde{\mathbf{s}}_n^{(r)})}{d\tilde{\mathbf{y}}_{n+k}^{(r)}} \end{bmatrix} \quad (9.30)$$

with

$$\frac{d\Psi(\tilde{\mathbf{s}}_n^{(r)})}{d\tilde{\mathbf{y}}_{n+i}^{(r)}} = \sum_{j=i}^k (-1)^{(j-i)} \alpha_j \frac{H^j}{T^j} \begin{pmatrix} j \\ j-i \end{pmatrix} \mathbf{I} \quad (9.31)$$

The NR iterations start by an initial guess $\tilde{\mathbf{y}}_{n+i}^{(0)}$, $i = 0, \dots, k$, to formulate the vector $\tilde{\mathbf{s}}_n^{(0)}$. Next, for each $j = 0, \dots, k-1$, a SPICE-based transient simulation is executed in parallel starting from $\mathbf{x}_{n+j}^{(0)}$ at $t = (n+j)T$ and terminated at $t = (n+j+1)T$ to find $\phi(\tilde{\mathbf{y}}_{n+j}^{(0)})$. The sensitivity matrices $d\phi(\tilde{\mathbf{y}}_{n+j}^{(0)})/d\tilde{\mathbf{y}}_{n+j}^{(0)}$ are calculated using the results of this simulation. The Jacobian matrix is constructed using the sensitivity matrices found in the previous step and a new update $\tilde{\mathbf{s}}_n^{(1)}$ is then computed using (9.29).

It is worth emphasizing that:

- The optimum number of required CPUs to execute the SPICE simulation in parallel is equal to k ,
- Although the size of the Jacobian matrix in (9.30) is now $N(k+1) \times N(k+1)$ (compared to $N \times N$ in (9.9)), the extra computational time needed for the LU factorization is well counter-balanced by the superior reduction in the CPU-time gained by partitioning the total time into k sub intervals and executing the SPICE simulation in parallel.

9.6 Numerical Examples

This section presents three simulation examples. The circuits used in these examples were simulated using three different approaches:

- The proposed approach (ObF-EF),
 - The low-order TR-EF, and
-

- The classical (non-EF) SPICE-based simulation to generate the detailed highly oscillatory waveform.

The results (envelope points) obtained from the proposed EF technique were also used to initiate a detailed SPICE simulation in order to compare the accuracy of the proposed algorithm in simulating the detailed waveform at any point. The second example verifies the accuracy improvements resulting from using the proposed finite-difference formula (9.15) for estimating the higher-order derivatives. The third example illustrates the advantage of using the initial guess (9.14) in reducing the number of iterations.

9.6.1 Example 1

Figure 9.1: Block diagram of the circuit used in Example 1

In this example, we consider a mixer circuit followed by a low pass filter, as represented by the block diagram in Fig. 9.1. The mixer circuit is shown in Fig. 9.2. The Op-Amp used in the low-pass filter is modeled using ideal voltage-controlled voltage-source with a gain equals to 10000.

Figure 9.2: Schematic of the mixer circuit (Example 1)

Table 9.4 compares the CPU-time of the EF-TR and the proposed ObF-EF (order 4). As shown, using high-order ObF-EF allows using larger step size without losing accuracy.

Table 9.1: CPU-time Comparison (Example 1)

Order	Max. Abs. error	H (Step size)	Total CPU time	Speedup w.r.t SPICE
EF-TR (Order 2)	0.030	4T	55.89 sec	4.9
EF-ObF (order 4) ($k = 2, l = 2$)	0.022	13T	19.96 sec	13.9
SPICE-Based Transient Analysis			277.07 sec	

Fig. 9.3 shows the envelope waveform obtained using the proposed method compared to the conventional transient-simulation. Fig. 9.4 shows the waveform obtained when one of the obtained envelope points is used as an initial point and a SPICE simulation is used to plot the detailed response.

Next, Fig. 9.5 compares the time-domain steady-state obtained using the proposed

Figure 9.3: Output waveform (Example 1)

Figure 9.4: Detailed simulation using the envelope points (Example 1)

method versus the steady-state obtained using Harmonic-Balance method. As seen, the steady state responses are in an excellent agreement. It is worth noting that the EF method is also able to provide details for the transient response of the circuit.

9.6.2 Example 2

In this example, we consider a band pass filter (Fig. 9.6). Filter circuits are often used in RF circuit components, and typically lead to long transient response with fast oscillations and are therefore ideal vehicles to test the proposed EF approach.

Figure 9.5: Steady-State response at the output node (Example 1)

Figure 9.6: Bandpass filter. $R_1 = R_6 = 10k\Omega$, $R_2 = 1k\Omega$, $R_3 = 650k\Omega$, $R_4 = R_5 = 15.9k\Omega$, $C_1 = C_2 = 1nF$ (Example 2).

The OpAmp used in the filter was constructed from a two-stage amplifier as shown in Fig. 9.7.

Table 9.2 shows the CPU-time and maximum absolute error obtained using the above-mentioned three approaches. As seen, employing the proposed ObF-EF allows using very large step sizes without losing accuracy. The last column shows the speedup obtained w.r.t. TR-EF.

Figure 9.7: The OpAmp circuit used in Example 2

Table 9.2: CPU-time Comparison (Example 2)

Order	Max. Abs. error	H (Step size)	Total CPU time	Speedup w.r.t SPICE
EF-TR (Order 2)	0.34	5T	410.23 sec	2.1
EF-ObF (order 4) ($k = 2, l = 2$)	0.32	20T	229.97 sec	3.7
EF-ObF (order 6) ($k = 3, l = 3$)	0.30	40T	184.01 sec	4.6
SPICE-Based Transient Analysis			850.90 sec	

9.6.3 Example 3

The tuned amplifier circuit, shown in Fig. 9.8, has been considered for this example. In this example, we show the improvement obtained when using the explicit formula in (9.14) to provide a good initial guess to initiate the Newton-Raphson iterations at each time step.

Table 9.3 compares the total number of Newton-Raphson iterations required in two

Figure 9.8: Tuned amplifier circuit (Example 3)

different cases: using the explicit method presented in Section 9.2 to provide a good initial guess (*referred to as Proposed Initial Guess*) and using the previously obtained envelope-point as an initial guess, i.e., $\tilde{\mathbf{y}}_n^{(0)} = \mathbf{y}_m$ (*referred to as Conventional Initial Guess*). As seen, a reduction in the total number of iterations is obtained resulting in a good speedup improvement. It should be mentioned that in both cases the maximum absolute error in the output waveform compared to conventional SPICE simulation was less than 0.003.

Table 9.3: Effect of Explicit-Predictor on the total number of iterations (Example 3)

EF-ObF	Conventional Initial Guess		Proposed Initial Guess	
	Iterations	CPU-time	Iterations	CPU-time
Order 4	19	83.11	14	55.25
Order 6	15	66.25	11	40.15

Figure 9.9: DC-DC converter circuit.

9.6.4 Example 4

We consider a DC-DC converter circuit shown in Fig. 9.9 [87]. The circuit steps down the input DC voltage from 12V to 5V. The classical SPICE-based transient analysis of this circuit on a single core required 1192.47 seconds.

Next, the circuit is analyzed using the proposed parallel ObF-EF method with order 6, obtained by setting $k = l = 3$, and the necessary envelope-points are obtained. The generated envelope-points are utilized to start SPICE analysis with a fine step size to compute the required detailed response corresponding to the time-interval of interest.

Fig. 9.10 shows the detailed response of the inductor current waveform. Table 9.4 shows the CPU time speedup obtained using the proposed method as a function of the number of processors.

Table 9.4: CPU-time Comparison for Obf-EF Order 6 with $H = 25T$

No. of Processors	CPU-time (Parallel ObF-EF)	Speedup w.r.t. SPICE
1	218.26 sec	5.5
2	155.77 sec	7.7
3	84.86 sec	14.1

Figure 9.10: Sample of the transient response for the inductor current waveform.

9.7 Summary

A high-order envelope following method, based on high-order ObF, is presented. Using the proposed high-order formulation, larger step sizes can be used to find the envelope points without losing accuracy. In addition, using larger step sizes allows skipping many high frequency cycles of the waveform, thus providing more speedups compared to low-order methods, such as TR based EF. In addition, a high-order Finite-Difference formula and an initial guess estimation method are presented to increase the accuracy and efficiency of the proposed method. Finally, a parallel implementation for the ObF-EF method is presented to take advantage of multicore machine.

Chapter 10

Summary and Future Work

This thesis presented time-domain analysis methods for very large and tightly coupled interconnects. In Chapter 4, an overlapping partitioning method is presented for the transient analysis of tightly coupled interconnect circuits. In this method, the multisplitting partitioning technique is applied to the Telegrapher's equations in order to provide high convergence rate for WR methods. In addition, it is shown in Chapter 5 that a reduction in the CPU-time can be obtained when MOR techniques are used to reduce the size of the partitioned subcircuits.

Chapter 6 presented a high-order and stable ObF-based integration methods for interconnect circuits simulation. It was shown in Chapter 6 that using ObF-based integration methods provides a significant reduction for the transient simulation time. Finally, Chapter 7 showed how the ObF-based methods can be extended to include large linear circuits that are terminated with nonlinear devices.

Finally, Chapter 9 presented a new high-order envelope-following method that is geared towards transient simulation of highly oscillatory circuits. The new method exploits the merits of ObF to provide efficient simulation.

10.1 Future Work

Based on the work in this thesis, several future research can be conducted:

1. **Increasing the scalability of parallel ObF-based method:** The partitioning in the parallel ObF-based method presented Chapter 8 is done such that the large linear circuit appears in one partition. The scalability of the method can be increased if the linear subcircuit can be further partitioned using Node tearing method. This technique can increase the scalability of the method when run on a machine with large number of processors.
 2. **Variability analysis of highly oscillatory circuits using EF technique:** Traditionally, variability analysis is performed using Monte Carlo (MC) simulation. For many years, Monte Carlo (MC) based methods have been used in several fields, such as electric circuits, fluid-dynamics, chemistry and medical applications, to estimate the effect of fabrication uncertainties. The main idea of MC is to repeat the simulation with different values for each uncertain parameter from its uncertainty range in order to estimate the probability density function of the output. Using The EF method along with MC simulation can accelerate the simulation in the case of oscillatory circuits.
-

Appendix A

Kronecker Product

Definition: If $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{B} \in \mathbb{R}^{l \times k}$, then the Kronecker product $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{nl \times mk}$ is given by [88]:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \dots & a_{1,m}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}\mathbf{B} & a_{n,2}\mathbf{B} & \dots & a_{n,m}\mathbf{B} \end{bmatrix} \quad (\text{A.1})$$

where $a_{i,j}$ is the entry at row i and column j of the matrix $\mathbf{A}$.

Appendix B

Illustrative Example for the Node Tearing algorithm

This example presents the node tearing algorithm for a simple circuit shown in Figure B.1. Here we assume that the nonlinear partition include a resistance in series with the diode R_2 . First, a short circuit is connected between the linear partition and the diode as shown in Figure B.1. The MNA equations can then be written as follows.

Figure B.1: An example used to illustrate the partitioning technique, with $N = 2$ and $P = 2$.

$$\begin{aligned}
 & \begin{bmatrix} \mathbf{G}_{\text{lin}} & \mathbf{0} & \mathbf{\Xi}_1 \\ \mathbf{0} & \mathbf{G}_{\text{nonlin}} & \mathbf{\Xi}_2 \\ \mathbf{\Xi}_1^\top & \mathbf{\Xi}_2^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\text{lin}} \\ \mathbf{x}_{\text{nonlin}} \\ \gamma \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{\text{lin}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{\text{nonlin}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}}_{\text{lin}} \\ \dot{\mathbf{x}}_{\text{nonlin}} \\ \dot{\gamma} \end{bmatrix} \\
 & + \begin{bmatrix} \mathbf{0} \\ \mathbf{f}(\mathbf{x}_{\text{nonlin}}(t)) \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\text{lin}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \quad (\text{B.1})
 \end{aligned}$$

where

- $\mathbf{G}_{\text{lin}} = \begin{bmatrix} \frac{1}{R_1} & \frac{-1}{R_1} \\ \frac{-1}{R_1} & \frac{1}{R_1} \end{bmatrix}, \quad \mathbf{G}_{\text{nonlin}} = \begin{bmatrix} \frac{1}{R_2} & \frac{-1}{R_2} \\ \frac{-1}{R_2} & \frac{1}{R_2} \end{bmatrix},$
- $\mathbf{C}_{\text{lin}} = \begin{bmatrix} C & -C \\ -C & C \end{bmatrix}, \quad \mathbf{C}_{\text{nonlin}} = \mathbf{0},$
- $\mathbf{\Xi}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top, \quad \mathbf{\Xi}_2 = \begin{bmatrix} -1 & 0 \end{bmatrix}^\top,$
- $\mathbf{b}_{\text{lin}} = \begin{bmatrix} J & 0 \end{bmatrix}^\top,$
- $\dot{\mathbf{x}}_{\text{lin}} = [v_1 \ v_2]^\top, \quad \dot{\mathbf{x}}_{\text{nonlin}} = [v_3 \ v_4]^\top$
- γ is the current in the link connecting the two subcircuits and
 $\mathbf{f}(\mathbf{x}_{\text{nonlin}}(t)) = [0 \ f(v_4)]^\top$ with $f(v_4)$ is a nonlinear function representing the current flowing in the diode.

References

- [1] N. Nakhla, A. Ruehli, M. Nakhla, and R. Achar, “Simulation of coupled interconnects using waveform relaxation and transverse partitioning,” *IEEE Transactions on Advanced Packaging*, vol. 29, pp. 78–87, Feb. 2006.
- [2] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equation*. Philadelphia: Society for Industrial and Applied Mathematics (SIAM), July 2007, ch. 7.
- [3] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*. New York: Van Nostrand Reinhold, 1983.
- [4] J.-F. Mao and E. Kuh, “Fast simulation and sensitivity analysis of lossy transmission lines by the method of characteristics,” *IEEE Transactions on Circuits and Systems—Part I: Fundamental Theory and Applications*, vol. 44, no. 5, pp. 391–401, 1997.
- [5] M. Celik, A. C. Cangellaris, and A. Yaghmour, “An all purpose transmission line model for interconnect simulation in SPICE,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 45, no. 10, pp. 1857–1867, Oct. 1997.

-
- [6] A. Dounavis, R. Achar, and M. Nakhla, "A general class of passive macro-models for lossy multiconductor transmission lines," *IEEE Transactions on Microwave Theory and Techniques*, vol. 49, no. 10, pp. 1686–1696, Oct. 2001.
- [7] N. M. Nakhla, A. Dounavis, R. Achar., and M. S. Nakhla, "DEPACT: Delay-extraction-based compact transmission-line macromodeling algorithm." *IEEE Transactions on Advanced Packaging*, vol. 28, no. 1, pp. 13–23, Feb. 2005.
- [8] D. O’Leary and R. White, "Multisplittings of matrices and parallel solution of linear systems," *SIAM J. Algebraic Discrete Methods*, pp. 630–440, 1985.
- [9] M. Jones and D. Szyld, "Two-stage multisplitting methods with overlapping blocks," *Numerical Linear Algebra with Applications*, vol. 3, pp. 113–124, 1996.
- [10] R. Jeltsch and B. Pohl, "Waveform relaxation with overlapping splittings," *SIAM Journal on Scientific Computing (SISC)*, vol. 1, pp. 40–49, 1995.
- [11] K. Kundert, *The Designer’s Guide to SPICE and Spectre*. Norwell, MA: Kluwer Academic Publishers, 1995.
- [12] C. Gear, "Simultaneous numerical solution of differential-algebraic equations," *IEEE Transactions on Circuit Theory*, vol. 18, no. 1, pp. 89–95, Jan. 1971.
-

-
- [13] G. Dahlquist, "A special stability problem for linear multistep methods," *BIT*, vol. 3, pp. 27–43, 1963.
- [14] E. Gad, M. Nakhla, R. Achar, and Y. Zhou, "A-stable and L-stable high-order integration methods for solving stiff differential equations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 9, pp. 1359–1372, Sep. 2009.
- [15] N. Obreshkov, "Sur les quadrature mecanique," (*Bulgarian, French Summary*) *Akad. Nauk.*, vol. 65, pp. 191–289, 1942.
- [16] Y. Zhou, E. Gad, M. S. Nakhla, and R. Achar, "Structural characterization and efficient implementation techniques for A-stable high-order integration methods," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 1, pp. 101–108, Jan. 2012.
- [17] E. Ngoya and R. Larcheveque, "Envelop transient analysis: a new method for the transient and steady state analysis of microwave communication circuits and systems," in *IEEE MTT-S International Microwave Symposium Digest*, vol. 3, June 1996, pp. 1365–1368.
- [18] E. de Cos, S. Sancho, and A. Suarez, "Noise analysis of modulated self-oscillating mixers," in *IEEE MTT-S International Microwave Symposium Digest*, vol. 3, June 2004, pp. 1875–1878.
- [19] E. Ngoya, J. Rousset, and D. Argollo, "Rigorous RF and microwave oscillator phase noise calculation by envelope transient technique," in *Microwave*
-

- Symposium Digest. 2000 IEEE MTT-S International*, vol. 1, June 2000, pp. 91–94.
- [20] J. Roychowdhury, “Making fourier-envelope simulation robust,” in *IEEE/ACM International Conference on Computer Aided Design, ICCAD*, Nov 2002, pp. 240–245.
- [21] L. Petzold, “An efficient numerical method for highly oscillatory ordinary differential equations,” *SIAM J.Numer. Anal.*, vol. 18, no. 3, 1981.
- [22] A. Brambilla and P. Maffezzoni, “Envelope following method for the transient analysis of electrical circuits,” *IEEE Transactions on Circuits and Systems—Part I: Fundamental Theory and Applications*, vol. 47, no. 7, pp. 999–1008, 2000.
- [23] —, “Envelope-following method to compute steady-state solutions of electrical circuits,” *IEEE Transactions on Circuits and Systems—Part I: Fundamental Theory and Applications*, vol. 50, no. 3, pp. 407–417, 2003.
- [24] E. Hairer, S. P. Nøsett, and G. Wanner, *Solving Ordinary Differential Equations I Nonstiff Problems*, 3rd ed. Berlin Heidelberg: Springer, 2008.
- [25] M. Farhan, N. Nakhla, M. Nakhla, R. Achar, and A. Ruehli, “Overlapping partitioning techniques for simulation of strongly coupled distributed interconnects,” *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 2, pp. 1193–1201, 2012.
-

-
- [26] ———, “Waveform relaxation and overlapping partitioning techniques for tightly coupled interconnects,” in *IEEE 20th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, Oct 2011, pp. 111–114.
- [27] M. Farhan, N. Nakhla, M. Nakhla, and R. Achar, “Fast transient analysis of tightly coupled interconnects via overlapping partitioning and model-order reduction,” *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 4, no. 10, pp. 1648–1656, Oct 2014.
- [28] M. Farhan, E. Gad, M. Nakhla, and R. Achar, “New method for fast transient simulation of large linear circuits using high-order stable methods,” *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 3, pp. 661–669, April 2013.
- [29] ———, “Efficient transient simulation of transmission lines and distributed circuits using high-order stable methods,” in *IEEE MTT-S International Microwave Symposium Digest (MTT)*, June 2012, pp. 1–3.
- [30] ———, “Fast simulation of microwave circuits with nonlinear terminations using high-order stable methods,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, pp. 360–371, 2013.
- [31] ———, “An efficient method for transient simulation of high-speed interconnects with nonlinear terminations,” in *IEEE 21st Conference on Electrical*
-

-
- Performance of Electronic Packaging and Systems (EPEPS)*, Oct 2012, pp. 119–122.
- [32] —, “Parallel simulation of large linear circuits with nonlinear terminations using high-order stable methods,” *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 4, no. 7, pp. 1201–1211, July 2014.
- [33] —, “A multi-core high-order A-stable and L-stable integration method for fast transient simulation of high-speed interconnect and transmission line circuits,” in *2012 IEEE 16th Workshop on Signal and Power Integrity (SPI)*, 2012, pp. 89–92.
- [34] —, “High order and a-stable envelope following method for transient simulations of oscillatory circuits,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 62, no. 12, pp. 3309–3317, Dec 2014.
- [35] M. A. Farhan, E. Gad, M. Nakhla, and R. Achar, “High-order envelope following method for accurate transient analysis of almost periodic electrical circuits,” in *Microwave Symposium Digest (MTT), 2014 IEEE MTT-S International*, June 2014.
- [36] M. Farhan, E. Gad, M. Nakhla, and R. Achar, “High order envelope following method for parallel simulation of power converter circuits,” in *IEEE Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, Oct. 2014.
-

-
- [37] N. Nakhla, A. Ruehli, M. Nakhla, R. Achar, and C. Changzhong, "Waveform relaxation techniques for simulation of coupled interconnects with frequency-dependent parameters," *IEEE Transactions on Advanced Packaging*, vol. 30, pp. 257–269, May 2007.
- [38] D. Paul, N. Nakhla, R. Achar, and M. Nakhla, "Parallel simulation of massively coupled interconnect networks," *IEEE Transactions on Advanced Packaging*, vol. 33, pp. 115–127, Feb. 2010.
- [39] U. Miekkala and O. Nevanlinna, "Convergence of waveform relaxation method," in *IEEE International Symposium on Circuits and Systems, 1988.*, June 1988, pp. 1643–1646 vol.2.
- [40] K. Urahama, "Local convergence of waveform relaxation method," in *Electronics and Communications in Japan, Part 1: Communications 71*, 1988, pp. 52–60.
- [41] Z. Jackiewicz and M. Kwapisz, "Convergence of waveform relaxation methods for differential-algebraic systems," *SIAM Journal on Numerical Analysis*, p. 23032317, 1996.
- [42] A. Ruehli and T. Johnson, *Circuit Analysis Computing by Waveform Relaxation*, in *Encyclopedia of Electrical and Electronics Engineering*. New York: John Webster, Wiley, 1999.
- [43] E. Lelarsmee, A. Ruehli, and A. Sangiovanni-Vincentelli, "The waveform relaxation method for time-domain analysis of large scale integrated cir-
-

-
- circuits,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 1, pp. 131 – 145, July 1982.
- [44] F. Y. Chang, “The generalized method of characteristics for waveform relaxation analysis of lossy coupled transmission lines,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 37, pp. 2028 –2038, Dec. 1989.
- [45] R. Wang and O. Wing, “Transient analysis of dispersive VLSI interconnects terminated in nonlinear loads,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, pp. 1258 –1277, Oct. 1992.
- [46] F. Y. Chang, “Waveform relaxation analysis of RLCG transmission lines,” *IEEE Transactions on Circuits and Systems*, vol. 37, pp. 1394 –1415, Nov. 1990.
- [47] C. Paul, *Analysis of Multiconductor Transmission Lines*. New York: Wiley, 1994.
- [48] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, December 1997.
- [49] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*. Wiley, 2003.
- [50] A. Iserles and S. P. Nørsett, *Order Stars*. Chapman & Hall, 1991.
-

-
- [51] C. W. Gear, *Numerical initial value problems in ordinary differential equations*. N.J.: Prentice-Hall, 1971.
- [52] B. L. Ehle, “High-order A-stable methods for the numerical solution of D.E.’S,” *BIT*, vol. 8, pp. 276–278, 1968.
- [53] T. A. Davis and E. Palamadai, *User Guide for KLU and BTF*. Dept. of Computer and Information Science and Engineering, University of Florida, 2009.
- [54] J. White and S. Leeb, “An envelope-following approach to switching power converter simulation,” *IEEE Transactions on Power Electronics*, vol. 6, no. 2, pp. 303–307, 1991.
- [55] A. Ruehli, *Circuit Analysis, Simulation and Design, Part 2*. North-Holland: Elsevier, 1987.
- [56] A. Frommer and B. Pohl, “A comparison result for multisplittings based on overlapping blocks and its application to waveform relaxation methods,” *Numerical Linear Algebra with Applications*, vol. 2, pp. 335–346, 1995.
- [57] G. Alefeld, I. Lenhardt, and G. Mayer, “On multisplitting methods for band matrices,” *Numerische Mathematik*, vol. 75, pp. 267–292, 1997.
- [58] B. Gustavsen and A. Semlyen, “Rational approximation of frequency responses by vector fitting,” *IEEE Transactions on Power Delivery*, vol. 14, no. 3, pp. 1052–1061, July 1999.
-

-
- [59] ———, “Enforcing passivity for admittance matrices approximated by rational functions,” *IEEE Transactions on Power Systems*, vol. 16, no. 1, pp. 97–104, Feb. 2001.
- [60] D. Saraswat, R. Achar, and M. Nakhla, “A fast algorithm and practical considerations for passive macromodeling of measured/simulated data,” *IEEE Transactions on Advanced Packaging*, vol. 27, no. 1, pp. 57–70, 2004.
- [61] HSPICE, *Star-Hspice Manual*, Meta-Software Inc., Mountain View, CA, Synopsis, 2010.
- [62] F. M. Tesche, M. V. Ivanov, and T. Karlsson, *EMC Analysis Methods and Computational Models*. New York: Wiley, 1997.
- [63] J. Arrillaga and N. Watson, *Computer Modelling of Electrical Power Systems*. John Wiley & Sons, 2001.
- [64] A. Odabasioglu, M. Celik, and L. T. Pileggi, “PRIMA: passive reduced-order interconnect macromodeling algorithm,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 8, pp. 645–654, Aug. 1998.
- [65] W. E. Arnoldi, “The principle of minimized iteration in the solution of the matrix eigenvalue problem,” *Quart. Appl. Math.*, vol. 9, pp. 17–29, 1951.
- [66] M. Celik, L. Pileggi, and A. Odabasioglu, *IC Interconnect Analysis*. Kluwer Academic Publishers, 2002.
-

-
- [67] R. Achar and M. Nakhla, "Simulation of high-speed interconnects," *Proceedings of IEEE*, vol. 89, no. 5, pp. 693–728, May 2001.
- [68] J. Silva, J. Villena, P. Flores, and L. Silveira, "Outstanding issues in model order reduction," in *Scientific Computing in Electrical Engineering*. Springer Berlin Heidelberg, 2007, vol. 11, pp. 139–152.
- [69] N. Kapre and A. DeHon, "SPICE² spatial processors interconnected for concurrent execution for accelerating SPICE circuit simulator using an FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 1, pp. 9–22, January 2012.
- [70] G. Wanner, E. Hairer, and S. P. Nøsett, "Order stars and stability theorems," *BIT*, vol. 18, pp. 475–489, 1978.
- [71] N. F. Najm, *Circuit Simulation*. Wiley-IEEE Press, 2010, ch. 5, Solution of Differential Circuit Equations, pp. 282–285.
- [72] T. Pillage, R. Rohrer, and C. Visweswariah, *Electronic Circuit and System Simulation Methods*. New York: McGRAW-HIL, 1994, ch. 8, pp. 217–237.
- [73] D. Paul, M. Nakhla, R. Achar, and N. Nakhla, "Parallel circuit simulation via binary link formulations (PvB)," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 3, no. 5, pp. 768–782, 2013.
- [74] D. Paul, R. Achar, M. Nakhla, and N. Nakhla, "Efficient parallel scheduler for circuit simulation exploiting binary link formulations," in *IEEE Fourth*
-

-
- Latin American Symposium on Circuits and Systems (LASCAS)*, 2013, pp. 1–4.
- [75] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, ser. Computer science and applied mathematics. New York, NY: Academic Press, 1970.
- [76] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*. Springer, 1996, vol. II.
- [77] L. M. Milne-Thomson, *The calculus of finite differences*. Chelsea, New York, 1981.
- [78] G. Engeln-Mullges and F. Uhlig, *Numerical algorithms with C*. Springer, 1996, ch. 6.
- [79] J. Dobe, “Advanced types of the sensitivity analysis in frequency and time domains,” *AEU - International Journal of Electronics and Communications*, vol. 63, no. 1, pp. 52 – 64, 2009.
- [80] T. J. Aprille, Jr. and T. N. Trick, “A computer algorithm to determine the steady-state response of nonlinear oscillators,” *IEEE Transactions on Circuit Theory*, vol. 19, no. 4, pp. 354–360, July 1972.
- [81] X. Wu and J. Xia, “Study of general Taylor-like explicit methods in solving stiff ordinary differential equations,” *International Journal of Computer Mathematics*, vol. 84, pp. 1795–1803, 2007.
-

-
- [82] X.-Y. Wu and J.-L. Xia, “The vector form of a sixth-order A-stable explicit one-step method for stiff problems,” *Computers and Mathematics with Applications*, vol. 39, no. 34, pp. 247 – 257, 2000.
- [83] I. R. Khan and R. Ohba, “New finite difference formulas for numerical differentiation,” *Journal of Computational and Applied Mathematics*, vol. 126, no. 12, pp. 269 – 276, 2000.
- [84] —, “Taylor series based finite difference approximations of higher-degree derivatives,” *Journal of Computational and Applied Mathematics*, vol. 154, no. 1, pp. 115 – 124, 2003.
- [85] G. A. Baker Jr., *Essentials of Padé Approximants*. New York: Academic Press, 1975.
- [86] G. Birkoff and R. S. Varga, “Discretization errors for well-set cauchy problems,” *J. Math. and Physics*, vol. 44, pp. 1–23, 1965.
- [87] R. Muyshondt and P. Krein, “20 w benchmark converters for simulation and control comparisons,” in *1998. 6th Workshop on Computers in Power Electronics*, 1998, pp. 201–212.
- [88] A. J. Laub, *Matrix Analysis for Scientists and Engineers*. Philadelphia: SIAM, 2005.
-