UNIVERSITY OF CALIFORNIA, SAN DIEGO

Exponential Time Integration for Transient Analysis of Large-Scale Circuits

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Hao Zhuang

Committee in charge:

Professor Chung-Kuan Cheng, Chair
Professor Li-Tien Cheng
Professor Bo Li
Professor Bill Lin
Professor Yuan Taur

2016

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Learning the analysis of dynamical systems from large-scale circuits via numerical algorithms has definitely changed the perspective of how I view problems in the real world. Thank Professor Chung-Kuan Cheng for the research guidance during the last four years. Along this Ph.D. journey, I have learned that positive attitude, perseverance, and grind are important factors when addressing a challenging problem. Also I would thank this journey, which confirms my passion for seeking the beauty of mathematics even from daunting engineering problems. Thank Professors Li-Tien Cheng, Bo Li, Bill Lin, and Yuan Taur, who serve as my Ph.D. committee members. Thank Professors Wenjian Yu, Pengwen Chen, Patrick Mercier, Zuochang Ye, and Scott Baden for the research collaboration. Thank John Loffeld, Zheng Zhang, Jianlei Yang, Professors Mike Botchev, Nicholas Higham, Marlis Hochbruck, Alexander Ostermann, and Mayya Tokman for the discussions through emails.

I am very fortunate to have the support from my colleges at ANSYS Apache since 2015 summer. Especially, I owe a great deal of gratitude to Steven P. McCormick and Norman Chang, who always gave me luxurious freedom and encouraged me to keep moving forward. Besides, I also want to thank them for letting me have opportunity to explore and build internal large-scale systems for machine learning and big data processing, which was totally mind-blowing back then. Before my part-time job at ANSYS, I also spent a meaningful summer internship at Synopsys in 2014 with Zongwu Tang, Han Young Koh, Qing Su, Chen Li, Wenwen Chai, and Dustin Liu, from whom I learned the advanced techniques and implementation skills in graph partitioning algorithms and direct matrix solvers for the linear systems from real VLSI designs, which had impacted my later research. Thank Song Han, Wei Wei, Tao Du, and Rui Hu for helping me settle down at Stanford University and enjoyed my time at Nerd Nation during that summer.

I would like to thank Yutao Ma, Bruce W. McGaughy at Proplus Design Solutions, and An-Chang Deng at Cadence Design Systems for their research suggestions. Thank Kambiz Samadi and Yang Du for providing the research opportunity from Qualcomm Research and the support of Qualcomm Fellow-Mentor-Advisor (FMA) Fellowship. I would also thank for the support of Powell Fellowship from UCSD Jacobs School of Engineering. Those two fellowships supported the first three year of my Ph.D. research. I thank Lining Zhang, Chenyue Ma, Professors Xinnan Lin, Bei Yu, Mansun Chan, Quan Chen, and Ngai Wong for hosting me in Shenzhen and Hong Kong.

Thank all my colleagues and friends at San Diego, including but not limited to Tuck-Boon Chan, Wei-Ting Chan, Ryan Coutts, Zhou Fang, Sidi Fu, Kwangsoo Han, Albert Hsu, Xiang Hu, Jae Jung, Ilgweon Kang, Deokseong Kim, Hyein Lee, Jiajia Li, Chia-Hung Liu, Hao Liu, Jeng-Hau Lin, Jingwei Lu, Mulong Luo, Hui Wang, Lutong Wang, Junchuan Wang, Xinan Wang, Xinyuan Wang, Yu Wang, Yute Wang, Shih-Hung Weng, Chicheng Zhang, Lu Zhang, Yunqi Zhang, and Xiang Zhang. Especially, thank Shih-Hung for the guidance in the early year of simulation research. Thank Jingwei for inviting me to ePlace project, which let me enjoy applying numerical optimization techniques quite a bit . Thank Ilgweon for building VLSI Place-and-Route flow to generate test cases even when he was extremely busy. Thank Xinyuan for proofreading this thesis. Thank Mulong and Zhou for the discussion of computer system design. Thank my roommate Chicheng for sharing the cars for the daily commute, plus his insightful discussion about the theory of general machine learning and active learning during the commute almost everyday. I would love to thank Julie Conner, our great graduate student program director, for scheduling my exams and providing many useful pieces of advice.

To my parents and Xiao Liu, I cannot express how much I owe them in words. This thesis is dedicated to my family.

This thesis uses the material from several papers during my Ph.D. research. They are listed as follows:

Chapter 3, in part, is a reprint of the material as it appears in “From Circuit Theory, Simulation to SPICE^{Diego}: A Matrix Exponential Approach for Time Domain Analysis of Large Scale Circuits” by Hao Zhuang, Xinyuan Wang, Quan Chen, Pengwen Chen, and Chung-Kuan Cheng in *IEEE Circuits and Systems Magazine*. The chapter also contains the content from “Simulation Algorithms with Exponential Integration for Time-Domain Analysis of Large-Scale Power Delivery Networks” by Hao Zhuang, Wenjian Yu, Shih-Hung Weng, Ilgweon Kang, Jeng-Hau Lin, Xiang Zhang, Ryan Coutts, and Chung-Kuan Cheng in *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems*. The thesis author was the primary investigator and author of the papers.

Chapter 4, in part, is a reprint of the material as it appears in “Simulation Algorithms with Exponential Integration for Time-Domain Analysis of Large-Scale Power Delivery Networks” by Hao Zhuang, Wenjian Yu, Shih-Hung Weng, Ilgweon Kang, Jeng-Hau Lin, Xiang Zhang, Ryan Coutts, and Chung-Kuan Cheng in *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems*. The chapter also contains the content from “Power Grid Simulation using Matrix Exponential Method with Rational Krylov Subspaces” by Hao Zhuang, Shih-Hung Weng, and Chung-Kuan Cheng in *Proceedings of IEEE International Conference on ASIC 2013*, and “MATEX: A Distributed Framework for Transient Simulation of Power Distribution Networks” by Hao Zhuang, Shih-Hung Weng, Jeng-Hau Lin, and Chung-Kuan Cheng in *Proceedings of IEEE/ACM Design Automation Conference 2014*. The thesis author was the primary investigator and author of the papers.

Chapter 5, in part, is currently being prepared for submission for publication of the material by Hao Zhuang, Wenjian Yu, Deokseong Kim, Xinyuan Wang, and

Chung-Kuan Cheng. The thesis author was the primary investigator and author of this material. This chapter also contains the content from “Dynamic Analysis of Power Delivery Network with Nonlinear Components Using Matrix Exponential Method” by Hao Zhuang, Xinan Wang, Ilgweon Kang, Jeng-Hau Lin, and Chung-Kuan Cheng in *Proceedings of IEEE International Symposium on Electromagnetic Compatibility 2015*, and “An Algorithmic Framework of Large-Scale Circuit Simulation Using Exponential Integrators” by Hao Zhuang, Wenjian Yu, Ilgweon Kang, Xinan Wang, and Chung-Kuan Cheng in *Proceedings of IEEE/ACM Design Automation Conference 2015*. The thesis author was the primary investigator and author of the papers.

VITA

2016	Ph.D. in Computer Science, University of California, San Diego
2015-2016	Ph.D. candidate, University of California, San Diego
2015	C.Phil. in Computer Science, University of California, San Diego
2012-2015	Ph.D. student, University of California, San Diego

PUBLICATIONS

2012-2016

Hao Zhuang, Xinyuan Wang, Quan Chen, Pengwen Chen, and Chung-Kuan Cheng, "From Circuit Theory, Simulation to SPICE^{Diego}: A Matrix Exponential Approach for Time Domain Analysis of Large Scale Circuits," *IEEE Circuits and Systems Magazine*, vol. 16, no. 2, pp. 16-34, 2016.

Hao Zhuang, Wenjian Yu, Shih-Hung Weng, Ilgweon Kang, Jeng-Hau Lin, Xiang Zhang, Ryan Coutts, and Chung-Kuan Cheng, "Simulation Algorithms with Exponential Integration for Time-Domain Analysis of Large-Scale Power Delivery Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems*, vol. PP., no. 99., pp. 1, 2016.

Qinggao Mei, Wim Schoenmaker, Shih-Hung Weng, **Hao Zhuang**, Chung-Kuan Cheng, and Quan Chen, "An Efficient Transient Electro-Thermal Simulation for Power Integrated Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems*, vol. 35, no. 5, pp. 823-843, 2016.

Jingwei Lu, **Hao Zhuang**, Ilgweon Kang, Pengwen Chen, and Chung-Kuan Cheng, "ePlace-3D: Electrostatics based Placement for 3D-ICs," *Proceedings of ACM/IEEE International Symposium on Physical Design*, 2016.

Hao Zhuang, Wenjian Yu, Ilgweon Kang, Xinan Wang, and Chung-Kuan Cheng, "An Algorithmic Framework for Efficient Large-Scale Circuit Simulation using Exponential Integrators", *Proceedings of IEEE/ACM Design Automation Conference*, June 2015.

Jingwei Lu, **Hao Zhuang**, Pengwen Chen, Hongliang Chang, Chin-Chih Chang, Yiu-Chung Wong, Lu Sha, Dennis Huang, Yufeng Luo, Chin-Chi Teng, Chung-Kuan Cheng, "ePlace-MS: Electrostatics based Placement for Mixed-Size Integrated Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems*, vol. 34, no. 5, pp. 685-698, 2015

Hao Zhuang, Xinan Wang, Ilgweon Kang, Jeng-Hau Lin, and Chung-Kuan Cheng, “Dynamic Analysis of Power Delivery Network with Nonlinear Components Using Matrix Exponential Method,” *Proceedings of IEEE International Symposium on Electromagnetic Compatibility*, 2015.

Jeng-Hau Lin, Hao Liu, Chia-Hung Liu, Phillip Lam, Gung-Yu Pan,**Hao Zhuang**, Ilgweon Kang, Patrick P. Mercier, and Chung-Kuan Cheng, “An Interdigitated Non-Contact ECG Electrode for Impedance Compensation and Signal Restoration,” *Proceedings of IEEE Biomedical Circuits and Systems Conference*, 2015.

Hao Zhuang, Shih-Hung Weng, Jeng-Hau Lin, and Chung-Kuan Cheng, “MATEX: A Distributed Framework for Transient Simulation of Power Distribution Networks,” *Proceedings of IEEE/ACM Design Automation Conference*, 2014.

Haibing Su, Hao Liu, Shih-Hung Weng, Hui Wang, Aliasgar Presswala, **Hao Zhuang**, Jeng-Hau Lin, Patrick Mercier, and Chung-Kuan Cheng, “Non-contact Biopotential Sensing System with Motion Artifact Suppression,” *Proceedings of IEEE Conference on Communications, Circuits and Systems*, 2013.

Hao Zhuang, Jingwei Lu, Kambiz Samadi, Yang Du and Chung-Kuan Cheng, “Performance-Driven Placement for Design of Rotation and Right Arithmetic Shifters in Monolithic 3D ICs,” *Proceedings of IEEE Conference on Communications, Circuits and Systems*, 2013.

Hao Zhuang, Shih-Hung Weng, and Chung-Kuan Cheng, “Power Grid Simulation using Matrix Exponential Method with Rational Krylov Subspaces,” *Proceedings of IEEE International Conference on ASIC*, 2013.

Kuangya Zhai, Wenjian Yu, and **Hao Zhuang**, “GPU-Friendly Floating Random Walk Algorithm for Capacitance Extraction of VLSI Interconnects,” *Proceedings of IEEE Design, Automation and Test in Europe*, 2013.

Wenjian Yu, **Hao Zhuang**, Chao Zhang, Gang Hu, and Zhi Liu, “RWCap: A Floating Random Walk Solver for 3-D Capacitance Extraction of VLSI Interconnects,” *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems*, vol. 32, no. 3, pp. 353-366, 2013.

Wenjian Yu, Kuangya Zhai, **Hao Zhuang**, and Junqing Chen, “Accelerated Floating Random Walk Algorithm for the Electrostatic Computation with 3-D Rectilinear-shaped Conductors,” *Elsevier Simulation Modelling Practice and Theory*, 34 (5), pp. 20-36, 2013.

Hao Zhuang, Wenjian Yu, Gang Hu, Zhi Liu, and Zuochang Ye, “Fast Floating Random Walk Algorithm For Multi-dielectric Capacitance Extraction with Numerical Characterization of Green’s Functions,” *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference*, 2012.

ABSTRACT OF THE DISSERTATION

Exponential Time Integration for Transient Analysis of Large-Scale Circuits

by

Hao Zhuang

Doctor of Philosophy in Computer Science

University of California, San Diego, 2016

Professor Chung-Kuan Cheng, Chair

Transient analysis of large-scale circuits relies on efficient numerical time integration algorithms. In this thesis, we focus on the high-order exponential integration and the explicit formulation for solving large-scale dynamical systems of VLSI designs. First, we demonstrate the advantages of exponential integration for the application to linear systems. To accelerate the computation of matrix exponential and vector product, Krylov subspace method and Arnoldi algorithm with different preconditioned matrices are explored. Second, we integrate the exponential integration based algorithms into a simulator for power network analysis, which is a challenging task for modern VLSI signoff. We verify the capability of adaptive stepping with high accuracy and the model of distributed computation. Comparing with the traditional approach, we observe the speedups up to 14X and 98X without the loss of accuracy by single-core and distributed computation models, respectively. Third, we devise a novel integration framework with the explicit formulation for nonlinear dynamical systems. This framework reduces the number of computationally expensive matrix factorizations required by traditional integration approaches. Furthermore, we demonstrate that the Krylov subspace methods can reduce the complexity of strongly coupled dynamical problems such as post layout analysis.

Chapter 1

Introduction and Research Motivation

1.1 Circuit Simulation

Circuit simulation is the standard step to verify the design of integrated circuits. The performance of a circuit design should be verified via circuit simulation before the fabrication processes. SPICE was the first general-purpose circuit simulator developed by L. W. Nagel at University of California, Berkeley in the early 1970s. Since then, SPICE-like simulation tools have become indispensable during the cycle of VLSI design. Many Electronic Design Automation (EDA) tools rely on the circuit-level simulators, such as logic synthesis, power and timing analysis, placement and routing. Some semiconductor companies run SPICE-like simulation over 1 million times per week [3]. As the technology scaling down, multi-core and many-core microprocessors put billions of transistors into a single chip. The complicated interaction among interconnects and nano-scale transistors requires the help from circuit simulation to analyze and verify the unpredictable electronic behavior, such as signal noise, power noise, and post-layout effect. Therefore, the performance and effectiveness become extremely critical in the cycle of IC design. What makes the research field of circuit simulation unique is its multi-disciplinary nature. It is a set of concepts adopted from numerical mathematics, circuit theory, graph theory, device modeling, and software development.

1.2 SPICE-like Simulation and Numerical Time Integration

The flow of SPICE can be described in Fig. 1.1. The input is a circuit netlist. The device evaluation is through device models, e.g., BSIM3 [4]. After the evaluation, the matrix is stamped with corresponding linearized resistance, capacitance, inductance, voltage, and current sources, etc., and form the linearized system. DC analysis is applied to obtain the initial solution for the circuit, which may contain Newton-Raphson iterations if the system contains nonlinear devices.

Transient analysis starts with the DC solution. During the process, numerical integration schemes are used to solve differential equations step by step until the end of the time span specified by the netlist. In each step, the device evaluation is required to form the corresponding linearized dynamical system. When the system is nonlinear, Newton-Raphson iteration is usually performed for the implicit numerical integration approach in order to get a converged solution. Besides, the convergence and error check are used to check whether the solution is accurate enough. The step control is used to decide the step sizes according to the numerical integration errors, e.g., local truncation error (LTE).

Transient simulation is the key component in SPICE [3, 5–8]. As we can see in Fig 1.1, device evaluation and numerical integration are the major components in the flow. Device evaluation is easy to be accelerated by parallel processing, etc. However, it is hard for the acceleration of numerical integration part. Therefore, numerical integration algorithms usually decide the efficiency and accuracy of the simulation tool, especially when the circuits are large and the portion of runtime dominates the overall performance. There has been a large amount of research to improve integration algorithms [9–14].

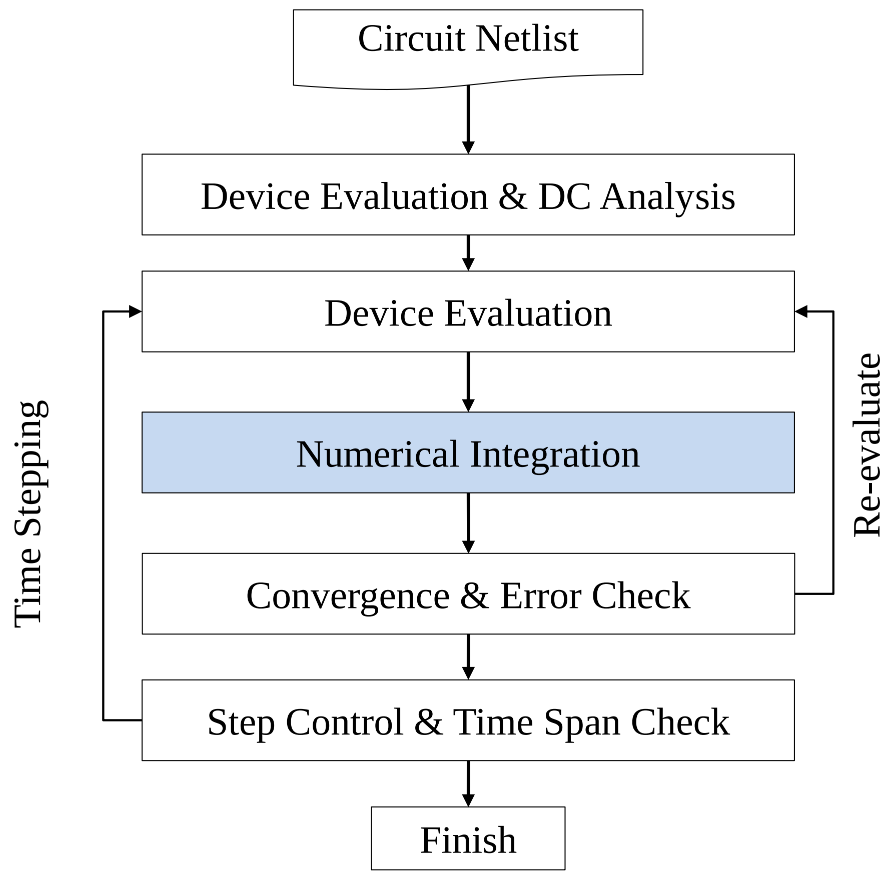

Circuit Netlist
Device Evaluation & DC Analysis
Device Evaluation
Numerical Integration
Convergence & Error Check
Step Control & Time Span Check
Finish
Time Stepping
Re-evaluate

Figure 1.1: SPICE-like time-domain simulation flow.

In circuit simulation society, researchers and engineers often resort to implicit linear multi-step methods, such as Backward Euler, Trapezoidal, and Gear's schemes [3]. Implicit methods have much better stability over their explicit counterparts. However, the size of a circuit could be up to billions that makes solving a linearized matrix system extremely challenging. For linear multi-step methods, four points are quite important to keep in mind:

(1) Linear multi-step is formulated according to the model of Taylor expansion of

differential equation systems.

(2) For the implicit linear multi-step method, the time step length h is restricted by accuracy requirements. The local truncation error is in proportion to the time step h^p ($p \geq 2$), where p is the order of the error term [3, 5, 7, 8]. Therefore, the longer the time step takes, the larger the error.

(3) Implicit formulation contains a linear system, of which the matrix includes the conductance/resistance matrix G and the capacitance/inductance matrix C with time step h. The combination is fixed for each matrix factorization during the process of solving the linear system. For the case where C and G are not sparse, the linear combination of C and G in implicit methods are even more complicated.

(4) Linear multi-step method is bounded by the Dahlquist barrier, i.e., the most accurate A-stable approach cannot exceed an order of 2. Therefore, the linear multi-step integration method is called the low order approach in this paper.

Summing up, those characteristics from traditional time integration kernels pose limitation with the problem scaling up.

1.3 Scope of the Thesis

In this paper, we focus on the exponential time integration, which is a high-order integration method and jumps beyond the conventional low-order linear multi-step approaches. The exponential time integration approach offers a variety of convergence integration formations that break the limitations of the Dahlquist stability barrier [7]. In 1978, Moler and Van Loan [15] classified the matrix exponential solvers into 19 approaches and claimed that the problem remains open. To best of our knowledge, Saad was the first to provide theoretical foundation to solve the matrix exponential with the

Krylov subspace approach [16], which was later termed the 20-th way in contrast to previous 19 dubious ways by Moler and Van Loan [15]. Since then, many related works have been published in this field of applied mathematics [17–19]. In this study, we want to investigate the related techniques in circuit simulation. The scope of this thesis is summarized in the following items:

- Analyzing the advantages of exponential time integration over traditional numerical integration approaches, such as Forward Euler, Backward Euler, and Trapezoidal method.
- Developing Krylov subspace methods for matrix exponential and vector computation for large-scale and stiff circuits.
- Presenting formulation of solving differential equations from circuit systems via exponential integration. Investigating the performance of different integration formulations.
- Integrating proposed exponential integrators into the linear circuit simulation framework and applying the approaches for large-scale VLSI power network analysis.
- Proposing an explicit integration framework for transient analysis of nonlinear dynamical systems. Carrying out performance test for proposed exponential time integration algorithm in large-scale nonlinear circuits.

1.4 Thesis Organization

We start in Chapter 2 by introducing the dynamical systems from circuits and the traditional numerical integration methods for solving the corresponding differential equation systems.

Chapter 3 presents the exponential time integration formulation and demonstrates its advantages over traditional low-order integration approaches through linear systems. In order to accelerate the matrix exponential computation, Krylov subspace methods with different preconditioned Arnoldi algorithms are exploited and compared.

In Chapter 4 we design algorithms with exponential time integration and Krylov subspace methods for simulating of linear circuits. Besides, we apply the techniques to the analysis of VLSI power network, which is a demanding task during modern VLSI signoff. In this chapter, we also leverage the adaptive time control and distributed computation framework to further accelerate the runtime of whole power network simulation process. At the end of chapter, we verify our results via IBM power grid benchmarks and achieve substantial speedups.

In Chapter 5, we focus on nonlinear dynamical systems. We devise a novel framework with the explicit integration scheme for general nonlinear dynamical systems. Thanks to the explicit formulation, we replace the traditional Newton-Raphson method with our proposed residue checking and compensation iteration process, where we follow Kirchoff's Current Law (KCL) and Voltage Law (KVL) to achieve converged solutions. Therefore, this framework reduces the number of computationally expensive matrix factorizations required by traditional integration approaches.

Finally, in Chapter 6, we summarize our contributions and list some possible directions for future work.

Chapter 2

Mathematical Background

This chapter briefly introduces the problem formulation and numerical integration algorithms used in the transient analysis of circuit simulation.

2.1 Differential Equations and Linear Multi-Step Numerical Integration

In order to transfer a circuit to a simulation program (SPICE), one must specify the circuit topology and the element constitutive equations. The circuit topology represents how the circuit elements are connected. The element constitutive equations defines the relations among node voltages and branch currents. Circuit differential equations are enforced by conservation laws, which are usually referred to as the Kirchhoff's current law (KCL) and voltage law (KVL). The circuit components, such as linear resistors, capacitors and inductors, as well as nonlinear devices (MOSFETs), are modeled and stamped into a matrix system via modified nodal analysis (MNA) [20]. The fundamental circuit simulation theory starts from differential equations as follows.

Given a differential equation system

$$\frac{dx}{dt} = f(x,t),$$

we want to compute the approximate solution $x(t)$ on an internal $a \leq t \leq b$ by numerical integration method, i.e., linear multi-step method.

Linear multi-step (k-step) method: the integration has the form [3],

$$\sum_{j=0}^{k} \alpha_j x(t_{i+j}) = h \sum_{j=0}^{k} \beta_j f(t_{i+j}, x(t_{i+j})), \ \ \alpha_k = 1, \ i \geq 0,$$

where time $t_j = a + jh$, $0 \leq j \leq \frac{b-a}{h}$. The method is explicit when $\beta_k = 0$, otherwise it is implicit. When $k = 1$, the integration is also called a linear one-step method.

2.2 Dynamical System in Circuit Simulation

The general formulation of circuit simulation is described as follows,

$$\frac{dq(x(t))}{dt} + f(x(t)) = Bu(t), \tag{2.1}$$

where vector $x(t) \in \mathbb{R}^{n\times 1}$ denotes nodal voltages and branch currents and n is the length of vector. Vector $q \in \mathbb{R}^{n\times 1}$ and function $f \in \mathbb{R}^{n\times 1}$ represent the charge/flux and current/voltage terms, respectively. The derivate $\frac{dq}{dt}$ represents the energy storage elements, such as capacitors or inductors, which have time-dependent effects. Vector $u(t)$ represents all the external excitations at time t; Matrix B is an incident matrix that inserts those signals to the system. If the element constitutive equations are linearized, Eq. (2.1) can be reduced in matrix form as

$$C\frac{dx}{dt} + Gx = Bu(t) + F(x), \tag{2.2}$$

where $F(x)$ is the nonlinear dynamics evaluated by device model, which puts into the right hand side; $C \in \mathbb{R}^{n\times n}$ results from capacitive and inductive elements (capacitance/inductance matrix). Matrix $G \in \mathbb{R}^{n\times n}$ is the conductance/resistance matrix. The entries are given by

$$C_{i,j} = \frac{\partial q_i}{\partial x_j},$$

and

$$G_{i,j} = \frac{\partial f_i}{\partial x_j},$$

where q_i and f_i represents i-th equation in the system of q and f;

$$C = \begin{bmatrix} Q & 0 \\ 0 & H \end{bmatrix}, \; G = \begin{bmatrix} M & E \\ -E^T & R \end{bmatrix}, \; x = \begin{bmatrix} x_v \\ x_i \end{bmatrix}, \; u = \begin{bmatrix} u_i \\ u_v \end{bmatrix},$$

where $Q, M \in \mathbb{R}^{c\times c}$ represent capacitance and conductance, respectively. Matrix E is the incident matrix. Vector x_v is the node voltage vector; x_i represents the branch current; u_i is the current input; u_v is the voltage input. Scalar c is the number of nodes. Therefore, the first c equations represent the connections of nodes and are enforced by KCL. Matrices $H, R \in \mathbb{R}^{l\times l}$ represent inductance and resistance, respectively. Vector x_i is the current vector. Scalar l is the number of branches. The next l equations are governed by KVL. The whole system dimension is $n = c + l$.

2.3 Conventional Numerical Integration Approaches

Starting from a linear differential system Eq. (2.3) as

$$C\frac{dx}{dt} = -Gx + Bu(t), \tag{2.3}$$

and the initial vector $x(t)$ at time t, we compute the solution $x(t+h)$ with time step h.

2.3.1 Forward Euler Time Integration (FE)

Forward Euler time integration scheme starts with the approximation

$$x(t+h) = x(t) + h\frac{dx}{dt}\Big|_{x=x(t)},$$

which leads to

$$\frac{C}{h}x(t+h) = \left(\frac{C}{h} - G\right)x(t) + Bu(t) \tag{2.4}$$

in the circuit simulation formulation.

2.3.2 Backward Euler Time Integration (BE)

Backward Euler time integration scheme starts with

$$x(t+h) = x(t) + h\frac{dx}{dt}\Big|_{x=x(t+h)}.$$

Then,

$$\left(\frac{C}{h} + G\right)x(t+h) = \frac{C}{h}x(t) + Bu(t+h). \tag{2.5}$$

2.3.3 Trapezoidal Time Integration (TR)

Trapezoidal time integration scheme starts with

$$x(t+h) = x(t) + \frac{h}{2}\left(\left.\frac{dx}{dt}\right|_{x=x(t)} + \left.\frac{dx}{dt}\right|_{x=x(t+h)} \right).$$

We have

$$\left(\frac{C}{h}+\frac{G}{2}\right)x(t+h) \quad = \quad \left(\frac{C}{h}-\frac{G}{2}\right)x(t)+B\frac{u(t)+u(t+h)}{2}. \tag{2.6}$$

Methods FE, BE, and TR all belong to linear multi-step method, also known as the linear one-step method. A-stable linear multi-step methods are favored in circuit simulation to solve time integration problems, since the numerical error is only caused by local truncation error and would not be amplified by the instability of numerical integration itself.

Definition 2.3.1 (A-stability)**.** A linear multi-step method is said to be *A-stable* if its region of absolute stability includes the whole left half-plane[1].

The stability regions of FE, BE and TR are shown in Fig. 2.1. Method FE has a very limited stability region, while BE covers the largest region in the complex plane. Time step h in FE is constrained by $\min(|\lambda_i|^{-1})$ (λ_i: an eigenvalue of matrix A). Electronic circuits have eigenvalue magnitudes spanning at least several decades, which leads to impractically tiny time step h for simulation using FE. Circuit systems with a wide range of eigenvalues are said to be stiff [21]. BE and TR are all A-stable and served as baseline methods in this paper. We keep the other linear multi-step schemes out of this

[1]Another equivalent way to interpretation of A-stable: The numerical integration method is A-stable. For the linear system $dx/dt = Ax$ with time step h, the solution $x(t+h)$ obtained by the numerical integration approaches 0, or $x(t+h) \to 0$ when $h \to \infty$ and the real parts of all eigenvalues of A are negative.

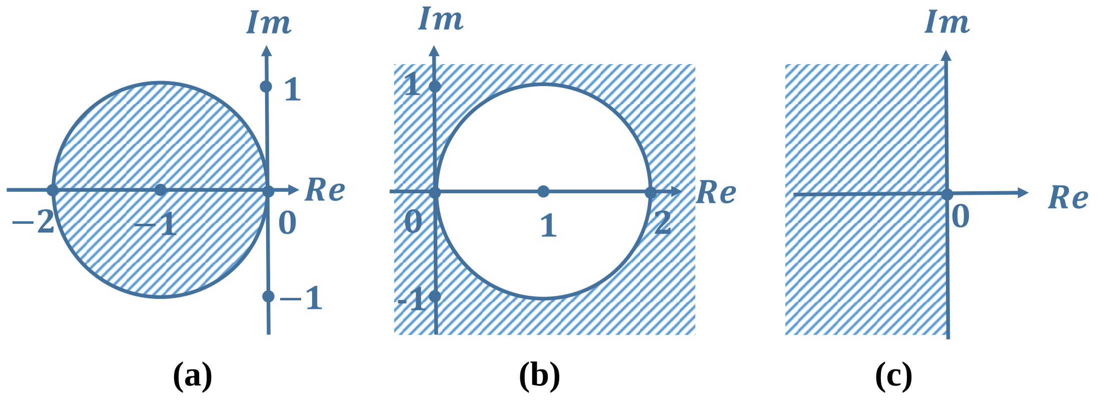


Figure 2.1: Stability regions (shaded) of (a) Forward Euler (FE), (b) Backward Euler (BE), and (c) Trapezoidal methods in the complex plane.

paper, since the numerical integration in SPICE-like tools usually use linear multi-step methods so that they cannot exceed the second Dahlquist barrier.

Theorem 2.3.1 (the second Dahlquist barrier)**.** *There are no explicit A-stable and linear multi-step methods. The implicit ones have order of convergence at most* 2*. The trapezoidal rule has the smallest error constant amongst the A-stable linear multistep methods of order* 2 *[7][22].*

Interested readers can refer to [3, 7, 8] for more details of numerical stability in circuit simulation.

2.4 Nonlinear Dynamical Systems and Newton-Raphson Method

For example, BE is used to approximate the system first,

$$\frac{q(x) - q(x(t))}{h} + f(x) = Bu(t+h). \tag{2.7}$$

For the nonlinear system, $q(x)$ and $f(x)$ are nonlinear functions of vector x. Newton-Raphson method (NR) is often used to obtain a converged solution x of

$$F(x) = \frac{q(x) - q(x(t))}{h} + f(x) - Bu(t+h) = 0. \tag{2.8}$$

NR method can be derived by examining the first terms in a Taylor series expansion around a guess solution x

$$0 = F(x^*) \approx F(x) + J(x)(x^* - x_i), \tag{2.9}$$

where x^* is the exact solution to Eq. (2.8). Matrix $J(x)$ is the $n \times n$ Jacobian matrix whose elements are given by

$$J_{i,j}(x) = \frac{\partial F_i(x)}{\partial x_j}, \tag{2.10}$$

where F_i represents the i-th equation in the system of F.

Given $x^{(i)}$ the i-th iteration process to refine the solution. Each NR iteration, direct solver (e.g., LU decomposition) is applied to solve Eq. (2.11) until the series of $\{x, x^{(1)}, \cdots, x^{(i)}, x^{(i+1)}\}$ are converged, which means the difference of the solution from i-th iteration $x^{(i)}$ and $x^{(i+1)}$ is "small enough".

$$J(x^{(i)})(x^{(i+1)} - x^{(i)}) = -F(x^{(i)}) \tag{2.11}$$

where $x^{(i+1)}$ is the "improved" estimation of x^*. If $F(x)$ and $J(x)$ are "well-behaved" matrices, NR will converge quadratically given a good initial guess solution. The errors generated by NR satisfy the condition

$$\|x^{(*)} - x^{(i+1)}\| \leq \kappa \|x^{(*)} - x^{(i)}\|^2,$$

where κ is proportional to bounds on $\|J(x^{(i)})^{-1}\|$ and the ratio of $\|F(x) - F(z)\|/\|x - z\|$ [23]. The above process is called *BENR* in this thesis, which is used in Chapter 5 as the baseline for SPICE-like nonlinear circuit simulation.

In practical circuit simulation, two challenges are likely to encounter. First, matrix solving processes are required because of implicit scheme, time step h is embedded in $J(x)$ of Eq. (2.10). For example,

$$J(x) = \frac{\partial F(x)}{\partial x} = \frac{C(x)}{h} + G(x).$$

If the estimated local truncation error (LTE) [3] violates numerical error budget, h should be reduced. Then new NR iterations for x^* are re-launched with the updated h.

Second, matrix system is hard to solve. A post-layout extraction can expand a netlist 5-10 times larger. Huge volume of non-zeros of C are introduced to describe the parasitic effects after extraction [2, 24–26], resulting in huge computational challenges for the capability of numerical integration algorithms [9] and model order reductions [27]. In addition, the off-diagonal terms in C and G are usually mutually exclusive in VLSI circuits, which might bring the huge number of non-zero fill-ins after matrix factorization [28].

2.5 Direct Method (LU Decomposition)

In circuit simulation, solving a linear system

$$Ax = b$$

is a key component. LU decomposition method is a stable approach and widely adopted. It is used in the Newton-Raphson iterations, and Arnoldi algorithms for Krylov subspace in Chapters 3, 4, and 5. First, matrix A is factorized to a lower triangular matrix L and an upper triangular matrix U, which is

$$A = LU.$$

Then, the system can be solved by forward and backward substitutions, which is expressed as[2]

$$x = U\backslash(L\backslash b)$$

The complexity is $O(n^3)$ for dense matrix and $O(n^{1.5})$ for sparse matrix. In order to reduce the number of non-zero fill-ins generated by factorization, matrices are usually to be reordered based on the structure [29]. However, it is still a hard problem, and excites many researchers [28, 30–33]. In this study, we use LU decomposition as the direct matrix solver to solve linear system.

[2]Follow MATLAB's syntax.

2.6 Summary of Conventional Approaches

- The methods listed in this section are all low order approximation of the exact solution of differential equation system. Local truncation error also limits the time step size in widely used implicit methods.

- Implicit method is preferred in the circuit simulation for its stability property. We need to solve linear systems as Eq. (2.5) and Eq. (2.6).

- Due to the implicit integration scheme, Eq. (2.10) need to embed time step h in T and the Jacobian matrix J. If SPICE-like local truncation estimation [3, 34] violates numerical error budget, h will be reduced. Then new NR iterations for finding $x(t+h)$ will be re-launched with the updated h. When matrix J is large and complicated, the matrix solver will cost huge runtime.

- Direct matrix solver is more widely used over iterative solver because of its robustness. Therefore, matrix factorization (LU decomposition) is required when the linear system changes.

- Due to the large amount and complicated distributions of non-zeros in C, the post-layout or strong coupled system sometimes adds huge computational complexity, which may extend beyond existing hardware and software capacity. This rational is also applied to other low order linear approximation integration kernels, such as TR, Gear's methods.

Chapter 3

Exponential Integration, Matrix Exponentials, and Krylov Subspace Methods for Computing the Product of Matrix Exponential and Vector

In this chapter, we briefly introduce the formulation of exponential integration in the circuit simulation. We also discuss matrix exponentials and Krylov subspace methods in order to compute the matrix exponential and vector product (MEVP). In addition, we state the connections between exponential integrators and conventional approaches in Chapter 2. We illustrate the accuracy advantage of exponential integrators using simple RC and RLC circuits.

3.1 Circuit Simulation via Exponential Integration

We follow the analytical solution with matrix exponentials for circuit simulation by Chua and Lin [7]. We apply the chain rule to Eq. (2.1),

$$\frac{dq(x(t))}{dx} \cdot \frac{dx(t)}{dt} = Bu(t) - f(x(t)). \tag{3.1}$$

Assume $C(x(t))$ is invertible[1].

$$\begin{aligned} \frac{dx(t)}{dt} &= g(x(t),u,t) = C^{-1}(x(t))(Bu(t) - f(x(t))) \\ &= Ax(t) + C^{-1}(x(t))\left(N(x(t)) + Bu(t)\right), \end{aligned} \tag{3.2}$$

where

$$f(x(t)) = G(x(t))x(t) - N(x(t))$$

and $N(x(t))$ is a nonlinear function of $x(t)$. Matrix A denotes the Jacobian matrix of $g(x(t),u,t)$ at $x(t)$ [17, 19, 35],

$$A = -C^{-1}G,$$

where matrices G and C are short for matrices $G(x(t))$ and $C(x(t))$, which are evaluated at $x(t)$.

We use Exponential Rosenbrock-Euler method [17] to compute $x(t+h)$ with step size h as follows,

$$x(t+h) = x(t) + \frac{e^{hA} - I}{A} \cdot g(x(t),u,t) + \frac{e^{hA} - hA - I}{A^2} \cdot \frac{\partial g(x(t),u,t)}{\partial t}, \tag{3.3}$$

If we only consider linear system with piecewise-linear input $u(t)$ from $[t,\ t+h]$

[1]The assumption is to simplify the explanation in this section. After Sec. 3.2.2, we use invert and rational Krylov subspace methods to compute the solution of DAE without inversion of C. Therefore, the methods are suitable for general DAE system, i.e., Eq. (4.1) without the assumption here.

[36–38]. We have

$$g(x(t),u,t) = Ax(t) + C^{-1}Bu(t), \tag{3.4}$$

and

$$\frac{\partial g(x(t),u,t)}{\partial t} = C^{-1}B\frac{u(t+h)-u(t)}{h}. \tag{3.5}$$

Then, the formulation in Eq. (3.3) is simplified to Eq. (4.6).

$$\begin{aligned} x(t+h) = &-\left(A^{-1}b(t+h) + A^{-2}\frac{b(t+h)-b(t)}{h}\right) + \\ &e^{hA}\left(x(t) + A^{-1}b(t) + A^{-2}\frac{b(t+h)-b(t)}{h}\right), \end{aligned} \tag{3.6}$$

where $b(t) = C^{-1}Bu(t)$. Note that Eq. (4.6) is the exact solution of the linear dynamical system under our given constraints.

To best of our knowledge, all of the numerical integration methods in SPICE-like simulators are from linear multi-step scheme, which try to approximate this solution via matrix exponential operators [7] in a low order way. To discuss the approximation schemes in last section, we treat $u(t) = 0$ for simplicity, and show the source of accuracy loss. We have the simplified homogeneous system of Eq. (4.6),

$$\frac{dx}{dt} = Ax. \tag{3.7}$$

The solution is

$$\begin{aligned} x(t+h) &= e^{hA}x(t) \qquad (3.8)\\ &= \sum_{k=0}^{\infty} \frac{h^k A^k}{k!} x(t)\\ &= x(t) + hAx(t) + \frac{h^2A^2}{2}x(t) + \frac{h^3A^3}{3!}x(t) + \cdots + \frac{h^kA^k}{k!}x(t) + \cdots . \end{aligned}$$

Method FE formulation

$$x(t+h) = \left(\frac{C}{h}\right)^{-1}\left(\frac{C}{h} - G\right)x(t) = (I + hA)x(t) \qquad (3.9)$$

fits the first two terms of Eq. (3.8). Therefore, the accuracy order of FE is $O(h)$.

Method BE formulation

$$x(t+h) = \left(\frac{C}{h} + G\right)^{-1}\frac{C}{h}x(t) = (I - hA)^{-1}x(t) \qquad (3.10)$$

also matches the first two terms by

$$(I - hA)^{-1} = \sum_{k=0}^{\infty} h^k A^k. \qquad (3.11)$$

The accuracy order of BE is also $O(h)$.

Method TR formulation

$$\begin{aligned} x(t+h) &= \left(\frac{C}{h} + \frac{G}{2}\right)^{-1}\left(\frac{C}{h} - \frac{G}{2}\right)x(t)\\ &= \left(I - \frac{hA}{2}\right)^{-1}\left(I + \frac{hA}{2}\right)x(t) \qquad (3.12) \end{aligned}$$

fits the first three terms.

$$\left(I-\frac{hA}{2}\right)^{-1}\left(I+\frac{hA}{2}\right)=\left(I+hA+\frac{h^2A^2}{2}+\frac{h^3A^3}{4}+\cdots\right). \tag{3.13}$$

The accuracy order of TR is $O(h^2)$.

Note that series of Eq. (3.11) and Eq. (3.13) only converge for hA of BE and $\frac{hA}{2}$ of TR with spectral radius less than one. Besides, the mismatch terms of Eq. (3.9), Eq. (3.10), and Eq. (3.12) against Eq. (3.8) introduce the local truncation error (LTE) to FE, BE, and TR, respectively, which constrain the time step with respect to the region of Taylor expansion.

Fig. 3.1 shows a test equation

$$\frac{dx}{dt}=-x(t)$$

solved by method exponential integration EXPM

$$x(h)=e^{-h}x(0),$$

analytically, as well as FE, BE, and TR with different time step h. The figure illustrates that mismatched results of FE, BE, and TR compared to EXPM with different time step h.

In other words, if e^{hA} is used to compute the solution of differential equation system directly, there is no local truncation error constraint for the time step choice. However, the question is how matrix exponential and vector product (MEVP) can be computed in an efficient way, since the size of A in $e^{hA}x(t)$ is usually above million, making the direct computation unfeasible. In addition, Fig. 3.2 describes a “hump” effect

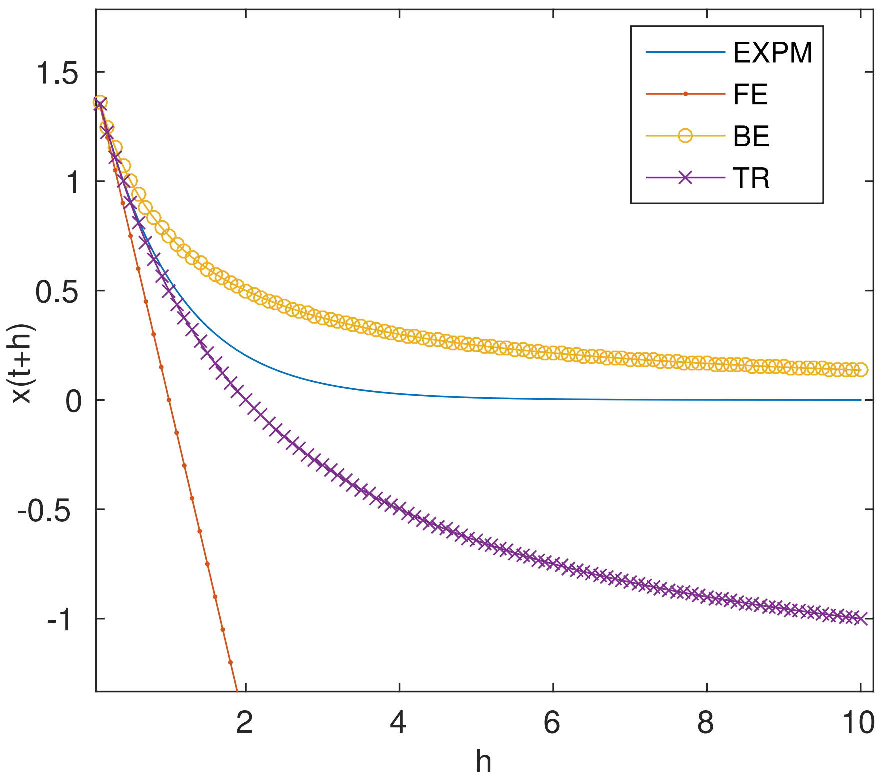


Figure 3.1: A test equation $\frac{dx}{dt} = -x(t)$, where $x(0) = 1.5$, $h \in [0, 10]$. Analytical solution is computed by EXPM $x(h) = e^{-h}x(0)$.

during the computation of e^A [1]. Term $A^k/k!$ of series

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}$$

may increase before the value can drop after $k > max|\lambda(A)|$. Therefore, we need high order k to converge the series, which makes MEVP computation even more challenging.

3.2 MEVP and Krylov Subspace Methods

One efficient way among different approaches is to compute MEVP through Krylov subspace method [1, 16]. The complexity of $e^A v$ can be reduced using Krylov

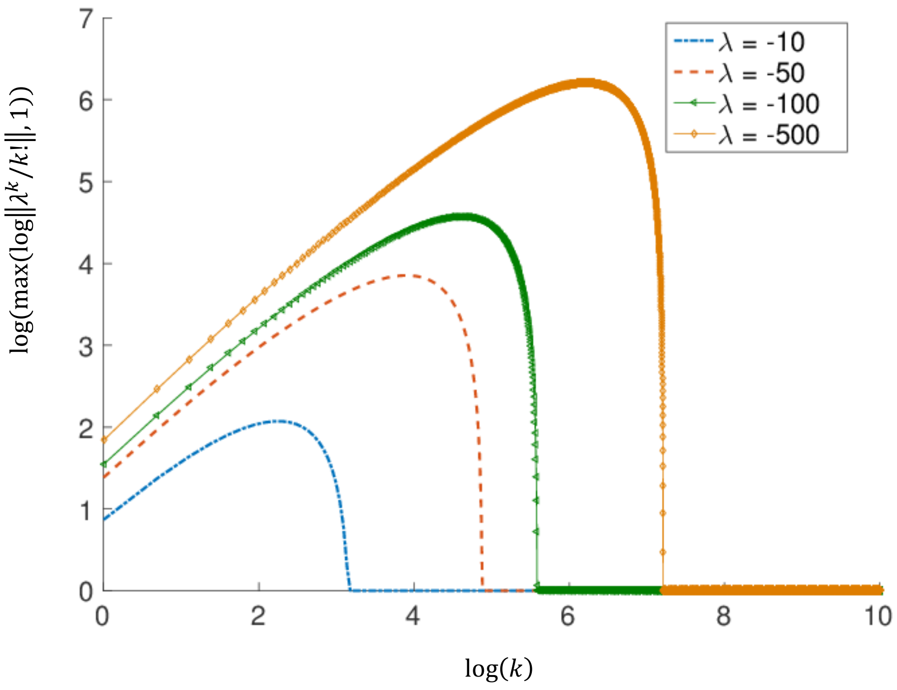


Figure 3.2: The “hump” effect mentioned in [1].

subspace method and still maintained in a high order polynomial approximation [16]. In this section, we first introduce the background of Krylov subspace for MEVP. Then, we discuss so-called standard (Std) [14], invert (Inv) [39] and rational (Rat) Krylov subspace methods [37, 38], which highly improve the runtime performance for MEVP in circuit simulation.

Definition 3.2.1 (Krylov Subspace)**.** Given a matrix A and a vector v, the Krylov subspace of order m, denoted by $K_m(A,v)$, is defined as the subspace spanned by the vectors $v, Av, \cdots, .A^{m-1}v$, or

$$K_m(A,v) := \text{span}\{v, Av, \cdots, A^{m-1}v\}. \tag{3.14}$$

It is convenient to work with an orthonormal basis for $K_m := K_m(A,v)$. Let $\{v_i\}_{i=0}^{m-1}$ be an orthonormal basis for K_m. Let V_m be the $n \times m$ matrix with $\{v_i\}_{i=0}^{m-1}$ as its columns. $V_m V_m^\top$ is the projection onto K_m. Let H_m be the $m \times m$ Hessenberg matrix expressing A as an operator restricted to K_m in the basis $\{v_i\}_{i=0}^{m-1}$, i.e.,

$$H_m = V_m^\top A V_m.$$

We have v, $Av \in K_m$, then

$$\begin{aligned} Av &= (V_m V_m^\top) A (V_m V_m^\top) v \\ &= V_m (V_m^\top A V_m) V_m^\top v \\ &= V_m H_m V_m^T v. \end{aligned} \tag{3.15}$$

Similarly, for all $i \leq m-1$,

$$A^i v = V_m H_m^i V_m^\top v,$$

we have $p(A)v = V_m p(H_m) V_m^T v$, for any polynomial p of degree at most $m-1$ [16].

Lemma 3.2.1 (Exact Computation with Polynomials. See e.g., [16,40])**.** *Let V_m and H_m be as defined above. For any polynomial p of degree at most $m-1$,*

$$p(A)v = V_m p(H_m) V_m^T v. \tag{3.16}$$

Thus, H_m can be used to compute matrix function and vector product $p(A)v$ for any degree $m-1$ polynomial p. This lemma suggests that a candidate for computing $f(A)v$ approximately is via $V_m f(H_m) V_m^\top v$. The metric to evaluate the result is the norm of error, such as $\|f(A)v - V_m f(H_m) V_m^\top v\|$ [40]. Define $r_{m-1}(x) = f(x) - p_{m-1}(x)$, where

p_{m-1} is any degree $m-1$ approximation to $f(x)$, and using Lemma 3.2.1. Then

$$f(A)v - V_m f(H_m) V_m^\top = r_{m-1}(A)v - V_m r_{m-1}(H_m) V_m^\top v.$$

Therefore, the norm of the error vector is at most $(\|r_{m-1}(A)\| - \|r_{m-1}(H_m)\|)\|v\|$, which is bounded by the value of r_{m-1} on the eigenvalues of A and H_m [40].

Lemma 3.2.2 (Approximation by Best Polynomial. See e.g., [16,40])**.** *Let V_m and H_m be as defined above. Let $f : \mathbb{R} \to \mathbb{R}$ be any function such that $f(A)$ and $f(H_m)$ are well-defined. Then,*

$$\begin{aligned} & \|f(A)v - V_m f(H_m) V_m^\top v\| \qquad (3.17) \\ \leq & \min_{p_{m-1} \in \Sigma_{m-1}} \big(\max_{\lambda \in \Lambda(A)} |f(\lambda) - p_{m-1}(\lambda)| \\ + & \max_{\lambda \in \Lambda(H_m)} |f(\lambda) - p_{m-1}(\lambda)| \big). \end{aligned}$$

Hence, $V_m f(H_m) V_m^\top v$ approximates $f(A)v$ as well as the best degree $m-1$ polynomial that uniformly approximates f. The question that remains is how to compute H_m and V_m for $f(A)v$.

3.2.1 MEVP via Standard Krylov Subspace Method (Std)

Arnoldi algorithm (Algorithm 1) is used to construct standard Krylov subspace Eq. (3.14) [14,16]. The steps from line 4 to 7 of Algorithm 1 form a modified Gram-Schmidt process. The process above produces an orthonormal basis $\{v_i\}_{i=1}^m$ of the Krylov subspace K_m. If we denote the $m \times m$ upper Hessenberg matrix H_m consisting of the $h_{i,j}$

Algorithm 1: Arnoldi Algorithm

```
1  $v_1 = v/\|v\|$;
2  for $j = 1 : m$ do
3      $w = Av_j$;
4      for $i = 1 : j$ do
5          $h_{i,j} = w^\top v_i$;
6          $w = w - h_{i,j} v_i$;
7      end
8      $h_{j+1,j} = \|w\|$;
9      $v_{j+1} = \frac{w}{h_{j+1,j}}$;
10 end
```

from the algorithm, we have the equation.

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^\top, \tag{3.18}$$

where V_m is a $n \times m$ matrix, and e_m is the m-th unit vector with dimension $m \times 1$. Then, MEVP $f(A)v = e^A v$ is computed via

$$e^A v \approx \beta V_m e^{H_m} e_1. \tag{3.19}$$

Besides, since $V_m^\top (hA) V_m = hH_m$ and Krylov subspaces associated with A and hA are identical, we have

$$e^{hA} v \approx \beta V_m e^{hH_m} e_1. \tag{3.20}$$

Note that Eq. (3.20) distinguishes approximation method from linear multi-step methods, which uses non-linear coefficients generated by e^{H_m}. *Therefore, the matrix exponential methods break away from linear multi-step methods and thus are not limited by the Dahlquist barrier.*

The posterior residue-based error term is

$$\|\beta h_{m+1,m} v_{m+1} e_m^\top e^{hH_m} e_1\|, \tag{3.21}$$

where $\beta = \|v\|$ [41] . However, in circuit theory, we actually need to only consider the residual between $C\frac{dx}{dt}$ and $-Gx$, which is

$$C\frac{dx}{dt} + Gx,$$

instead of $\frac{dx}{dt} - Ax$. This leads to the residual (error) approximation

$$r(m,h) = \|\beta h_{m+1,m} C v_{m+1} e_m^\top e^{hH_m} e_1\| \tag{3.22}$$

for our circuit simulation problem.

For the accuracy of approximation of $e^A v$, large dimension of Krylov subspace basis is required, which not only increases the computational complexity but also consumes huge amount of memory. The reason is that the Hessenberg matrix H_m and subspace of standard Krylov subspace method tend to approximate the large magnitude eigenvalues and corresponding eigenvectors of A [42]. Due to the exponential decay of higher order terms in Taylor expansion, such components are not the crux of circuit system's behavior [41,42]. Dealing with stiff circuits, therefore, needs to gather more vectors into subspace basis and increases the size of H_m to fetch more useful components, which results in both memory overhead and computational complexity into Krylov subspace generations during time stepping.

To improve the efficiency, we adopt the idea from *spectral transformation* [41,42] to effectively capture small magnitude eigenvalues and corresponding eigenvectors in A, leading to a fast yet accurate MEVP computation.

3.2.2 MEVP via Invert Krylov Subspace Method (Inv)

Instead of A, we use A^{-1} as our target matrix to form

$$K_m(A^{-1},v) := \text{span}\{v, A^{-1}v, \cdots, A^{-(m-1)}v\}. \tag{3.23}$$

Intuitively, by inverting A, the small magnitude eigenvalues become the large ones of A^{-1}. The resulting H_m is likely to capture these eigenvalues first. Based on Arnoldi algorithm, the invert Krylov subspace has the relation of matrices

$$A^{-1}V_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^\top. \tag{3.24}$$

The matrix exponential $e^A v$ is calculated as

$$e^A v \approx \beta V_m e^{hH_m^{-1}} e_1. \tag{3.25}$$

The residual (error) approximation [37] is

$$r(m,h) = \|\beta h_{m+1,m} G v_{m+1} e_m^\top H_m^{-1} e^{hH_m^{-1}} e_1\|. \tag{3.26}$$

3.2.3 MEVP via Rational Krylov Subspace Method (Rat)

The shift-and-invert Krylov subspace basis [42] is designed to confine the spectrum of A. Then, we generate Krylov subspace via

$$K_m((I-\gamma A)^{-1},v) := \text{span}\{v, (I-\gamma A)^{-1}v, \cdots, (I-\gamma A)^{-(m-1)}v\}, \tag{3.27}$$

where γ is a predefined parameter. With this shift, all the eigenvalues' magnitudes are larger than one. Then the invert limits the magnitudes smaller than one. According to

[41,42], the shift-and-invert basis for matrix exponential based transient simulation is not very sensitive to γ, once it is set to around the order near time steps used in transient simulation. The similar idea has been applied to simple power grid simulation with matrix exponential method [36–38]. Arnoldi process constructs V_m, H_m with the relationship

$$(I-\gamma A)^{-1}V_m = V_m H_m + h_{m+1,m}v_{m+1}e_m^{\top}. \tag{3.28}$$

We can project the e^A onto the rational Krylov subspace as follows.

$$e^{Ah}v \approx \beta V_m e^{h\frac{I-H_m^{-1}}{\gamma}} e_1. \tag{3.29}$$

The residual (error) approximation is derived as

$$r(m,h) = \|\beta h_{m+1,m}\frac{C+\gamma G}{\gamma}v_{m+1}e_m^{\top}H_m^{-1}e^{h\frac{I-H_m^{-1}}{\gamma}}e_1\|. \tag{3.30}$$

3.2.4 Algorithm for the Approximation of $e^{hA}v$

Algorithm 2 is listed with ε being the error budget constraint. First, we explain the inputs for different Krylov subspace basis choices,

- Std basis: $X_1 = C,\ X_2 = -G,\ H = H_m$.
- Inv basis: $X_1 = G,\ X_2 = -C,\ H = H_m^{-1}$.
- Rat basis: $X_1 = C+\gamma G,\ X_2 = C,\ H = \frac{I-H_m^{-1}}{\gamma}$.

Direct matrix solver (LU_Decompose) is applied before starting Algorithm 2.

$$[L,\ U] = LU\_Decompose(X_1). \tag{3.31}$$

Algorithm 2: Algorithm for $e^{hA}v$.

```
Input: $v, L, U, X_2, h, t, \varepsilon$, where $LU = X_1$
Output: $u$
1  $v_1 = v/\|v\|$;
2  for $j = 1 : m$ do
3      $w = U\backslash(L\backslash(X_2 v_j))$;
4      for $i = 1 : j$ do
5          $h_{i,j} = w^T v_i$;
6          $w = w - h_{i,j} v_i$;
7      end
8      $h_{j+1,j} = \|w\|$;
9      $v_{j+1} = w/h_{j+1,j}$;
10     if $r(j,h) < \varepsilon$ then
11         $m = j$;
12         break;
13     end
14 end
15 $u = \|v\| V_m e^{hH} e_1$ ;
```

When dealing with singular C, Std needs the regularization process [43] to remove the singularity of DAE. The reason is that Std needs to factorize C in Algorithm 1. This brings extra computational overhead. Actually, it is not necessary if we can obtain the generalized eigenvalues and corresponding eigenvectors for matrix pencil $(-G, C)$. Based on [44], we derive Lemma 3.2.3,

Lemma 3.2.3. *Considering a homogeneous system*

$$C\frac{dx}{dt} = -Gx$$

u and λ are the eigenvector and eigenvalue of matrix pencil $(-G, C)$, then

$$x = e^{t\lambda} u$$

is a solution of the system.

Proof. (See e.g. [44])[2] If λ and u are an eigenvalue and eigenvector of a generalized eigenvalue problem

$$-Gu = \lambda Cu.$$

Then, $x = e^{t\lambda}u$ is the solution of $C\frac{dx}{dt} = -Gx$. $\square$

The process of Inv and Rat are regularization-free, because of no computation and factorization of C^{-1}. Instead, we factorize G for Inv, and $(C+\gamma G)$ for Rat. Besides, the invertible Hessenberg matrices contain corresponding important generalized eigenvalues/eigenvectors from matrix pencil $(-G,C)$, and define the behavior of linear dynamic system of interest.

3.3 Comparisons of Numerical Integration Approaches

In this section, we test numerical integration methods in order to illustrate the salient features of matrix exponential based approaches.

3.3.1 Overview of the RC and RLC Mesh Circuit Benchmarks

We create an RC mesh circuit with 1600 nodes (40×40). The entries of G are in the interval $[0.01, 100]$. The diagonal entries of C are set in the interval $[8.5 \times 10^{-18}, 9.9 \times 10^{-16}]$. The resultant matrix $A = -C^{-1}G$ contains eigenvalues in the interval $[-3.98 \times 10^{17}, -8.49 \times 10^{10}]$, which is plotted in Fig. 3.4 (a).

For RLC system, we use the RC mesh circuit in Section 3.3.2 and append a grounded inductor $L = 2fH$ to 160 nodes in the mesh. The spectrum of the circuit is plotted in Fig. 3.4 (b). We observe the conjugate pairs of complex eigenvalues due to the unsymmetrical matrix A with inductance.

[2]We repeat the proof from [44] with some modifications based on our formulation

In Section 3.3.2, we investigate the error distributions with a single time step using numerical integration methods Std, Inv, Rat, FE, BE, and TR. We set the initial vector $x(0)$ with a random vector with element in the interval $(0,1)$, whose L_2-norm is 23.3 and L_∞-norm is 0.999.

For Section 3.3.3, we investigate the error distributions during the whole time span of transient simulation using numerical integration methods Std, Inv, Rat, FE, BE, and TR. We use the RC circuit and inject an input PWL (0 0mA, 100ps 0mA, 110ps 10mA, 300ps 10mA, 310ps, 0mA)[3] at an ungrounded node (3rd row and 5th column). A fixed time step size $h = 1ps$ is used for all the numerical integration methods.

3.3.2 Error Distributions of the Numerical Integration Approaches with a Single Time Step

Simulation of RC Interconnect System

Fig. 3.5 shows the error vs. step h in log scale for the methods of FE, BE, TR and MEVP via **(a)** rational (Rat), **(b)** standard (Std) and invert (Inv) Krylov subspace methods. For Rat, we set $\gamma = h/2$ in Eq. (3.27). The metric is formulated as

$$||x(h) - e^{hA}x(0)||_\infty,$$

where we use $e^{hA}x(0)$ the exact analytical solution as reference, and $x(h)$ is the solution computed by numerical integration methods that starts from $x(0)$ the initial vector.

In Fig. 3.5(a), for the case $h \leq min(|\lambda_i|^{-1})$, Taylor expansion is valid for BE and TR. Thus, the method BE has the error slope follow the 2nd order term, TR the 3rd order term, while Rat has the error slope increase with Krylov subspace dimension m. For the case $h \geq max(|\lambda_i|^{-1})$, the solution attenuates globally. Thus, Rat error curves drop, but

[3]PWL input is written in the SPICE syntax

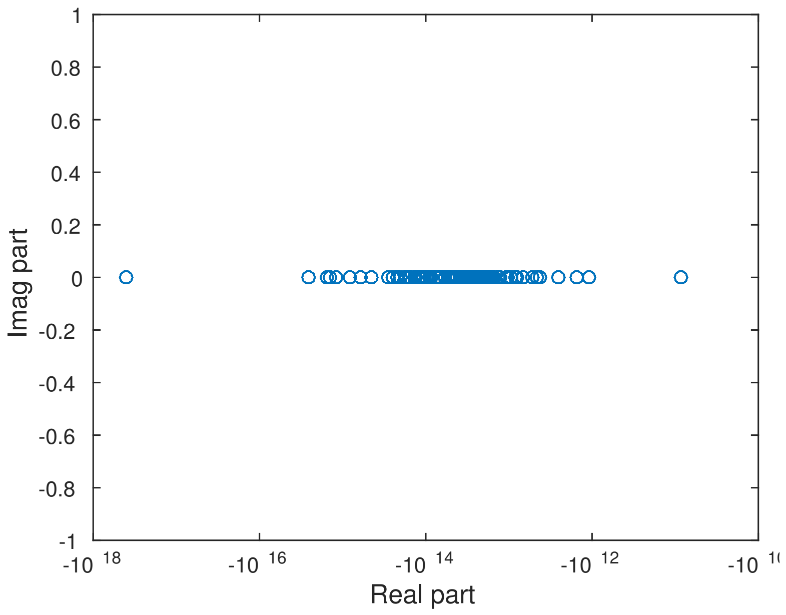


Figure 3.3: The spectrum of $-C^{-1}G$ of the RC sample case in Section 3.3.2.

BE and TR error curves remain flat. For the case that h is between the two bounds, most curves are flat. However, Rat improves as the dimension m increases. When $m = 2$,[4] Rat uses the same subspace as TR but achieves better accuracy. In other words, it is better off to replace TR with Rat for this circuit[5]. Note that we cannot further improve FE, BE and TR since their theoretical numerical orders have been already fixed. In Fig. 3.5(a), there are some abnormal curves in the small time step region is due to the numerical issues, when h is too small, and the matrix A disappear since $(I - \frac{h}{2}A)^{-1} \rightarrow I$. Fig. 3.6 plots the normal curves for Rat when $\gamma = 10^{-8}$.

Fig. 3.5(b) illustrates the error trends of Std, Inv with BE and TR. As the dimension m increases, Std error curves shift to the right and converge at the end on the right side; and Inv has its curves shift to the left and converge at the left side. For this circuit, we are interested in the behavior in the nano-second scale. At this time scale, Inv converges faster than Std as dimension m increases. This summary of error trend is listed

[4]The number m is referred to the line 2 of Algorithm 2

[5]In order to achieve $m = 2$, we need two times of matrix solving in Algorithm 2

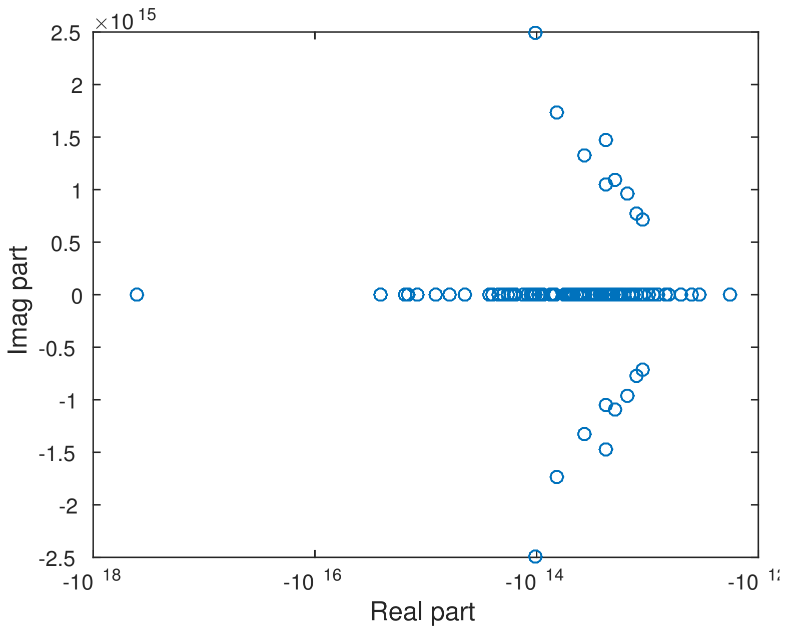


Figure 3.4: The spectrum of $-C^{-1}G$ of the RLC sample case in Section 3.3.2.

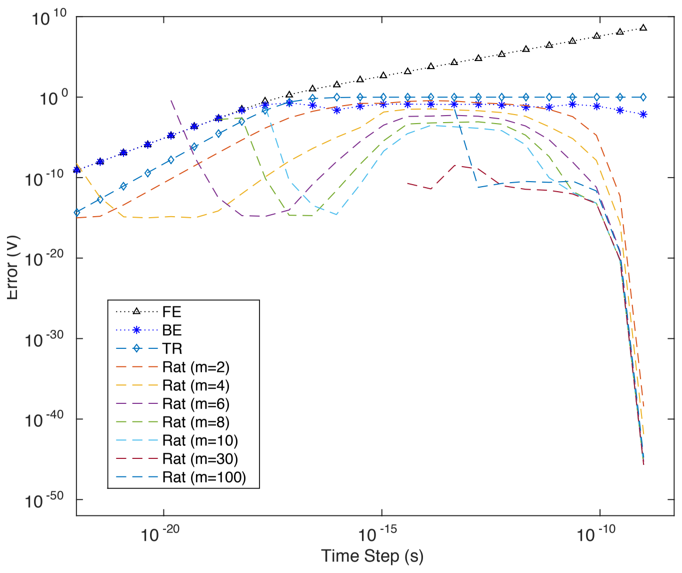
in Table 3.1.

Simulation of RLC Interconnect System

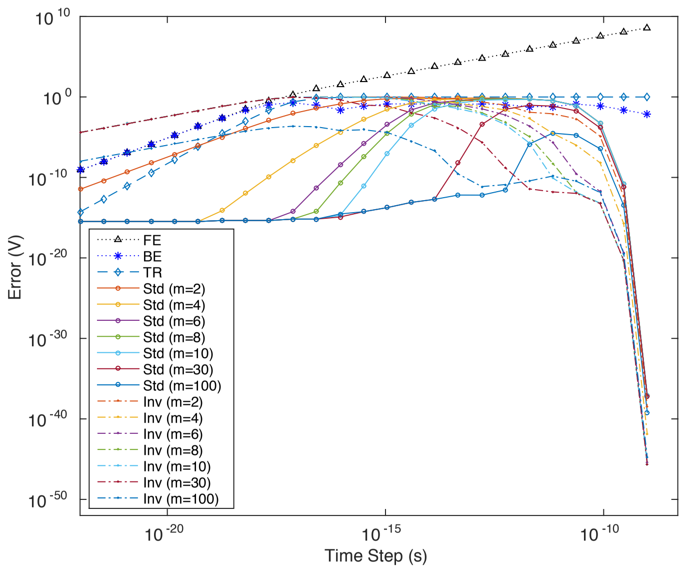
We investigate the numerical errors of Std, Inv, Rat, FE, BE, and TR using our RLC mesh. Fig. 3.7 shows the errors in the similar way as observed in Section 3.3.2. However, in the flat region ($\min(|\lambda_i|^{-1}) < h < \max(|\lambda_i|^{-1})$) the curves drop down slower as the dimension m increases, comparing with the curves in Fig. 3.5.

3.3.3 Error Distribution of Different Numerical Integration Approaches along the Simulation Time Span

Fig. 3.8 shows the simulation results of all the nodes. Fig. 3.10 (a) plots the distribution of global errors [7] of **(a)** Std, **(b)** Inv and **(c)** Rat vs. FE, BE and TR. The error of FE is extremely large due to the instability and jumps out of this figure after several steps. In Fig. 3.10(a), for Std, we need to increase dimension m to reduce the



(a)



(b)

Figure 3.5: RC circuit's error distribution of the one-step integration results via different linear integrators with the same initial vector $x(0)$ and different time step h. (a) Rat vs. FE, BE, and TR; (b) Std and Inv vs. FE, BE, and TR.

Table 3.1: Matrix Exponential Based High Order Integrators using Std, Inv, and Rat. vs. Low Order Integrators FE, BE and TR.

Method	$h \leq \min(\|\lambda_i\|^{-1})$	$\min(\|\lambda_i\|^{-1}) < h < \max(\|\lambda_i\|^{-1})$	$h \geq \max(\|\lambda_i\|^{-1})$
FE	2nd order	Diverge	Diverge
BE	2nd order	Flat	Flat
TR	3rd order	Flat (worse than BE)	Flat
Std ($m = 2$)	2nd order	Flat	Drop
Inv ($m = 2$)	1st order	Flat	Drop
Rat ($m = 2$)	1st order	Flat	Drop
Std ($m > 2$)	>2nd order	Curves shift to the right	Drop
Inv ($m > 2$)	1st order	Curves shift to the left	Drop
Rat ($m > 2$)	*	*	Drop

*: The curve of Rat depends on γ. For large γ, the curve is similar to Inv. For small γ, the curve is similar to Std. Otherwise, the shape of curve falls between Std and Inv. Moreover, for $m = 2$, the curve dips at $h = 2\gamma$. As dimension m increases, the dip point shifts to the right.

error even when the solution saturates toward steady state.

Fig. 3.10(b)(c) plot the global errors of Inv and Rat, respectively. As m increases, Inv and Rat improve the accuracy in faster pace than Std shown in Fig. 3.10(a). Fig. 3.9, we zoom into the time around $0.1ns$ of Fig. 3.10(b), where input changes from 0mA to 10mA. Inv has smaller error than BE when $m \geq 2$. (The error reduction of BE and TR in the time interval $[0.11ns, 0.3ns]$ is due to the solutions decay to the steady state, when the input becomes constant after $0.11ns$.)

In Fig. 3.11, we plot the distribution of local errors [7] of **(a)** Std, **(b)** Inv, and **(c)** Rat vs. FE, BE, and TR, which has the same trend with slightly smaller value than the global error in Fig. 3.10. The possible reason that Inv and Rat with $m = 31$ and $m = 78$ have reverse trend in Fig. 3.10 and Fig. 3.11 is due to the numerical precision.

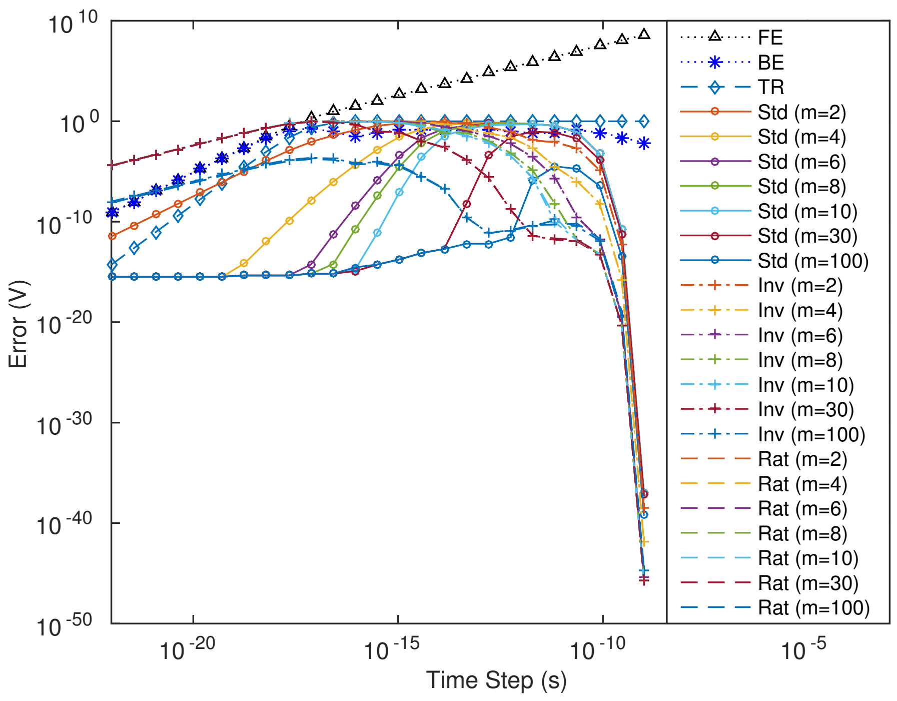


Figure 3.6: RC circuit's error distribution of the simulation results via different linear integrators with the same initial vector $x(0)$ and different time step h. Exponential integrators use the Krylov subspace basis dimension ($m = 2,\ 4,\ 6,\ 8,$ and 10) with $\gamma = 10^{-8}$ in Rat.

3.3.4 Comparisons among Different Krylov Subspace Algorithms for MEVP Computation and Convergence

In order to observe the error distribution versus dimensions of standard, invert, and rational Krylov subspace methods for MEVP, we reuse the RC circuit in this chapter, which has stiffness

$$\frac{Re(\lambda_{min})}{Re(\lambda_{max})} = 4.7 \times 10^6,$$

where $\lambda_{max} = -8.49 \times 10^{10}$ and $\lambda_{min} = -3.98 \times 10^{17}$ are the maximum and minimum eigenvalues of $A = -C^{-1}G$. Fig. 3.12 shows the relative error reductions along the increasing Krylov subspace dimension. The error reduction rate of rational Krylov subspace is the best, while the one of standard Krylov subspace requires huge dimension

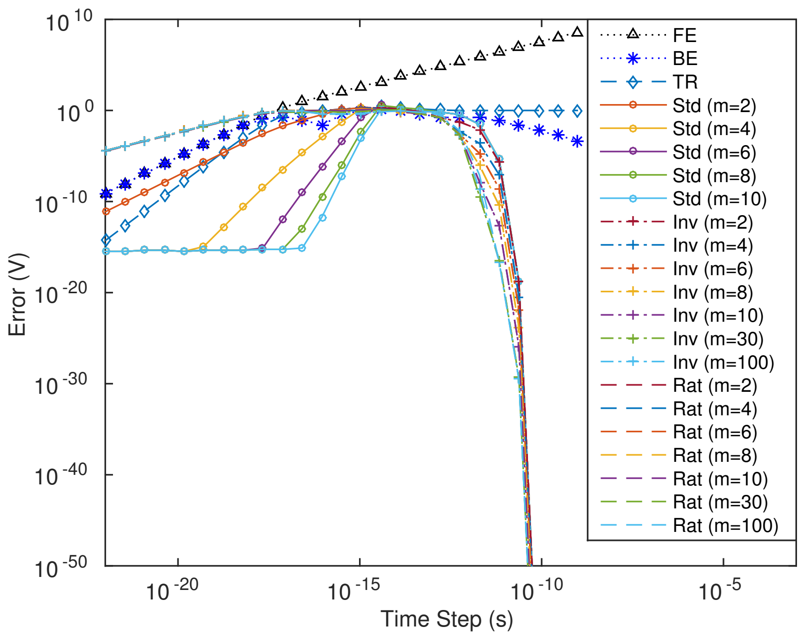


Figure 3.7: RLC circuit's error distribution of the simulation results via different linear integrators with the same initial vector $x(0)$ and different time step h. Exponential integrators use the Krylov subspace basis dimension ($m = 2,\ 4,\ 6,\ 8,\ 10$) with $\gamma = 10^{-8}$ in Rat.

to capture the same level of error. For example, it costs almost $10\times$ of the size to achieve around relative error 1% compared to invert and rational Krylov subspace methods. The relative error is

$$\frac{||e^{hA}v - \beta V_m e^{hH_m} e_1||}{||e^{hA}v||},$$

where $h = 0.4ps$ and $\gamma = 10^{-13}$. The matrix A is a relatively small matrix and computed by MATLAB *expm* function. The result of $e^{hA}v$ serves as the baseline for accuracy. The relative error is the real relative difference compared to the analytical solution $e^{hA}v$ of the ODE

$$\frac{dx}{dt} = Ax$$

with an initial vector v, which is generated by MATLAB *rand* function.

The error reduction rate of standard Krylov subspace is the worst, while the

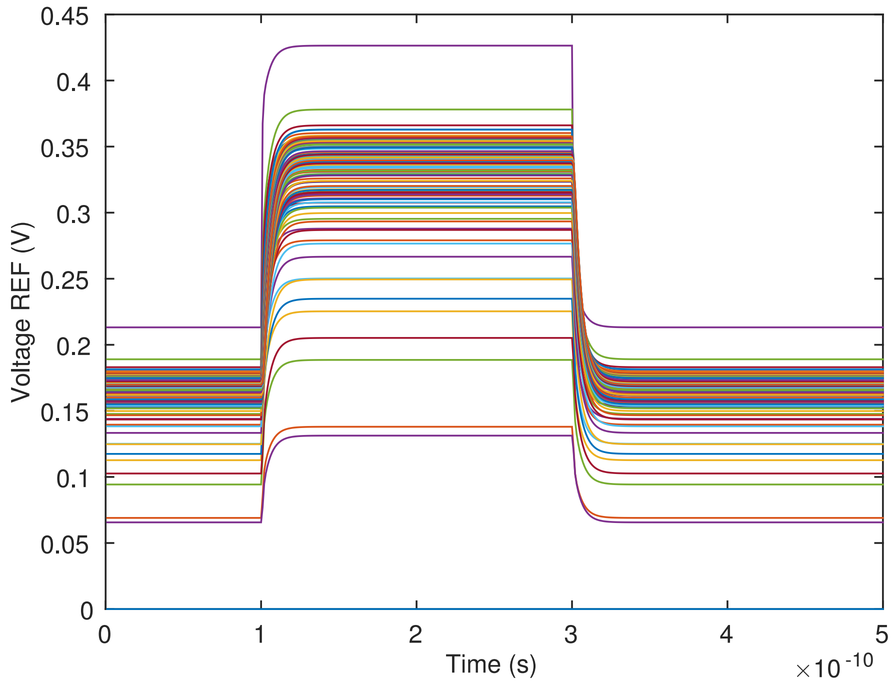


Figure 3.8: The reference simulation result of RC circuit with an input PWL (0 0mA, 100ps 0mA, 110ps 10mA, 300ps 10mA, 310ps, 0mA) during time span $[0,\ \ 5\times10^{-10}s]$.

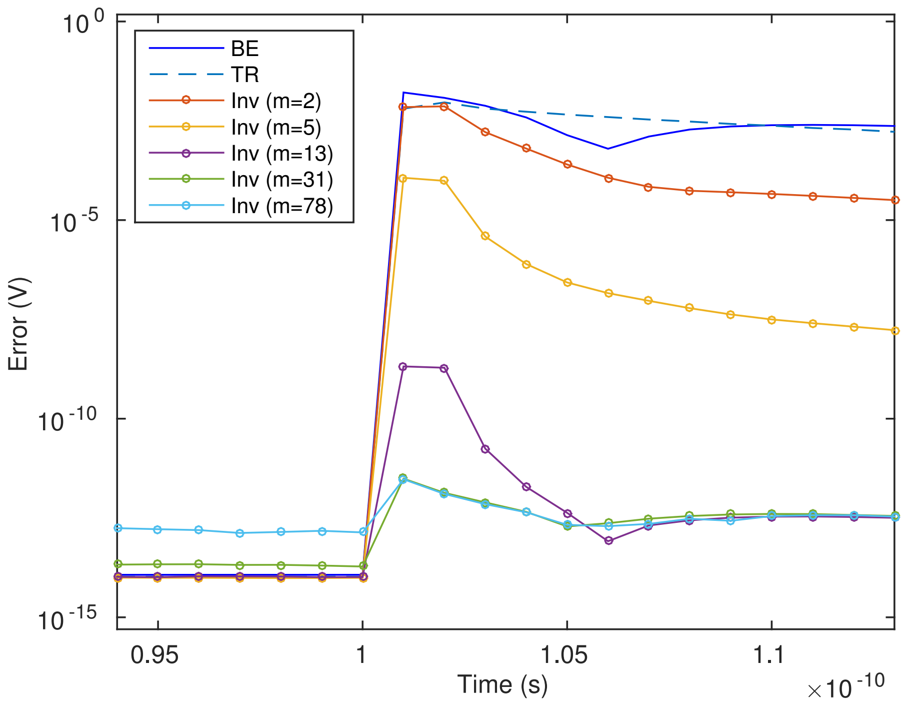


Figure 3.9: The zoom-in figure around $0.1ns$ of Fig. 3.10 (b).

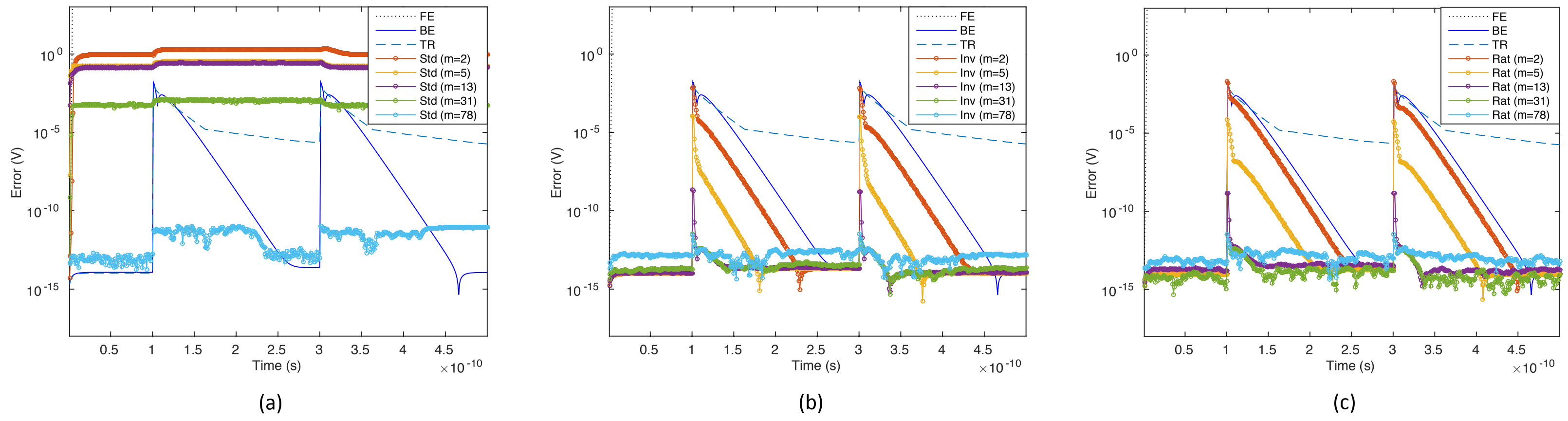


Figure 3.10: The global error distributions of the transient simulation using a RC circuit: (a) Std, (b), Inv and (c) Rat vs. FE, BE and TR.

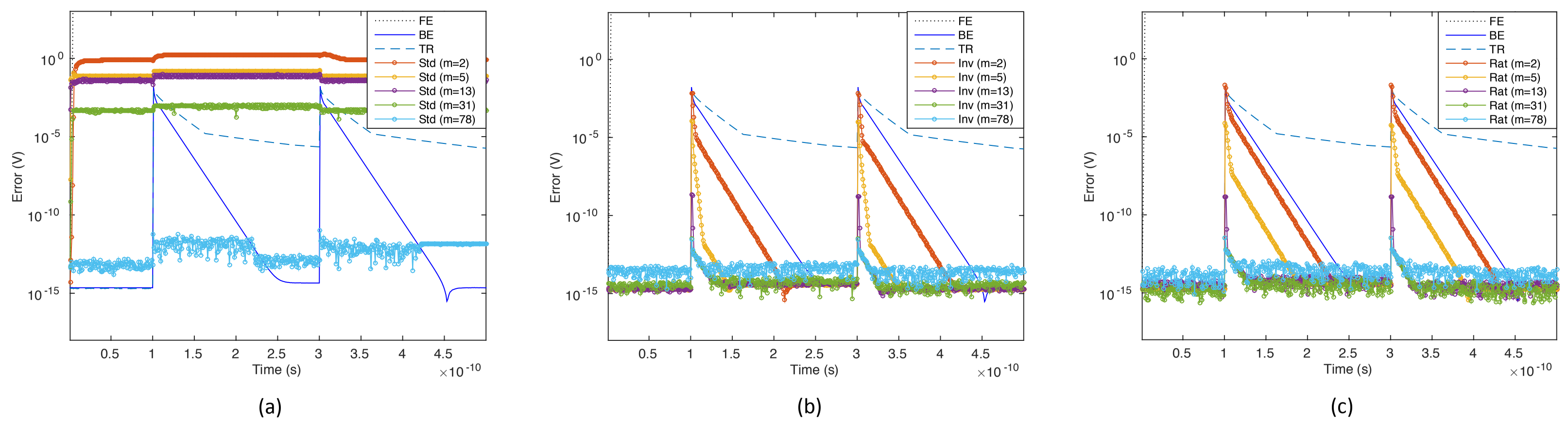


Figure 3.11: The local error distributions of the transient simulation using a RC circuit: (a) Std, (b), Inv and (c) Rat vs. FE, BE and TR.

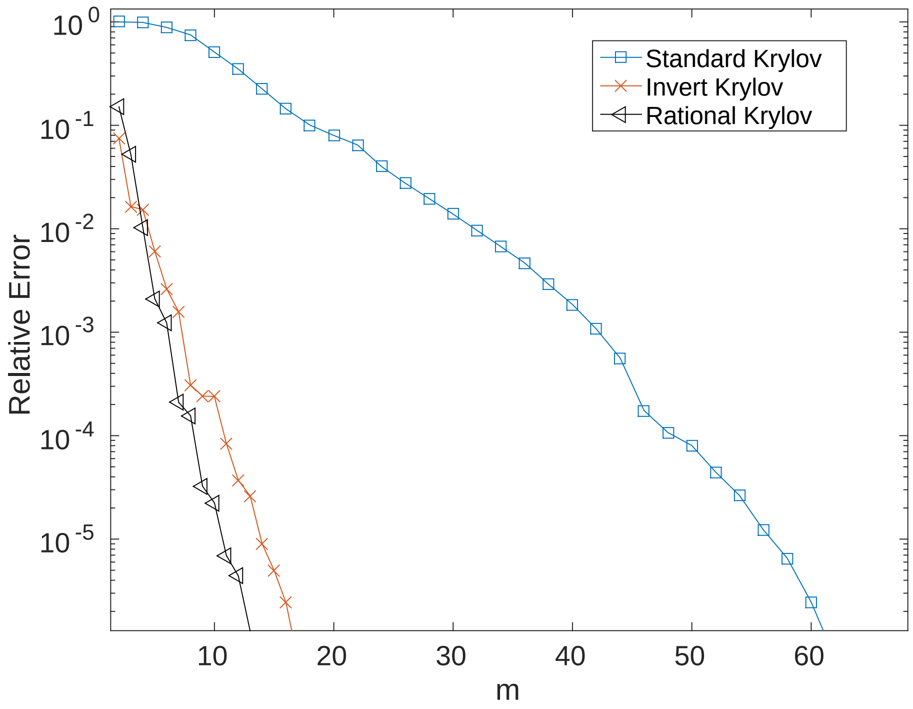


Figure 3.12: The relative error vs. dimensional m of different Krylov subspace methods.

rational Krylov subspace is the best. It is the reason that we prefer rational Krylov subspace. The relative errors of BE, TR and FE are 0.0594, 0.4628, and 2.0701×10^4, respectively. The large error of FE is due to the instability issue of its low order explicit time integration scheme. In Fig. 3.12, when $m = 3$, standard, invert and rational Krylov subspace methods have 0.8465, 0.0175, and 0.0065, respectively. It illustrates the power of matrix exponential method. Our proposed methods are all stable and can achieve improved error numbers when m increases.

In order to observe the different stiffness effects on Krylov subspace methods, we change the entries in C and G to make the different stiffness value 4.7×10^{10}. Fig. 3.13 illustrates the stable reduction rate of rational method. The stiffness degrades the performance of standard Krylov subspace method. Both invert and rational Krylov subspace methods are good candidates for stiff circuit system.

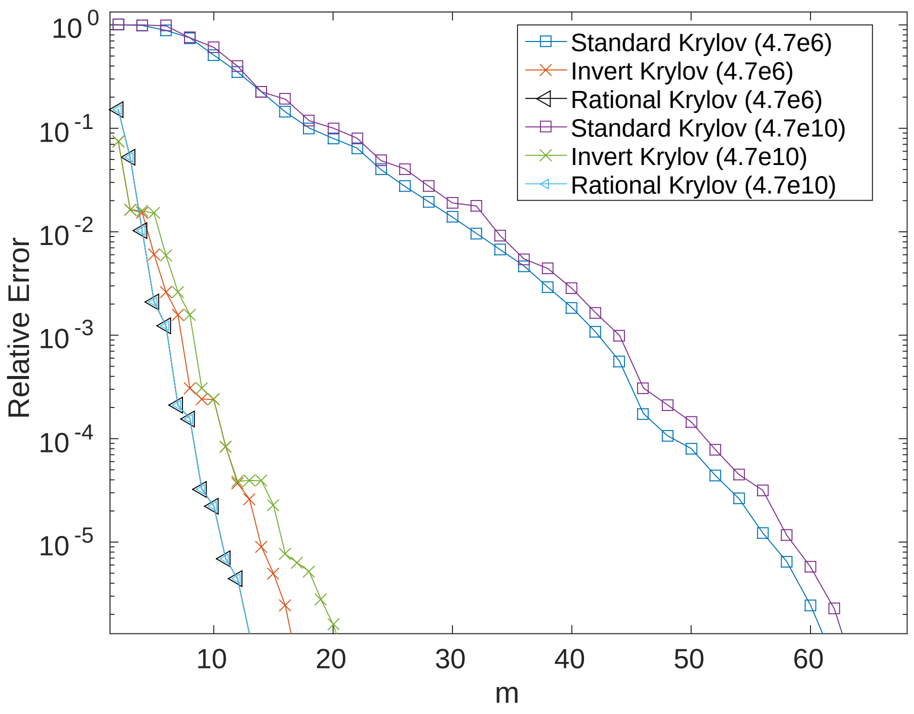


Figure 3.13: The relative error vs. dimension *m* of different Krylov subspace methods with two stiffness numbers.

Regarding the relative error distributions vs. time step *h* and dimension *m*, Fig. 3.14, Fig. 3.15, and Fig. 3.16 are computed by standard, invert, and rational Krylov subspaces ($\gamma = 5 \times 10^{-13}$), respectively. Fig. 3.14 shows that the errors generated by standard Krylov subspace method has flat region with high error values in time-step range of interests. The very small time step range has small error values. Compared to Fig. 3.14, invert (Fig. 3.15) and rational (Fig. 3.16) Krylov subspace methods reduce errors quickly for large *h*. The explanation is that a relatively small portion of the eigenvalues and corresponding invariant subspaces determines the final result (vector) when time step *h* is larger [42], which are efficiently captured by invert and rational Krylov subspace methods.

The error of rational Krylov subspace is relatively insensitive to γ when it is selected between the time-step range of interests (Fig. 3.17). Above all, rational Krylov

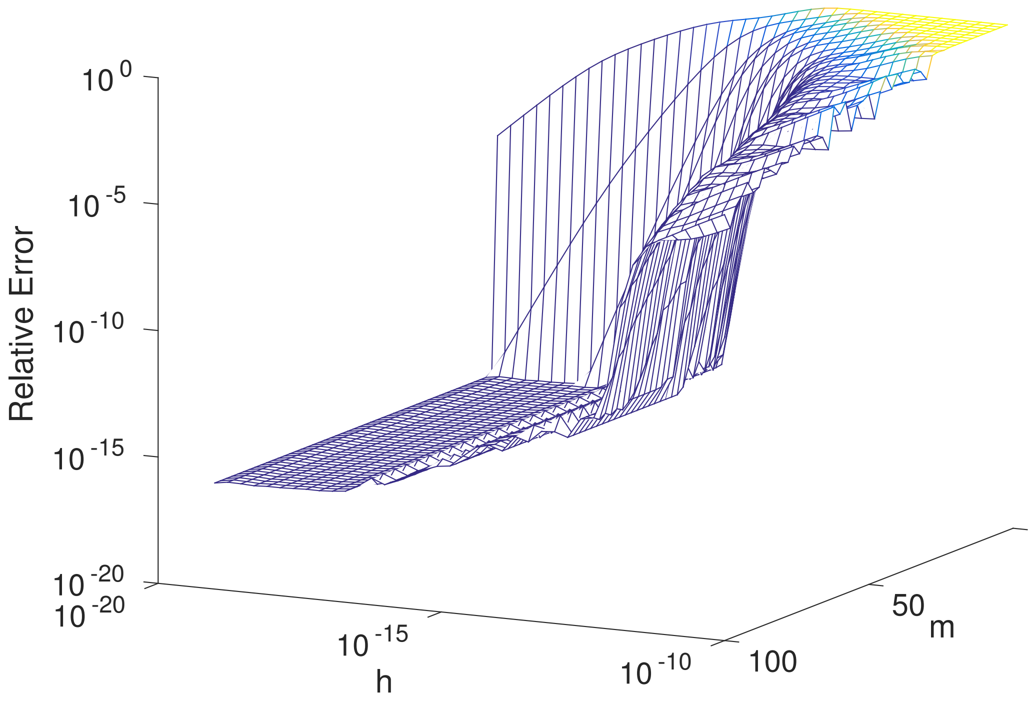


Figure 3.14: The error of MEVP via standard Krylov Subspace: $\frac{||e^{hA}v-\beta V_m e^{hH_m}||}{||e^{hA}v||}$ vs. h and the dimension of subspace (m). This method approximates the solution well in extremely small h, since it captures the important eigenvalues and eigenvectors of A at that region.

and invert Krylov subspace methods have much better performance than standard version. When we deal with stiff cases, standard Krylov subspace is not a feasible choice due to the large dimension m of Krylov subspace, which causes huge memory consumption and poor runtime performance.

3.4 Summary

In this section, we demonstrate the numerical performance of the matrix exponential based integrators. Krylov methods for MEVP can alter their orders to improve accuracy, which is not possible for traditional linear multi-step methods. In general, in

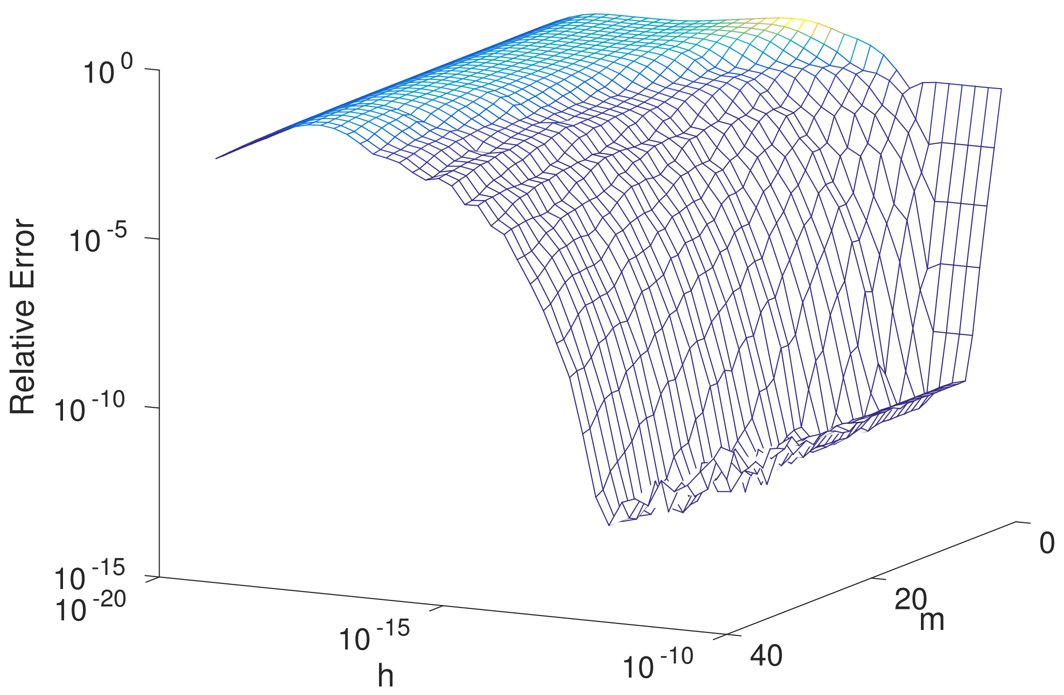


Figure 3.15: The error of MEVP via invert Krylov Subspace: $\frac{||e^{hA}v-\beta V_m e^{hH_m^{-1}}||}{||e^{hA}v||}$ vs. time step h and dimension of invert Krylov subspace basis (m).

a stiff system, simulation can have time step h much larger than the feasible range of Taylor expansion. Traditional linear multi-step approach relies on the marching in time to drive the errors down, while matrix exponential approach can pull down the error by increasing the dimension of the Krylov subspace. For transient analysis, the eigenvalues of small real magnitude are wanted to describe the dynamic behavior. Therefore, for the Krylov variants, invert (Inv) and rational (Rat) Krylov methods are good choices.

More importantly, exponential based integration schemes with Krylov subspaces have three distinguished features:

(1) For invert and rational Krylov subspace methods, the larger is time step, the smaller errors we will have. This phenomenon is consistent with the result of van den Eshof and Hochbruck in [42].

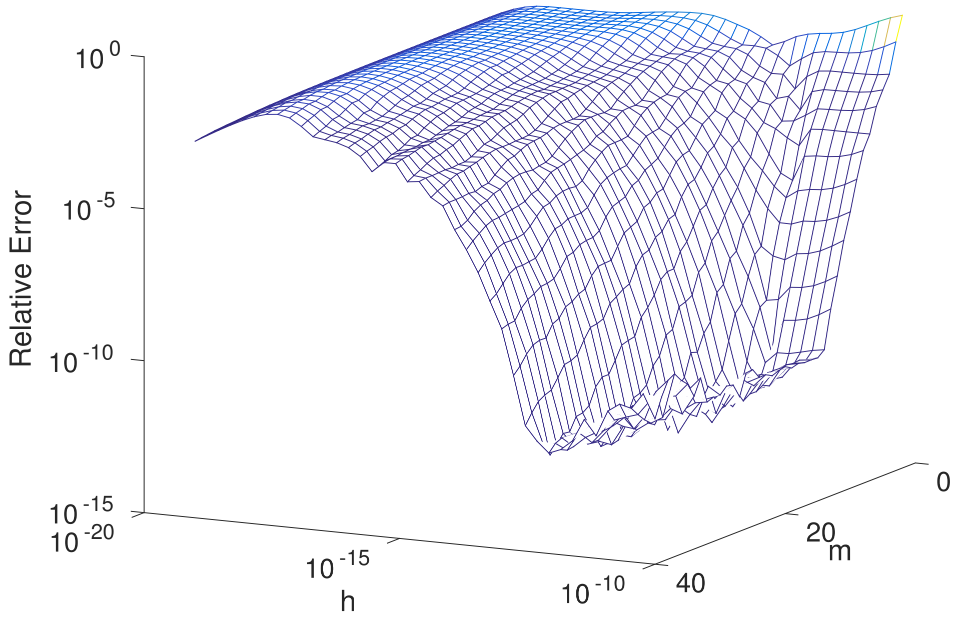


Figure 3.16: The error of MEVP via rational Krylov subspace: $\frac{||e^{hA}v-\beta V_m e^{h\frac{I-H_m^{-1}}{\gamma}} e_1||}{||e^{hA}v||}$, where $\gamma = 5\times 10^{-13}$, vs. time step h and dimension of subspace (m).

(2) Invert Krylov subspace method can avoid the factorization of matrix C, so that it can solve the post-layout simulation when the capacitance/inductance matrix C is complicated (relatively denser than pre-layout, or strong coupled systems), while the complexities by standard methods may increase dramatically.

(3) The explicit formulation is stable by matrix exponential operators and Krylov subspace methods. Thus, for nonlinear system, we can skip the procedures needed in implicit method such as NR iteration.

Chapter 3, in part, is a reprint of the material as it appears in "From Circuit Theory, Simulation to SPICE^{Diego}: A Matrix Exponential Approach for Time Domain Analysis of Large Scale Circuits" by Hao Zhuang, Xinyuan Wang, Quan Chen, Pengwen Chen, and Chung-Kuan Cheng in *IEEE Circuits and Systems Magazine*. The chapter

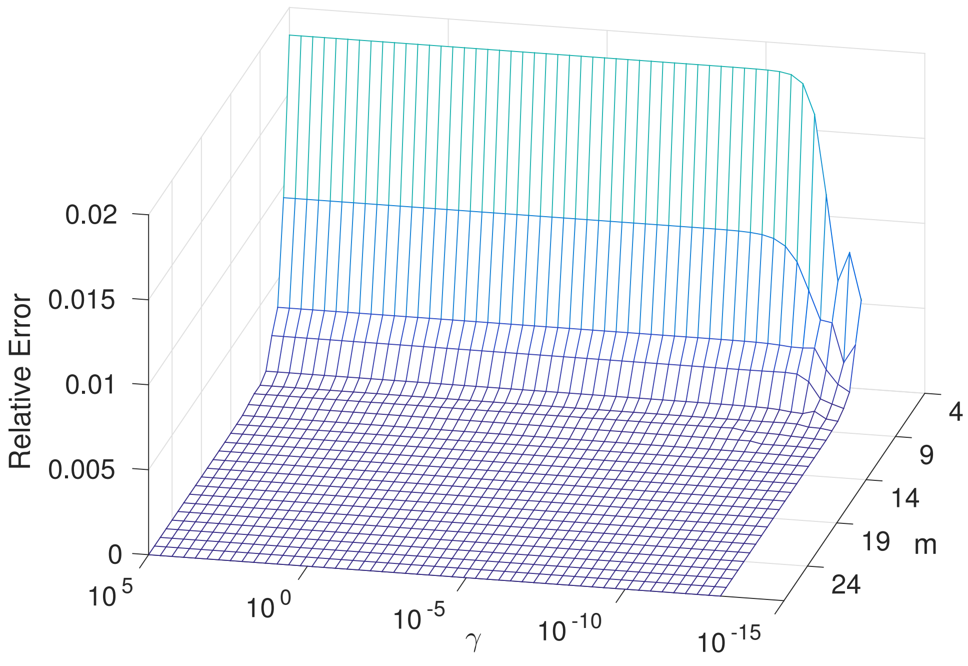


Figure 3.17: The error of MEVP via rational Krylov Subspace $\frac{||e^{hA}v-\beta V_m e^{h\frac{I-H_m^{-1}}{\gamma}} e_1||}{||e^{hA}v||}$, where $h = 4ps$.

also contains the content from ‘Simulation Algorithms with Exponential Integration for Time-Domain Analysis of Large-Scale Power Delivery Networks” by Hao Zhuang, Wenjian Yu, Shih-Hung Weng, Ilgweon Kang, Jeng-Hau Lin, Xiang Zhang, Ryan Coutts, and Chung-Kuan Cheng in *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems*. The thesis author was the primary investigator and author of the papers.

Chapter 4

Exponential Integration for Linear Dynamical Systems

In this chapter, we apply the proposed exponential integration into a linear circuit simulation framework for large-scale power network simulation, which is a very practical but challenging task during VLSI signoff. First, we summarize the motivation of this application, the problem formulation, and the conventional approaches in this area. Second, we propose the simulation framework with exponential integration and different Krylov subspace approaches. Third, a distributed computation model is also demonstrated in this chapter. In numerical results, IBM power grid benchmarks are used to test the performance of our approaches.

4.1 Motivation

The linear circuit simulation plays an important role in transient analysis of large scale circuits. A typical example is the power network simulation. VLSI design verification relies heavily on the analysis of power delivery network (PDN) to estimate power supply noises [45–52]. The performance of power delivery network highly impacts on the quality of global, detailed and mixed-size placement [53–55], clock tree synthesis [56], global and detailed routing [57], as well as timing [58] and power optimization. Lowering supply voltages, increasing current densities as well as tight design margins demand more accurate large-scale PDN simulation. Advanced technologies [59, 60], three dimensional (3D) IC structures [61, 62], and increasing complexities of system designs all make VLSI PDNs extremely huge and the simulation tasks time-consuming and computationally challenging. Due to the enormous size of modern designs and long simulation runtime of many cycles, instead of general nonlinear circuit simulation [39, 63], PDN is often modeled as a large-scale linear circuit with voltage supplies and time-varying current sources [64, 65]. Those linear matrices are obtained by parasitic extraction process [2, 24, 25, 48, 66]. After those processes, we need time-domain large-scale linear circuit simulation to obtain the transient behavior of PDN with above inputs.

Traditional methods in linear circuit simulation solve differential algebra equations (DAE) numerically in explicit ways, e.g., forward Euler (FE), or implicit ways, e.g., backward Euler (BE) and trapezoidal (TR), which are all based on low order polynomial approximations for DAEs [7]. Due to the stiffness of systems, which comes from a wide range of time constants of a circuit, the explicit methods require extremely small time step sizes to ensure the stability. In contrast, implicit methods can handle this problem with relatively large time steps because of their larger stability regions. However, at each time step, these methods have to solve a linear system, which is sparse and often ill-conditioned. Due to the requirement of a robust solution, compared to iterative methods [67], direct methods [28] are often favored for VLSI circuit simulation, and thus adopted by state-of-the-art power grid (PG) solvers in TAU PG simulation contest [68–70]. Those solvers only require one matrix factorization (LU or Cholesky factorization) at the beginning of the transient simulation. Then, at each fixed time step, the following transient computation requires only pairs of forward and backward substitutions, which achieves better efficiency over adaptive stepping methods by reusing the factorization matrix [65, 68, 70] in their implicit numerical integration framework. However, the maximum of step size choice is limited by the smallest distance h_{upper} among the breakpoints [34]. Some engineering efforts are spent to break this limitation by sacrificing the accuracy. In this study, we constraint our scope and always obey the upper limit h_{upper} of time step to maintain the fidelity of model, which means the fixed time step h cannot go beyond h_{upper} in case of missing breakpoints.

4.2 Problem Formulation of Transient Analysis of Linear Power Delivery Networks

Transient simulation of linear circuit is the foundation of modern PDN simulation. It is formulated as DAEs via modified nodal analysis (MNA),

$$C\frac{dx}{dt} = -Gx + Bu(t), \tag{4.1}$$

where C is the matrix for capacitive and inductive elements. G is the matrix for conductance and resistance, and B is the input selector matrix. $x(t)$ is the vector of time-varying node voltages and branch currents. $u(t)$ is the vector of supply voltage and current sources. In PDN, such current sources are often characterized as pulse or piecewise-linear inputs [64, 65] to represent the activities under the networks. To solve Eq. (4.1) numerically, the system is discretized with time step h and transformed to a linear algebraic system. Given an initial condition $x(0)$ from DC analysis or previous time step $x(t)$ and a time step h, $x(t+h)$ can be obtained by traditional *low order approximation* methods [7].

4.2.1 Traditional Low Order Time Integration Schemes with Fixed Time Step

Methods BE and TR with fixed time step (FTS) h are regarded as efficient approaches in large-scale PDN simulation, which were adopted by the top PG solvers in 2012 TAU PG simulation contest [65,68–71].

BE-FTS

Backward Euler based time integration scheme with a fixed h Eq.(4.2) is a robust implicit first-order method in the transient analysis of PDN.

$$\left(\frac{C}{h}+G\right)x(t+h)=\frac{C}{h}x(t)+Bu(t+h). \tag{4.2}$$

TR-FTS

Trapezoidal based time integration scheme with a fixed h Eq.(4.3) is a popular implicit second-order method in the transient analysis of PDN.

$$\left(\frac{C}{h}+\frac{G}{2}\right)x(t+h)=\left(\frac{C}{h}-\frac{G}{2}\right)x(t)+B\frac{u(t)+u(t+h)}{2}.$$

Take TR-FTS for example,

$$LUx(t+h)=(\frac{C}{h}-\frac{G}{2})x(t)+B\frac{u(t+h)+u(t)}{2}, \tag{4.3}$$

where

$$LU=\frac{C}{h}+\frac{G}{2}.$$

This formulation reuses LU matrix factorization, which is the most expensive step in the whole simulation. However, if only one h is used along the whole simulation, the choice is bounded by the minimum distance between breakpoints [34] among all the input sources. In Fig. 4.1 (a), the alignment of the two inputs makes $10ps$ as the upper limit for time step h. When the alignments of inputs shift by $5ps$ as shown in Fig. 4.1 (b), the resulting upper limit for h is $5ps$ for those fixed step size based approaches. If h is larger than the limit, it is impossible to guarantee the accuracy since we may skip pivot points of the inputs.

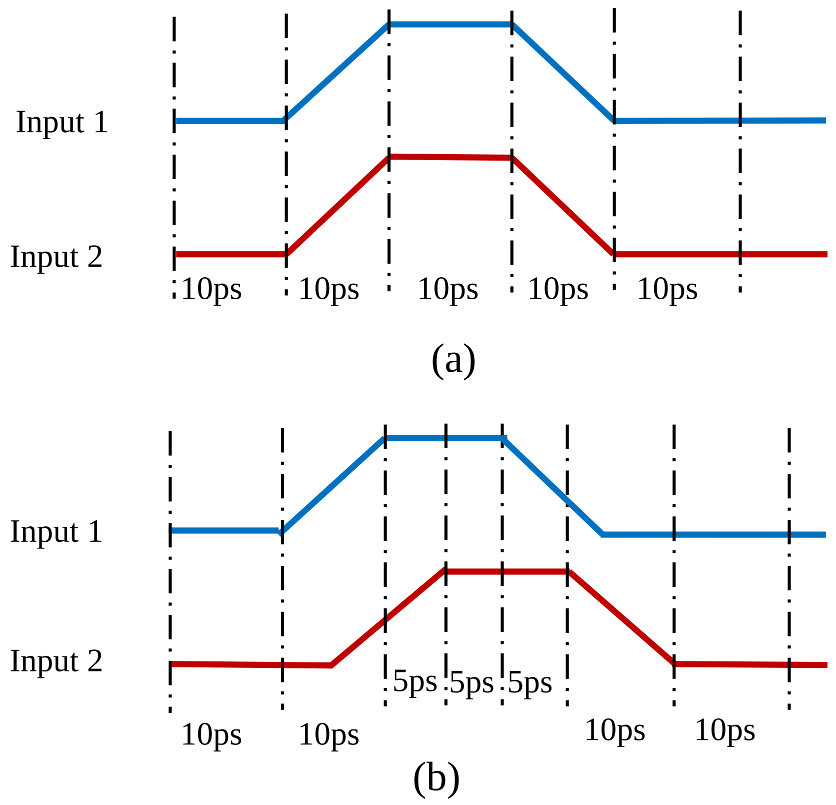


Figure 4.1: Example: interleaves two input sources to create smaller transition time. (**a**) Before interleaving, the input sources have smallest transition time $h_{upper} = 10ps$; (**b**) After interleaving, the input sources have the smallest transition time $h_{upper} = 5ps$.

In summary, there are major issues in the conventional PDN solver: (1) Step size is fixed to avoid multiple matrix factorizations, which constraints the time step choice.

(2) The relatively small time step is used in the low order numerical integration scheme, due to the requirement of accuracy. (3) In recent development [72], a set of step sizes is used to adjust according to LTE prediction. The approach accelerates runtime at the expense of pre-computed matrix factorizations for the set of h.

4.2.2 Exponential Time Integration Scheme

The solution of Eq. (4.1) can be obtained analytically [7]. For a simple illustration, we convert Eq. (4.1) into

$$\frac{dx}{dt} = Ax + b(t), \tag{4.4}$$

when C is not singular[1],

$$A = -C^{-1}G \text{ , and } b(t) = C^{-1}Bu(t).$$

Given a solution at time t and a time step h, the solution at $t+h$ is

$$x(t+h) = e^{hA}x(t) + \int_0^h e^{(h-\tau)A}b(t+\tau)d\tau. \tag{4.5}$$

Assuming that the input $u(t)$ is a piecewise linear (PWL) function of t, we can integrate the last term of Eq. (4.5) analytically, turning the solution with matrix

[1]The assumption is to simplify the explanation in this section. After Sec. 3.2.2, we use I-MATEX, R-MATEX and DR-MATEX to compute the solution of DAE without inversion of C. Therefore, the methods are suitable for general DAE system, i.e., Eq. (4.1) without the assumption here.

exponential operator:

$$\begin{aligned} x(t+h) &= -\left(A^{-1}b(t+h)+A^{-2}\frac{b(t+h)-b(t)}{h}\right)+ \\ &+ e^{hA}\left(x(t)+A^{-1}b(t)+A^{-2}\frac{b(t+h)-b(t)}{h}\right) \\ &= e^{hA}(x(t)+F(t,h))-P(t,h), \end{aligned} \tag{4.6}$$

where

$$F(t,h)=A^{-1}b(t)+A^{-2}\frac{b(t+h)-b(t)}{h} \tag{4.7}$$

and

$$P(t,h)=A^{-1}b(t+h)+A^{-2}\frac{b(t+h)-b(t)}{h}. \tag{4.8}$$

For the time step choice, breakpoints (also known as input transition spots (TS) [37]) are the time points where slopes of input vector change. Therefore, for Eq. (4.6), the maximum time step starting from t is (t_s-t), where t_s is the smallest one in TS larger than t. In matrix exponential based framework, the limitation of time step size is not the local truncation error (LTE), but the activities among all input sources.

4.2.3 Matrix Exponential Based PDN Solver with Rational Krylov Subspace Basis

Rational Krylov subspace method can use larger time step and still achieve smaller error (Lemma 3.1 in [42]). This property leads us to utilize large stepping and parallel computation for PDN simulation without accuracy compromise [37, 38].

Lemma 4.2.1. *(Lemma 3.1 in [42]) Let μ be such that $A-\mu I$ is positive semidefinite.*

Then

$$\|V_m f(H_m)e_1 - e^{-hA}v\| \leq 2e^{-h\mu}E_{m-1}^{m-1}(\widetilde{\gamma})$$

with $\widetilde{\gamma} = \frac{\gamma}{h(1+\gamma\mu)}$. *The term* $E_{m-1}^{m-1}(\widetilde{\gamma})$ *is defined in [42].*

Lemma 4.2.1 informs a trend that the error bound reduces due to the term $e^{-h\mu}$, when time step h becomes large enough after a certain scale (e.g., $\max(|\lambda_i|^{-1})$ in Fig. 3.6). Therefore, we can use a large step size and obtain accurate enough solution. With the capability of large time stepping, we can choose any time point $t+h \in [t, t_s]$ (t_s is the next input break point) when the matrices and vectors of system stay constant and share the same Krylov subspace at time point t. Based on Eq. (3.29), there is no matrix factorization when $h \leq t_s$. Since the model of PDN is a linear dynamic system, we can reuse Krylov subspace as long as there are no input breakpoints encountered. The computation formulation based on Eq. (4.6) and Krylov subspace is

$$x(t+h) = \|v\| V_m e^{hH_m} e_1 - P(t,h). \qquad (4.9)$$

Furthermore, if we fix the value of γ, we can reuse the matrices by one factorization process as TR-FTS for the whole transient simulation, and also utilize adaptive stepping via rational Krylov subspace. We sketch the process in Algorithm 4.

4.3 MATEX: A Exponential Integration Based Framework for Power Network Analysis

4.3.1 MATEX Circuit Solver

We incorporate matrix exponential based integration scheme with Krylov subspace method into our MATEX framework, which is summarized in Algorithm 4. We set

X_1 and X_2 in Line 1 based on the choice of Krylov subspace method as follows,

- I-MATEX: $X_1 = -G$, $X_2 = C$
- R-MATEX: $X_1 = C + \gamma G$, $X_2 = C$

For linear system of PDN, the matrix factorization in line 4 is only performed once, and the matrices L and U are reused in the while loop from line 5 to line 10. Line 8 uses Arnoldi process with corresponding inputs to construct Krylov subspace as shown in Algorithm 3.

Algorithm 3: MATEX_Arnoldi

```
    Input: $L, U, X_2, h, t, x(t), \varepsilon, P(t,h), F(t,h)$
    Output: $x(t+h), V_m, H, v$
 1  $v = x(t) + F(t,h)$;
 2  $v_1 = \frac{v}{\|v\|}$;
 3  for $j = 1 : m$ do
 4      $w = U\backslash(L\backslash(X_2 v_j))$ ; /* a pair of forward and backward
        substitutions.                                    */
 5      for $i = 1 : j$ do
 6          $h_{i,j} = w^T v_i$;
 7          $w = w - h_{i,j} v_i$;
 8      end
 9      $h_{j+1,j} = \|w\|$;
10      $v_{j+1} = \frac{w}{h_{j+1,j}}$;
11      if $r(j,h) < \varepsilon$ then
12          $m = j$;
13          break;
14      end
15  end
16  $x(t+h) = \|v\| V_m e^{hH} e_1 - P(t,h)$;
```

Algorithm 4: MATEX Circuit Solver

```
Input: $C, G, B, u, \varepsilon$, and time span $T$.
Output: The set of $x$ from $[0, T]$.
1  Set $X_1, X_2$;
2  $t = 0$;
3  $x(t)$ =DC_analysis;
4  $[L, U] = $ LU_Decompose$(X_1)$;
5  while $t < T$ do
6      Compute maximum allowed step size $h$;
7      Update $P(t,h), F(t,h)$;
8      Obtain $x(t+h)$ by Algorithm 3 with inputs
       $[L, U, X_2, h, t, x(t), \varepsilon, P(t,h), F(t,h)]$;
9      $t = t + h$;
10 end
```

4.4 DR-MATEX: A Distributed Framework of MATEX

4.4.1 Motivation

There are usually many input sources in PDNs as well as their transition activities, which might narrow the regions for the stepping of matrix exponential based method due to the unaligned breakpoints. In other words, the region before the next transition t_s may be shortened when there are a lot of activities from the input sources. It leads to more chances of generating new Krylov subspace bases. We want to reduce the number of subspace generations and improve the runtime performance.[2]

4.4.2 Treatment and Methodology

In matrix exponential based integration framework, we can choose any time spot $t + h \in [t, t_s]$ with computed Krylov subspace basis. The solution of $x(t+h)$ is computed

[2]The breakpoints also put the same constraint on TR-FTS and BE-FTS. However, their time steps are fixed already, which refrains them from reaching this problem in the first place.

by scaling the existing Hessenberg matrix H with the time step h as below

$$x(t+h) = \|v\| V_m e^{hH} e_1 - P(t,h). \tag{4.10}$$

This is an important feature for computing the solutions at intermediate time points without generating the Krylov subspace basis, when there is no current transition. Besides, since the PDN is linear dynamical system, we can utilize the well-known superposition property of linear system and distributed computing model to tackle this challenge.

To illustrate our distributed version of MATEX framework, we first define three terms to categorize the breakpoints of input sources:

- *Local Transition Spot* (LTS): the set of TS at an input source to the PDN.
- *Global Transition Spot* (GTS): the union of LTS among all the input sources to the PDN.
- *Snapshot*: a set $GTS \setminus LTS$ at one input source.

If we simulate the PDN with respect to all the input sources, the points in the set of GTS are the places where generations of Krylov subspace cannot be avoided. For example, there are three input sources in a PDN (Fig. 4.2). The input waveforms are shown in Fig. 4.3. The first line is GTS, which is contributed by the union of LTS in input sources #1, #2 and #3. However, we can partition the task into subtasks by simulating each input source individually. Then, each subtask generates Krylov subspaces based on its own LTS and keeps track of *Snapshot* for the later usage of summation via linear superposition. Between two LTS points t and $t+h$, the *Snapshot* points

$$t+h_1 < t+h_2 < \cdots < t+h_l \in (t, t+h]$$

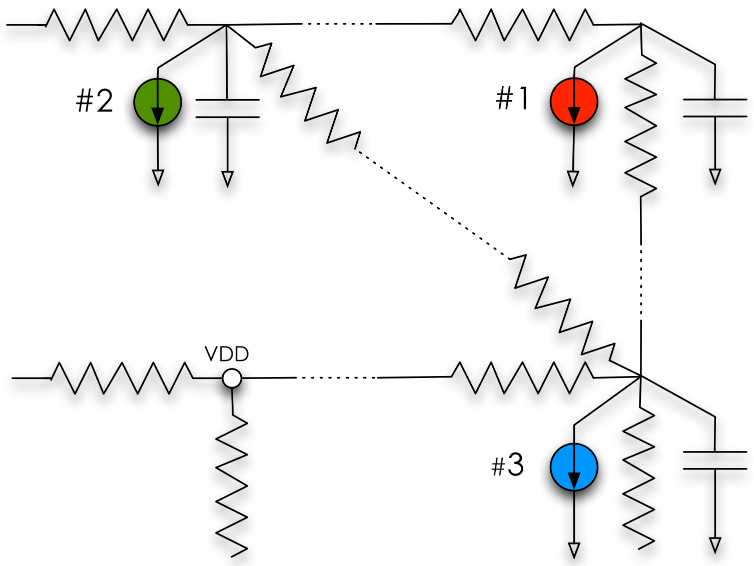


Figure 4.2: Part of a PDN model with input sources from Fig. 4.3.

can reuse the Krylov subspace generated at t. For each node, the chances of generation of Krylov subspaces are reduced. The time periods of reusing latest Krylov subspaces are enlarged locally and bring the runtime improvement. Besides, when subtasks are assigned, there is no communication among the computing nodes, which leads to so-called *Embarrassingly Parallel* computation model.

4.4.3 More Aggressive Tasks Decomposition

We divide the simulation task based on the alignments of input sources. More aggressively, we can decompose the task according to the "bump" shapes of the input sources.[3] We group the input sources, which have the same

$$(t_{delay}, t_{rise}, t_{fall}, t_{width})$$

[3]IBM power grid benchmarks provide the pulse input model in SPICE format.

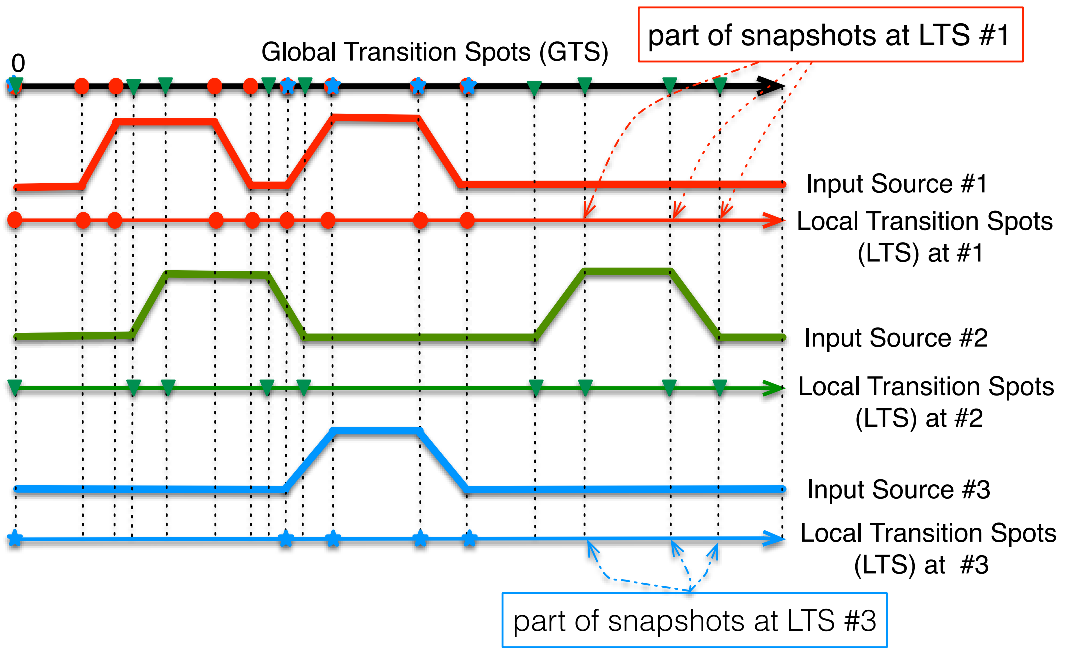

part of snapshots at LTS #1
0
Global Transition Spots (GTS)
Input Source #1
Local Transition Spots (LTS) at #1
Input Source #2
Local Transition Spots (LTS) at #2
Input Source #3
Local Transition Spots (LTS) at #3
part of snapshots at LTS #3

Figure 4.3: Illustration of input transitions. *GTS*: Global Transition Spots; *LTS*: Local Transition Spots; *Snapshots*: the crossing positions by dash lines and LTS #*k* without solid points.

into one set. For example, the input source #1 of Fig. 4.3 is divided to #1.1 and #1.2 in Fig. 4.4. The input source #2 in Fig. 4.3 is divided to #2.1 and #2.2 in Fig. 4.4. Therefore, there are four groups in Fig. 4.4, Group 1 contains *LTS*#1.1. Group 2 contains *LTS*#2.1. Group 3 contains *LTS*#2.2. Group 4 contains *LTS*#1.2 and #3. Our proposed framework MATEX is shown in Fig. 4.5. After pre-computing *GTS* and decomposing *LTS* based on "bump" shape (Fig. 4.4), we group them and form LTS #1 $\sim$ #K.[4]

4.4.4 MATEX Scheduler in DR-MATEX

In DR-MATEX, the role of MATEX scheduler is just to send out *GTS* and *LTS* to different MATEX slave nodes and collect final results after all the subtasks of transient simulation are finished. The node number is based on the total number of subtasks, which

[4]There are alternative decomposition strategies. It is also easy to extend the work to deal with different input waveforms. We try to keep this part as simple as possible to emphasize our framework.

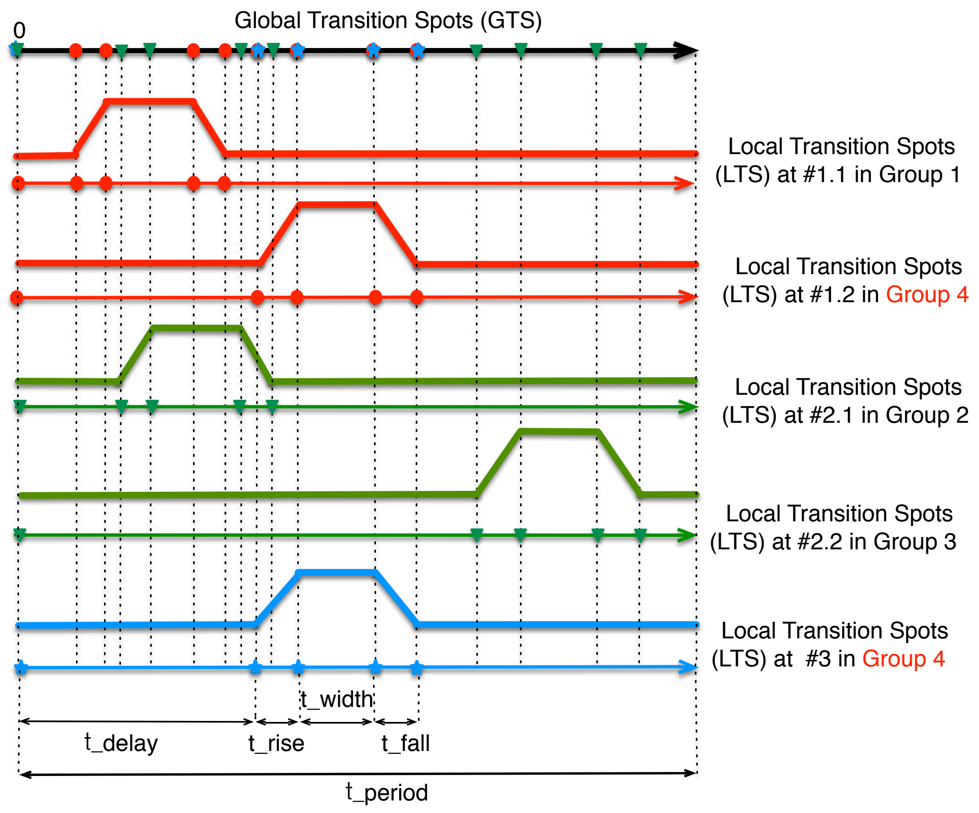

0
Global Transition Spots (GTS)
Local Transition Spots (LTS) at #1.1 in Group 1
Local Transition Spots (LTS) at #1.2 in Group 4
Local Transition Spots (LTS) at #2.1 in Group 2
Local Transition Spots (LTS) at #2.2 in Group 3
Local Transition Spots (LTS) at #3 in Group 4
t_width
t_delay
t_rise
t_fall
t_period

Figure 4.4: Grouping of "Bump" shape transitions for sub-task simulation. Proposed exponential based method can utilize adaptive stepping in each LTS and reuse subspace generated at the latest point in LTS.

is the group number after PDN source decomposition. Then the simulation computations are performed in parallel. Each node has its own inputs. For example, Node#k has GTS, $LTS\#k$, P_k and F_k, which contain the corresponding b for node k. Scheduler does not need to do anything during the transient simulation, since there are no communications among nodes before the stage of "write back" (in Fig. 4.5), by when all nodes complete their transient simulations.

Within each slave node, the circuit solver (Algorithm 5) computes transient response with varied time steps. Solutions are obtained without re-factorizing matrix during the computation of transient simulation. The computing nodes write back the results and inform the MATEX scheduler after finishing their own transient simulation.

Algorithm 5: DR-MATEX: The distributed MATEX framework using R-MATEX at Node#k.

```
Input: $LTS\#k$, $GTS$, $P_k$, $F_k$, error tolerance $E_{tol}$, and simulation time span $T$.
Output: Local solution $x$ along $GTS$ in node $k \in [1, \cdots, S]$, where $S$ is the
        number of nodes
1  $t = 0$, $X_1 = C + \gamma G$, and $X_2 = C$;
2  $x(t) = $ Local_Initial_Solution;
3  $[L, U] = $ LU_Decompose$(X_1)$;
4  while $t < T$ do
5      Compute maximum allowed step size $h$ based on $GTS$;
6      if $t \in LTS\#k$ then
           /* Generate Krylov subspace for the point at $LTS\#k$ and
              compute $x(t+h)$                                        */
7          $[x(t+h), V_m, H_m, v] = $ MATEX_Arnoldi$(L, U, X_2,$
           $h, t, x(t), \varepsilon, P_k(t,h), F_k(t,h))$;
8          $a_{lts} = t$;
9      end
10     else
           /* Obtain $x(t+h)$ at Snapshot with computed Krylov subspace
              */
11         $h_a = t + h - a_{lts}$;
12         $x(t+h) = \|v\| V_m e^{h_a H_m} e_1 - P_k(t,h)$;
13     end
14     $t = t + h$;
15 end
```

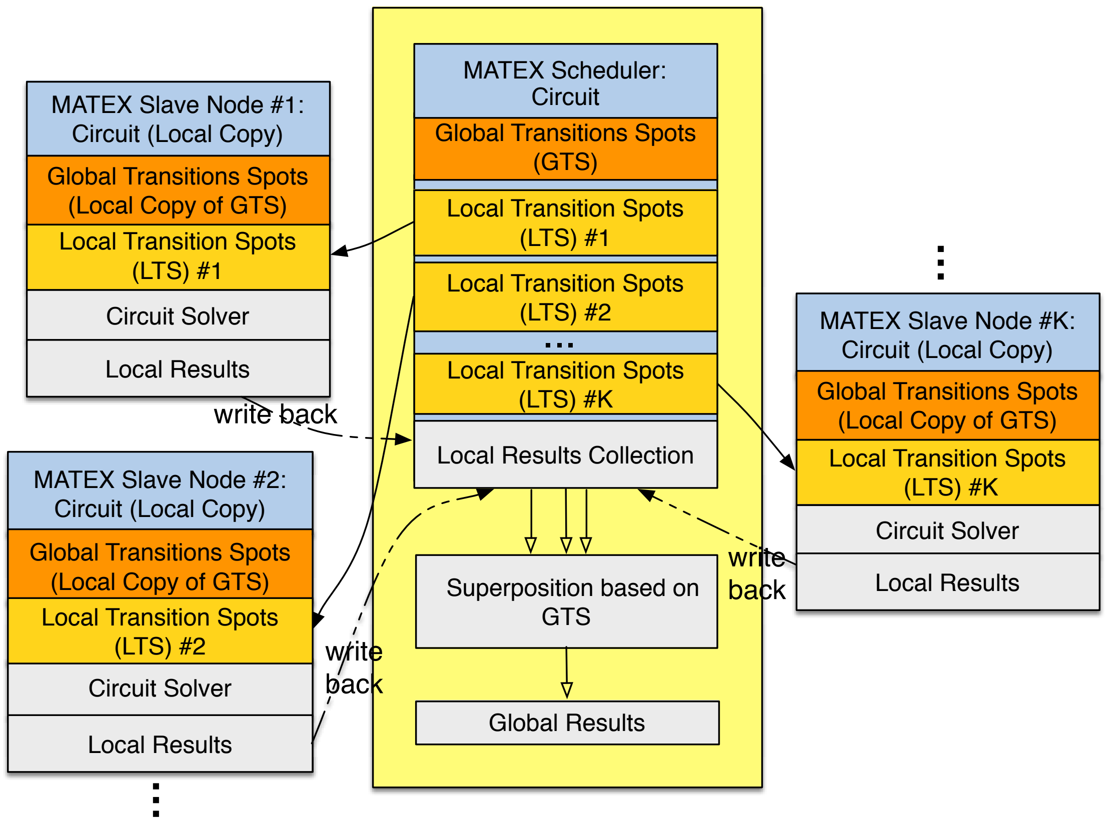

MATEX Scheduler: Circuit
Global Transitions Spots (GTS)
Local Transition Spots (LTS) #1
Local Transition Spots (LTS) #2
...
Local Transition Spots (LTS) #K
Local Results Collection
Superposition based on GTS
Global Results
MATEX Slave Node #1: Circuit (Local Copy)
Global Transitions Spots (Local Copy of GTS)
Local Transition Spots (LTS) #1
Circuit Solver
Local Results
write back
MATEX Slave Node #2: Circuit (Local Copy)
Global Transitions Spots (Local Copy of GTS)
Local Transition Spots (LTS) #2
Circuit Solver
Local Results
write back
MATEX Slave Node #K: Circuit (Local Copy)
Global Transitions Spots (Local Copy of GTS)
Local Transition Spots (LTS) #K
Circuit Solver
Local Results
write back

Figure 4.5: DR-MATEX: The distributed MATEX framework using R-MATEX circuit solver.

4.4.5 Runtime Analysis of MATEX PDN Solver

Suppose we have the dimension of Krylov subspace basis m on average for each time step and one pair of forward and backward substitutions consumes runtime T_{bs}. The total time of serial parts is T_{serial}, which includes matrix factorizations, result collection, etc. For $x(t+h)$, the evaluation of matrix exponential with e^{hH_m} is T_H, which is in proportion to the time complexity $O(m^3)$. Besides, we need extra T_e to form $x(t+h)$, which is proportional to $O(nm^2)$ by $\beta V_m e^{hH_m} e_1$.

Given K points of GTS, without decomposition of input sources, the runtime is

$$KmT_{bs} + K(T_H + T_e) + T_{serial}. \tag{4.11}$$

After dividing the input transitions and sending to enough computing nodes, we have k points of LTS for each node based on feature extraction and grouping (e.g., $k = 4$ for one "bump" shape feature). The total computation runtime is

$$kmT_{bs} + K(T_H + T_e) + T_{serial}, \tag{4.12}$$

where $K(T_H + T_e)$ contains the portion of computing *Snapshot* in DR-MATEX mode. The speedup of DR-MATEX over single MATEX is

$$\text{Speedup} = \frac{KmT_{bs} + K(T_H + T_e) + T_{serial}}{kmT_{bs} + K(T_H + T_e) + T_{serial}}. \tag{4.13}$$

For R-MATEX, we have small m. Besides, T_{bs} is relatively larger than $(T_H + T_e)$ in our targeted problem. Therefore, the most dominating part is the KmT_{bs} in Eq. (4.11). We can always decompose input source transitions, and make k smaller than K.

In contrast, suppose the traditional method with fixed step size has N steps for the entire simulation, the runtime is

$$NT_{bs} + T_{serial}.$$

Then, the speedup of distributed DR-MATEX over the traditional method is

$$\text{Speedup}' = \frac{NT_{bs} + T_{serial}}{kmT_{bs} + K(T_H + T_e) + T_{serial}}. \tag{4.14}$$

Note that, when the minimum distance among input source breakpoints decreases, large time span or many cycles is required to simulate PDNs, the schemes with such uniform step size would degrade runtime performance furthermore due to the increase of N. In contrast, in MATEX PDN solver, K is not so sensitive to such constraints. Besides,

k can be maintained in a small number based on the decomposition strategy. Therefore, the speedups of our proposed methods tend to be larger when the simulation requirements become harsher.

4.5 Numerical Results

We implement all the algorithms in MATLAB R2014b and use UMFPACK package for LU factorization. First, we compare I-MATEX, R-MATEX and TR in order to show our runtime improvements in single machine framework in Table 4.2. Second, we show our distributed framework DR-MATEX achieves large speedups in Table 4.3. The experiments are conducted on the server with Intel(R) Xeon (R) E5-2640 v3 2.60GHz processor and 125GB memory.

4.5.1 Performance of I-MATEX and R-MATEX in Sec. 4.3.1

We compare our proposed I-MATEX and R-MATEX against the popular TR-FTS on the IBM power grid benchmarks [64]. Among the current sources, the smallest interval between two breakpoints is $h_{upper} = 10ps$, which puts the upper limit of the TR's step size. All of these cases have very large numbers of input current sources. Table 4.1 shows the details of each benchmark circuit of which size ranges from 54K up to 3.2M. The simulation time is $10ns$. From ibmpg1t to ibmpg6t, TR uses fixed step size in $10ps$. We also change the IBM power grid benchmark to make the smallest distance among breakpoints $1ps$ by interleaving input sources' breakpoints (similar as Fig. 4.1). Therefore, the fixed step size method can only use at most $1ps$. The names of those benchmarks are ibmpg1t_new, ibmpg2t_new, ibmpg3t_new, ibmpg4t_new, ibmpg5t_new and ibmpg6t_new.

After DC analysis in TR-FTS, we LU factorize matrix once for the later transient

Table 4.1: Specifications of IBM power grid benchmarks.

Design	#R	#C	#L	#I	#V	#Nodes
ibmpg1t	41K	11K	277	11K	14K	54K
ibmpg2t	245K	37K	330	37K	330	165K
ibmpg3t	1.6M	201K	955	201K	955	1.0M
ibmpg4t	1.8M	266K	962	266K	962	1.2M
ibmpg5t	1.6M	473K	277	473K	539K	2.1M
ibmpg6t	2.4M	761K	381	761K	836K	3.2M

simulation, which only contains time stepping. Actually, multiple factorized matrices can be deployed [10,72]. We can choose one of them during the stepping. The problem is the memory and runtime overhead for the multiple matrix factorizations. Another point is if large time step h' is chosen, the standard low order scheme cannot maintain the accuracy.

Experiment is conducted on a single computing node. In Table 4.2, we record the total simulation runtime **Total(s)**, which includes the processes of DC and transient simulation, but excludes the non-numerical computation before DC, e.g., netlist parsing and matrix stamping. We also record the part of transient simulation **Tran(s)**, excluding DC analysis and LU decompositions. The speedup of I-MATEX is not as large as R-MATEX, because I-MATEX with a large spectrum of A generates large dimension m of Krylov subspace. Meanwhile, the step size is not large enough to let it fully harvest the gain from time marching with stepping. In contrast, R-MATEX needs small dimension numbers m of rational Krylov subspace, which ranges from 2 to 8 in those cases. Therefore, they can benefit from large time stepping, shown as SPDP^{r}_{tr}. For ibmpg4t, R-MATEX achieves maximum speedup resulted from the relatively small number of breakpoints in that benchmark, which is around 44 points, while the majority of others have over 140 points.

In Table 4.2, our single mode R-MATEX achieves the average speedup $5\times$ over TR-FTS. Note the average speedup number of single mode R-MATEX over TR-FTS for the original IBM benchmark (ibmpg1t$\sim$ibmpg6t) is less than the speedup of the new test cases (ibmpg1t_new$\sim$ibmpg6t_new). As we mentioned before, ibmpg1t_new$\sim$ibmpg6t_new have harsher input constraints, making the available step size only $1ps$. Therefore, the adaptive stepping by R-MATEX is more beneficial to the runtime performance in ibmpg1t_new$\sim$ibmpg6t_new than ibmpg1t$\sim$ibmpg6t.

4.5.2 Performance of DR-MATEX in Sec. 4.4

We test our distributed DR-MATEX in the following experiments with the same IBM power grid benchmarks. These cases have many input transitions (GTS) that limit step sizes of R-MATEX. We divide the region before the computation of simulation. We decompose the input sources by the approach discussed in Sec. 4.4.3 and obtain much fewer transitions of LTS for computing nodes. The original input source numbers are over ten thousand in the benchmarks. However, based on "bump" feature (as shown in Fig. 4.4), we obtain a fairly small numbers for each computing node, which is shown as *Group #* in Table 4.3. (Now, the fact that hundred machines to process in parallel is quite normal [73, 74] in the industry.) We pre-compute GTS and LTS groups and assign sub-tasks to corresponding nodes[5]. MATEX scheduler is only responsible for simple superposition calculation at the end of simulation. Since the slave nodes are in charge of all the computing procedures (Fig. 4.5) for the computation of their own transient simulation tasks, and have no communications with others, our framework falls into the category of *Embarrassingly Parallelism* model. We can easily emulate the multiple-node environment. We simulate each group using the command "matlab -singleCompThread" in our server. We record the runtime numbers for each process (slave nodes) and report the maximum runtime as the total runtime "Total(s)" of DR-MATEX in Table 4.3. We also record "pure transient simulation" as "Tran(s)", which is the maximum runtime of

[5]Based on the feature of input sources available, the preprocessing is very efficient, which takes linear time complexity to obtain GTS, LTS and separates the sources into different groups.

Table 4.2: Performance comparisons (single computing node): TR-FTS, I-MATEX, and R-MATEX.

Design	DC(s)	TR-FTS		I-MATEX				R-MATEX				Speedups	
		Tran(s)	Total(s)	m_I	Tran(s)	Total(s)	Df(uV)	m_R	Tran(s)	Total(s)	Df(uV)	SPDP^r_{tr}	SPDP^r_i
ibmpg1t	0.2	5.7	6.00	30	28.8	28.9	58\9.8	5	10.1	10.3	45\6.8	0.6×	2.9×
ibmpg2t	0.8	40.0	41.9	28	130.0	130.9	92\10.5	5	35.6	37.4	45\6.8	1.1×	3.7×
ibmpg3t	16.4	263.2	295.0	29	1102.5	1115.1	95\20.4	5	275.5	301.0	95\18.5	1.0×	4.0×
ibmpg4t	13.5	460.8	501.9	29	433.8	458.2	101\39.3	5	200.5	239.1	99\34.2	2.3×	2.2×
ibmpg5t	9.0	476.6	498.0	30	1934.4	1944.5	29\5.6	5	383.1	401.9	29\4.4	1.2×	5.0×
ibmpg6t	15.3	716.0	749.1	25	2698.9	2713.7	39\8.6	5	773.5	800.5	33\5.6	0.9×	3.5×
ibmpg1t_new	0.2	51.3	51.7	30	27.2	27.4	58\9.8	5	11.7	12.1	53\6.9	4.4×	2.3×
ibmpg2t_new	0.9	431.4	433.5	28	114.9	115.7	49\10.5	5	43.3	44.9	33\5.6	10.0×	2.7×
ibmpg3t_new	16.3	3716.5	3749.0	29	1219.3	1232.6	95\20.4	5	481.7	508.2	95\18.9	7.7×	2.5×
ibmpg4t_new	18.3	5044.6	5085.3	29	753.5	776.4	101\39.3	6	350.9	387.2	99\34.2	14.4×	2.1×
ibmpg5t_new	10.5	5065.9	5110.1	30	2494.0	2504.7	30\5.6	5	746.2	766.4	30\4.4	6.8×	3.3×
ibmpg6t_new	13.1	7015.3	7059.7	25	3647.9	3663.1	39\8.6	6	895.1	923.1	33\7.3	7.8×	4.1×
Average	—	—	—	—	—	—	65\15.7	—	—	—	57\12.8	5×	3×

DC(s): Runtime of DC analysis (seconds); m_I: The maximum m of Krylov subspace in I-MATEX. **Tran(s)**: Runtime of transient simulation after DC (seconds), excluding the matrix factorization runtime; **Total(s)**: Runtime of overall transient simulation (seconds); **Df(uV)**: Maximum and average voltage differences compared to provided solutions (uV); m_R: The maximum m of Krylov subspace in R-MATEX $\mathbf{SPDP}^r_{tr}$: Speedup of R-MATEX over TR-FTS with respect to **Tran(s)**; $\mathbf{SPDP}^r_i$: Speedup of R-MATEX over I-MATEX with respect to **Tran(s)**.

the counterparts among all computing nodes.

For TR-FTS, we use $h = 10ps$, so there are 1,000 pairs of forward and backward substitutions during the process of pure transient simulation for ibmpg1t∼ibmpg6t; We use $h = 1ps$ for ibmpg1t_new∼ibmpg6t_new. Therefore, we have 10,000 pairs of forward and backward substitutions for stepping. In DR-MATEX, the circuit solver uses R-MATEX with $\gamma = 10^{-10}$, which is set to sit among the order of varied time steps during the simulation (since Sec. 3.3.4 discusses the insensitivity of γ around the step size of interests). TR-FTS is not distributed because it has no gain by dividing the current source as we do for the DR-MATEX. TR-FTS cannot avoid the repeated pairs of forward and backward substitutions. Besides, adaptive stepping for TR-FTS only degrades the performance, since the process requires extra matrix factorizations.

In Table 4.3, our distributed mode gains up to 98× for the pure transient computing. The average peak dimension m of rational Krylov subspace is 7. The memory overhead ratio for each node (around 1.6× over TR-FTS on average) is slightly larger, which is worthwhile with respect to the large runtime improvement. With the huge reduction of runtime for Krylov subspace generations, the serial parts, including LU and DC, play more dominant roles in DR-MATEX, which can be further improved using advance matrix solvers, such as [30].

4.6 Summary

In this chapter, we propose an efficient framework MATEX for accurate PDN time-domain simulation based on the exponential integration scheme. For the PDN simulation, our time integration scheme can perform adaptive time stepping without repeating matrix factorizations, which cannot be achieved by traditional methods using implicit numerical integration with fixed time-step scheme. Compared to the commonly adopted framework TR with fixed time step (TR-FTS), our single mode framework (R-MATEX) gains runtime speedup up to around 15$\times$. We also show that the distributed MATEX framework (DR-MATEX) leverages the superposition property of linear system and decomposes the task based on the feature of input sources, so that we reduce chances of Krylov subspace generations for each node. We achieve runtime improvement up to 98$\times$ speedup.

Chapter 4, in part, is a reprint of the material as it appears in “Simulation Algorithms with Exponential Integration for Time-Domain Analysis of Large-Scale Power Delivery Networks” by Hao Zhuang, Wenjian Yu, Shih-Hung Weng, Ilgweon Kang, Jeng-Hau Lin, Xiang Zhang, Ryan Coutts, and Chung-Kuan Cheng in *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems*. The chapter also contains the content from “Power Grid Simulation using Matrix Exponential Method with Rational Krylov Subspaces” by Hao Zhuang, Shih-Hung Weng, and Chung-Kuan Cheng in *Proceedings of IEEE International Conference on ASIC 2013*, and “MATEX: A Distributed Framework for Transient Simulation of Power Distribution Networks” by Hao Zhuang, Shih-Hung Weng, Jeng-Hau Lin, and Chung-Kuan Cheng in *Proceedings of IEEE/ACM Design Automation Conference 2014*. The thesis author was the primary investigator and author of the papers.

Table 4.3: The performance of DR-MATEX (Distributed R-MATEX).

Design	DR-MATEX					Speedups		Peak	Mem. Ratio
	Group #	Tran(s)	Total(s)	Max Df.(V)	Avg Df.(V)	SPDP$_{tr}$	SPDP$_r$	m	over TR-FTS
ibmpg1t	100	1.4	1.9	5.3e-5	8.6e-6	4.0×	7.1×	6	1.9
ibmpg2t	100	8.9	11.4	4.6e-5	8.6e-6	4.5×	4.0×	7	1.9
ibmpg3t	100	91.7	129.9	9.6e-5	19.7e-6	2.9×	4.4×	6	1.5
ibmpg4t	15	52.3	112.2	9.9e-5	27.9e-6	8.8×	3.8×	8	1.4
ibmpg5t	100	148.4	178.9	9.0e-5	1.1e-6	3.2×	2.6×	7	1.5
ibmpg6t	100	189.9	234.2	3.4e-5	7.2e-6	3.8×	4.1×	7	1.5
ibmpg1t_new	100	2.4	2.8	5.3e-5	8.6e-6	21.8×	5.0×	6	1.9
ibmpg2t_new	100	5.6	7.0	4.6e-5	8.6e-6	61.6×	6.2×	7	1.9
ibmpg3t_new	100	103.0	140.9	9.8e-5	19.9e-6	25.6×	3.3×	7	1.5
ibmpg4t_new	15	51.5	108.4	9.9e-5	27.6e-6	98.0×	6.8×	8	1.4
ibmpg5t_new	100	185.6	227.8	9.9e-5	2.2e-6	27.3×	4.0×	7	1.5
ibmpg6t_new	100	274.8	317.7	3.4e-5	7.1e-6	25.5×	3.3×	7	1.5
Average	—	—	—	7.1e-5	12.3e-6	26×	5×	6.7	1.6

Group #: Group number of the testcases. This number represents the total number of simulation sub-tasks for the design; **Tran(s)**: Runtime of transient simulation after DC (seconds); **Total(s)**: Runtime of overall transient simulation (seconds); **Max. Df.(V)** and **Avg. Df.(V)**: maximum and average differences compared to the solutions of all output nodes provided by IBM power grid benchmarks. **SPDP$_{tr}$**: Speedup over TR-FTS's **Tran(s)** in Table 4.2; **SPDP$_r$**: Speedup over R-MATEX's **Tran(s)** in Table 4.2; **Peak** m: the peak dimension used in DR-MATEX for MEVP; **Mem. Ratio over TR-FTS**: The peak memory comparison between the maximum memory consumption of DR-MATEX over TR-FTS in Table 4.2.

Chapter 5

Exponential Integration for Nonlinear Dynamical Systems

5.1 Motivation

Large-scale circuits with nonlinear transistors form nonlinear dynamical systems. In the conventional implicit numerical integration for solving such systems, Newton-Raphson (NR) iteration is applied to obtain the converged solution of the nonlinear function at each time step (Section 2.4). The circuit simulator needs to linearize and solve the system for each NR iteration, where direct solvers [28, 30] are usually applied because of their robustness and ease of use. However, it is known that direct solvers, e.g., LU decomposition, have super-linear computational complexities and very expensive to simulate large-scale and strongly coupled circuit systems. For instance, the cost can approach the worst case $O(n^3)$ [75,76]. In other words, the implicit integration algorithms are more computationally expensive per step than the explicit integration approach due to the requirement of solving linear matrix system. The widely application of implicit integration is because that the low order explicit scheme is numerically unstable, which uses extremely small time step size when the dynamical system is stiff [3,7]. However, the computation from implicit formulation itself does not actually improve the accuracy of solution, but ensure the stability [77] for the process of numerical integration. Therefore, leveraging the explicit formulation and retaining accuracy is still a direction of the circuit simulation research. Researchers still devise efficient algorithms that can reduce or remove expensive LU operations or NR iterations [12,39,77–79] in order to scale up the capability of circuit simulators.

In this study, we devise the framework EI to utilize explicit exponential integration to simulate nonlinear systems. The stability is ensured by the exponential integrators. The features of proposed method in chapter are listed as follows:

- **EI removes Newton-Raphson iteration because of its explicit formulation.** EI takes only one LU decomposition per time step, while conventional methods, e.g., BE with NR (BENR), require at least two times of LU to verify the convergence. Note that the reason EI contains no Newton-Raphson iteration is that it treats dynamical system in a fully explicit way, which was also stated by Luan and Ostermann in the work [80]. The stability of the explicit formulation is preserved by the high-order approximation of exponential operator [16,19,35]. The convergence of the solution is checked and refined by the compensation iteration with system residue and the KCL/KVL condition, which is proposed in this chapter.

- **EI adopts invert Krylov subspace method as the computation building block for matrix function computation.** Invert Krylov subspace improves the convergence rate for matrix exponential and vector product compared to previous nonlinear circuit simulation via matrix exponentials and standard Krylov subspace [14]. Besides, this approach also removes the regularization process for possibly singular matrix C [43], which is impractical for large designs. Therefore, this building block enables the application of exponential integration for large-scale nonlinear circuits.

- **EI has better properties in flexible time stepping and higher order accuracy.** EI does not need to repeat LU when it adjusts the length of time steps for step and error controls. It is because the explicit formulation and (time-step) scaling-invariant property of Krylov subspace [14, 16]. On the contrary, the low order approximation schemes force time step embedded in the linear matrix and conduct matrix factorization. Once the time step is adjusted, LU decomposition is unavoidable in order to solve the new linearized system.

- **EI is suited to handle strongly coupled post-layout circuits.** Invert Krylov subspace strategy removes capacitance/inductance matrix C from matrix factorization processes. Building the subspace only needs to factorize G, which is much sparser and simpler than C in the strongly coupled post-layout circuits. In contrast, conventional implicit methods always require LU decomposition of the combinations of G and C.

5.2 Exponential Integration for Nonlinear Dynamical Systems

For a generalized nonlinear differential equation system

$$\frac{dx}{dt} = N(x,t) \tag{5.1}$$

Explicit exponential integration method is applied to solve the system, which has been investigated in [17, 39, 80]. The idea is that the vector field $N(x,t)$ is linearized along the numerical solution x_k of Eq. (5.1), which leads to a semilinear problem,

$$\frac{dx}{dt} = J_k x + u(x,t), \tag{5.2}$$

where the Jacobian matrix

$$J_k = J(x_k) = \frac{\partial N}{\partial x}\Big|_{x=x_k}$$

and

$$u(x,t) = N(x,t) - J_k x.$$

Jacobian matrix J_k of $u(x_k,t)$ diminishes at the state x_k.

The solution x_{k+1} at time $t_{k+1} = t_k + h$ via exponential integration is cast into

$$\begin{aligned} x_{k+1} &= \phi_0(hJ_k)x_k + h\phi_1(hJ_k)u(x_k,t_k) + h^2\phi_2(hJ_k)\frac{\partial N}{\partial t}\Big|_{x=x_k,t=t_k} + O(h^3) \\ &= x_k + h\phi_1(hJ_k)N(x_k,t_k) + h^2\phi_2(hJ_k)\frac{\partial N}{\partial t}\Big|_{x=x_k,t=t_k} + O(h^3) \end{aligned} \tag{5.3}$$

When $s > 0$, matrix function $\phi_s(hJ)$ follows

$$hJ\phi_s(hJ) = \phi_{s-1}(hJ) - \frac{1}{(s-1)!}I. \tag{5.4}$$

When $s = 0$, we have

$$\phi_0(hJ) = e^{hJ}, \tag{5.5}$$

And it is called matrix exponential function.

To transfer the above equations into the dynamical circuit system, we start from the following equation,

$$C(x_k)\frac{dx}{dt}+G(x_k)x = Bu(t_{k+1})+F(x_k)-\varepsilon(x_k,x_{k+1}), \tag{5.6}$$

where

$$\begin{aligned}\varepsilon(x_k,x_{k+1}) &= (C(x_{k+1})-C(x_k))\frac{dx}{dt} \\ &+ (G(x_{k+1})-G(x_k))x-(F(x_{k+1})-F(x_k))\end{aligned} \tag{5.7}$$

is the nonlinear function , which serves as a compensation component. A corresponding approximation theory will be discussed in Section 5.2.1.

Based on Eq. (5.3), the explicit formulation to compute x_{k+1} is written as

$$x_{k+1} = x_k + h\phi_1(hJ_k)g_k + h^2\phi_2(hJ_k)b_k \tag{5.8}$$

for the formulation of circuit simulation, where

$$J_k = J(x_k) = -C(x_k)^{-1}G(x_k),$$

$$g_k = g(x_k,u,t_k) = J_k x_k + C^{-1}(x_k)\left(F(x_k)+Bu(t_k)\right),$$

and

$$b_k = b(x_k,u,t_k) = \frac{\partial}{\partial t}g(x_k,u,t_k) = C^{-1}(x_k)B\frac{u(t_{k+1})-u(t_k)}{h},$$

where u or $u(t)$ is a piecewise-linear input vector from the external input sources. Furthermore, the fidelity of this numerical solution needs to be checked due to the nonlinearity of dynamical system.

5.2.1 Residue Checking for Nonlinear Dynamical Systems

The residue vector $r(x,t)$ is defined as follows to check the convergence by the KCL/KVL condition,

$$r(x,t) = C(x)\frac{dx}{dt} + G(x)x - F(x) - Bu(t). \tag{5.9}$$

Therefore, for the numerical solution x_{k+1}, the residue $r_{k+1} = r(x_{k+1}, t_{k+1})$ is used to check the solution at time $t_{k+1} = t_k + h$,

$$r_{k+1} = C(x_{k+1})\frac{dx}{dt}\bigg|_{x=x_{k+1}} + G(x_{k+1})x_{k+1} - F(x_{k+1}) - Bu(t_{k+1}), \tag{5.10}$$

This residue vector r_{k+1} should be small enough (e.g., smaller than a threshold Err) in order to meet KCL/KVL approximately.[1]

Since EI uses explicit formulation, in order to avoid the undershoot or overshoot from the Jacobian matrix evaluated at state x_k, the compensation vector $\varepsilon_{k+1} = \varepsilon(x_k, x_{k+1})$ is approximated by the following series

$$\varepsilon_{k+1} := \sum_{i=0}^{k} \varepsilon_{k+1}^{(i)}, \tag{5.11}$$

which is used to "correct" the mismatch from the direction projected by the system $G(x_k), C(x_k)$ at state x_k and time t_k. The system equation from Eq. (5.6) is approximated

[1]In this chapter, we use L_∞ as the error metric, $|r_{k+1}|_\infty < Err$, where Err is a pre-defined threshold.

to Eq. (5.6) by

$$\begin{aligned} C(x_k)\frac{dx}{dt}+G(x_k)x &= Bu(t_{k+1})+F(x_k)-\varepsilon_{k+1} \qquad (5.12)\\ &= Bu(t_{k+1})+F(x_k)-\sum_{i=0}^{k}\varepsilon_{k+1}^{(i)}, \end{aligned}$$

The assumption of above approximation theory is that the nonlinearity $u(x,t)$ of Eq. (5.2) can be approximated well by the Taylor expansion series

$$u(x,t)=u_1+tu_2+\cdots+\frac{t^{p-1}}{(p-1)!}u_p, \qquad (5.13)$$

where u_i is i-th expansion vector for the nonlinear function $u(x,t)$ [81]. The differential equation system Eq. (5.1) becomes

$$\begin{aligned} \frac{dx}{dt} &= Jx+u(x,t)\\ &= Jx+u_1+tu_2+\cdots+\frac{t^{p-1}}{(p-1)!}u_p\\ &= Jx+u_1+tu_2+O(t^2) \qquad (5.14) \end{aligned}$$

The solution of Eq. (5.14) is

$$\begin{aligned} x_{k+1} &= \phi_0(tJ)x_k+t\phi_1(tJ)u_1+t^2\phi_2(tJ)u_2+\cdots+t^p\phi_p(tJ)u_p\\ &= \phi_0(tJ)x_k+t\phi_1(tJ)u_1+t^2\phi_2(hJ)u_2+O(t^3) \qquad (5.15) \end{aligned}$$

The derivative is

$$\begin{aligned} \left.\frac{dx}{dt}\right|_{x=x_{k+1}} &= J\phi_0(tJ)x_k+\phi_0(tJ)u_1+t\phi_1(tJ)u_2+\cdots+t^{p-1}\phi_{p-1}(tJ)u_p\\ &= J\phi_0(tJ)x_k+\phi_0(tJ)u_1+t\phi_1(tJ)u_2+O(t^2) \qquad (5.16) \end{aligned}$$

by applying Eq. (5.4) and

$$\frac{d}{dt}(t^{s+1}\phi_{s+1}(tJ)) = t^s\phi_s(tJ). \tag{5.17}$$

recursively.

We use induction to prove Eq. (5.17),

Proof.

When $s = 0$,

$$\frac{d}{dt}(t\phi_1(tJ)) = \frac{d}{dt}(t\frac{e^{tJ}-I}{tJ}) = \frac{d}{dt}(\frac{e^{tJ}-I}{J}) = e^{tJ} = \phi_0(tJ). \tag{5.18}$$

When $s > 1$,

$$\frac{d}{dt}(t^s\phi_s(tJ)) = t^{s-1}\phi_{s-1}(tJ). \tag{5.19}$$

We have

$$\begin{aligned} \frac{d}{dt}(t^{s+1}\phi_{s+1}(tJ)) &= \frac{d}{dt}(t^s(t\phi_{s+1}(tJ)) = \frac{d}{dt}(t^s(\frac{\phi_s(tJ)-\frac{1}{s!}}{J})) \\ &= \frac{d}{dt}\frac{t^s\phi_s(tJ)-t^s\frac{1}{s!}}{J} \\ &= \frac{t^{s-1}\phi_{s-1}(tJ)-t^{s-1}\frac{1}{(s-1)!}}{J} \\ &= t^s\frac{\phi_{s-1}(tJ)-\frac{1}{(s-1)!}}{tJ} = t^s\phi_s(tJ). \end{aligned}$$

□

Solution Eq. (5.15) and derivative Eq. (5.16) fit

$$\frac{dx}{dt} = Jx + u(x,t).$$

Proof.

$$\begin{aligned}
Jx+u(x,t) &= J(\phi_0(tJ)x+t\phi_1(tJ)u_1+t^2\phi_2(tJ)u_2+\cdots+t^p\phi_p(tJ)u_p) \\
&+ (u_1+tu_2+\cdots+\frac{t^{p-1}}{(p-1)!}u_p) \\
&= J\phi_0(tJ)x+(tJ\phi_1(tJ)+I)u_1 \\
&+ t(tJ\phi_2(tJ)+I)u_2+t^2(tJ\phi_3(tJ)+\frac{I}{2})u_3 \\
&+ \cdots+t^{p-1}(tJ\phi_p(tJ)+\frac{I}{(p-1)!})u_p \\
&= J\phi_0(tJ)x+\phi_0(tJ)u_1+t\phi_1(tJ)u_2+\cdots+t^{p-1}\phi_{p-1}(tJ)u_p \\
&= \frac{dx}{dt}
\end{aligned}$$

□

From above, we can have a nice property that when we have a solution component $t^s\phi_s(tJ)u_s$, its derivative can be computed easily via

$$\frac{d}{dt}(t^s\phi_s(tJ)u_s) = t^{s-1}\phi_{s-1}(tJ)u_s.$$

For the compensation iteration, it is started when r_{k+1} in Eq. (5.10) is larger than the error threshold Err. In this study, only the second term of Eq. (5.14) is used to model the missing dynamics and correct the solution x_{k+1}.[2]

The approximation formula is

$$u(x,t) = u_1+hu_2+O(h^2) \approx u_1+hu_2. \tag{5.20}$$

[2]The higher order terms are with higher order ϕ_i, the efficient computation of those high order function ϕ_i is non-trivial [82].

where we approximate u_2 in the setting of our circuit simulation formulation by

$$u_2 = \frac{\partial}{\partial t}\sum_{i=0}^{k} C^{-1}(x_k) r_{k+1}^{(i)},$$

Therefore, we have

$$\varepsilon_{k+1}^{(i)} \approx h^2 \phi_2(hJ_k)\frac{\partial}{\partial t} C^{-1}(x_k) r_{k+1}^{(i)}.$$

where the superscript i represents i-th iteration during compensation iteration. The solution is

$$\begin{aligned}
x_{k+1} &= x_k + h\phi_1(hJ_k)g_k + h^2\phi_2(hJ_k)b_k - h^2\phi_2(hJ_k)\frac{\partial}{\partial t}\sum_{i=0}^{k} C^{-1}(x_k) r_{k+1}^{(i)} \qquad (5.21)\\
&= x_k + h\phi_1(hJ_k)g_k + h^2\phi_2(hJ_k)b_k - \sum_{i=0}^{k} h^2\phi_2(hJ_k)\frac{C^{-1}r_{k+1}^{(i)}}{h},\\
&= x_k + (\phi_0(hJ_k)-I)\widetilde{g}_k + h(\phi_1(hJ_k)-I)\widetilde{b}_k - h\sum_{i=0}^{k}(\phi_1(hJ_k)-I)\widetilde{r}_{k+1}^{(i)},
\end{aligned}$$

where

$$\begin{aligned}
\widetilde{g}_k &= J_k^{-1}(x_k)g_k\\
&= -G^{-1}(x_k)C(x_k)(J_k x_k + C^{-1}(x_k)(F(x_k)+Bu(t_k)))\\
&= x_k - G^{-1}(x_k)(F(x_k)+Bu(t_k)), \qquad (5.22)
\end{aligned}$$

$$\begin{aligned}
\widetilde{b}_k &= J_k^{-1}(x_k)b_k\\
&= -G^{-1}(x_k)C(x_k)C^{-1}(x_k)B\frac{u(t_k+h)-u(t_k)}{h}\\
&= -G^{-1}(x_k)B\frac{u(t_k+h)-u(t_k)}{h}, \qquad (5.23)
\end{aligned}$$

and

$$\begin{aligned}\widetilde{r}_{k+1}^{(i)} &= J_k^{-1}C^{-1}(x_k)\frac{r_{k+1}^{(i)}}{h} \\ &= -G^{-1}(x_k)C(x_k)C^{-1}(x_k)\frac{r_{k+1}^{(i)}}{h} \\ &= -G^{-1}(x_k)\frac{r_{k+1}^{(i)}}{h}. \end{aligned} \tag{5.24}$$

The above three terms are derived to avoid the inversion or factorization of the possibly singular capacitance/inductance matrix C.

The derivative at x_{k+1} is computed by

$$\left.\frac{dx}{dt}\right|_{x=x_{k+1}} = hJ_k\phi_0(hJ_k)\widetilde{g}_k + \phi_0(hJ_k)\widetilde{b}_k - \sum_{i=0}^{k}\phi_0(hJ_k)\widetilde{r}_{k+1}^{(i)}. \tag{5.25}$$

5.2.2 Compensation Iteration for KCL/KVL

At the initial stage during the computation flow, we have

$$x_{k+1}^{(0)} = x_k + (\phi_0(hJ_k) - I)\widetilde{g}_k + h(\phi_1(hJ_k) - I)\widetilde{b}_k, \tag{5.26}$$

where the superscript 0 is the solution before the compensation iterations. Since how much offset projected by $C(x_k)$ and $G(x_k)$ is unavailable at this moment, the system residue $r_{k+1} = 0$. After obtaining $x_{k+1}^{(0)}$, residue $r_{k+1}^{(0)}$ is available by Eq. (5.10).

If $r_{k+1}^{(0)} < Err$, the solution is accurate enough. Otherwise, we need the compensation iteration. The solution of the dynamical system can be refined through

$$x_{k+1}^{(i+1)} = x_{k+1}^{(i)} - h(\phi_1(hJ_k) - I)\widetilde{r}_{k+1}^{(i)} \tag{5.27}$$

And the resulting residue/compensation vector for $x_{k+1}^{(i+1)}$ is

$$r_{k+1}^{(i+1)} = C(x_{k+1}^{(i+1)})\frac{dx}{dt}\Big|_{x=x_{k+1}^{(i+1)}} + G(x_{k+1}^{(i+1)})x_{k+1}^{(i+1)} - F(x_{k+1}^{(i+1)}) - Bu(t_{k+1}) \tag{5.28}$$

The process is repeated until $r_{k+1}^{(i+1)}$ is small enough. However, when the iteration number exceeds a pre-defined threshold $Iter$, we shrink the time step h, The reason is that the term $O(h^2)$ in our nonlinearity approximation Eq. (5.20) actually cannot be ignored. Intuitively, the step size reduction is one straightforward way to contain the nonlinearities.

5.3 Invert Krylov Subspace for ϕ-Function

Algorithm 6 *InvKrylovMatEx* summarizes the way to compute the product of matrix function ϕ_s and vector via invert Krylov subspace basis. In Line 3 of Algorithm 6, we reuse factorized matrices from LU decomposition of G to solve the linear system $-Gx = b$, where $b = Cv_j$ is formed in j-th iteration. Algorithm 7 *ResidueCheck* is applied in Line 10 and serves as accuracy monitor, which checks the estimated error of the ϕ_i function and vector product by computed matrices subspace. If the condition in Line 2 of Algorithm 7 fails, it means the residue is too large. We need to increase the dimension of subspace in Algorithm 6 to reduce the error. Note that the implementation of ϕ_s computation can utilize the approach designed by Al-Mohy and Higham [83].

Using invert Krylov subspace, the solution Eq (5.26) is

$$x_{k+1}^{(0)} = x_k + ||\widetilde{g}_k||V_{m_0}(\phi_0(hH_{m_0}^{-1}) - I_{m0})e_{1_{m_0}} + h||\widetilde{b}_k||V_{m_1}(\phi_1(hH_{m_1}^{-1}) - I_{m_1})e_{1_{m_1}}, \tag{5.29}$$

where V_{m_i} and H_{m_i} are generated from Algorithm 6; m_0 and m_1 represent different dimensions for invert Krylov subspaces for $\widetilde{g}_k$ and $\widetilde{b}_k$, respectively; I_m is an $m \times m$

Algorithm 6: *InvKrylovMatEx* for $\phi_s(hJ)v$ function

```
    Input: $C, G, v, h, s$
    Output: $H_m, V_m$
1   $v_1 = \frac{v}{\|v\|}$;
2   for $j = 1 : m$ do
3       Solve $-Gw = Cv_j$ and obtain $w$;
4       for $i = 1 : j$ do
5           $h_{i,j} = w^\top v_i$;
6           $w = w - h_{i,j} v_i$;
7       end
8       $h_{j+1,j} = \|w\|$;
9       $v_{j+1} = \frac{w}{h_{j+1,j}}$;
10      if Algorithm 7 ResidueCheck(v, V, H, j, G, h, s) then
11          $m = j$;
12          break;
13      end
14  end
```

Algorithm 7: *ResidueCheck*

```
    Input: $v, V, H, m, G, h, s$
    Output: True or False
1   $r_s = ||v|||h^s G v_{m+1} h_{m+1,m} e_m^\top H_m^{-1} \phi_s(hH_m^{-1}) e_1|_\infty$;
2   if $r_s > Err$ then
3       Return $False$;
4   end
5   Return $True$;
```

identity matrix.

The corresponding derivative is

$$\left.\frac{dx}{dt}\right|_{x=x_{k+1}^{(0)}} = h||\widetilde{g}_k||V_{m_0}H^{-1}\phi_0(hH_{m_0}^{-1})e_{1_{m_0}} + ||\widetilde{b}_k||V_{m_1}H_{m_1}^{-1}\phi_0(hH_{m_1}^{-1})e_{1_{m_1}} \quad (5.30)$$

For the compensation process, we also need to build Krylov subspace for ϕ-function and residue vectors. For example, from i-th to $(i+1)$-th iteration, the compensation computation is

$$x_{k+1}^{(i+1)} = x_{k+1}^{(i)} - h||\widetilde{r}_{k+1}^{(i)}||V_{m^{(i)}}(\phi_1(hH_{m^{(i)}}^{-1}) - I_{m^{(i)}})e_{1_{m^{(i)}}} \tag{5.31}$$

where $x_{k+1}^{(i)}$ is from the solution at the last iteration. The corresponding derivative is

$$\left.\frac{dx}{dt}\right|_{x=x_{k+1}^{(i+1)}} = \left.\frac{dx}{dt}\right|_{x=x_{k+1}^{(i)}} - ||\widetilde{r}_{k+1}^{(i)}||V_{m^{(i)}}(h\phi_0(hH_{m^{(i)}}^{-1}))e_{1_{m^{(i)}}} \tag{5.32}$$

where $\left.\frac{dx}{dt}\right|_{x=x_{k+1}^{(i)}}$ is from the last iteration.

5.4 Overall Framework

The overall framework of EI is summarized in Algorithm 8. Line 5 is the only place we factorize G_k at each time step, where k represents the k-th step during the simulation, so a matrix G with superscript k means the matrix at t_k. The lines between Line 12 and Line 18 are the iteration for correction process. E_{rr} is the error budget. During the compensation iteration, the residue $r_{k+1}^{(i)}$ is computed based on the KCL/KVL from the whole systems, which means it checks the branch current and node voltage in order to preserve the fidelity of solution. When the residue $r_{k+1}^{(i)}$ is larger than E_{rr} but i is smaller than $Iters$, we use compensation iteration to refine the solution; When $r_{k+1}^{(i)}$ is larger than E_{rr} and i is larger than $Iters$, we think that the nonlinearity is too strong to converge under the constraint of KVL/KCL, so we need to shrink the time step in order to get a more closed linearized model. The algorithm shrinks the time step by μ in Line 19. When the iteration is small to converge, we can enlarge the time step h by α correspondingly.[3]

[3]The parameters in this paragraph, $Iters = 10$, $Iters_{small} = 4$, $\mu = 0.5$, $\alpha = 1.2$. The parameters in this paper are chosen empirically after several trial run. The parameter tunning is beyond the scope of this paper.

Algorithm 8: EI: Explicit Circuit Simulation with Exponential Integration Kernel

```
Input: Circuit netlist;
Output: Voltage/current solution vectors $x_k$ at $k = 0, \cdots, Step$ for time period $[0, T]$
1  Initialization phase: (a) Load the netlist; (b) Build linear matrices $C_l$, $G_l$, $B$; Set $t = 0; k = 0$;
2  $x(0) = x_k =$ DC_solution;
3  while $t \leq T$ do
4      Set $i = 0$; Derive nonlinear matrices and vectors $C(x_k), G(x_k), F(x_k)$, $\widetilde{g}_k, \widetilde{b}_k$ from device
       models at the state $x_k$ with linear matrices $C_l$ and $G_l$;
5      Perform $LU\_decompose(G(x_k))$ for all the following Algorithm 6 ;
6      Use Algorithm 6 $InvKrylovMatEx$ to compute $(H_{m_0}, V_{m_0})$ for the component with $\phi_0$
       and $\widetilde{g}_k$ in Eq. (5.26);
7      Use Algorithm 6 $InvKrylovMatEx$ to compute $(H_{m_1}, V_{m_1})$ for the component with $\phi_1$
       and $\widetilde{b}_k$ in Eq. (5.26);
8      while $True$ do
9          Compute solution $x_{k+1}^{(i)}$ as Eq. (5.29);
10         Compute derivative $\frac{dx}{dt}\Big|_{x=x_{k+1}^{(i)}}$ as Eq. (5.30);
11         Compute $r_{k+1}^{(i)}$ as Eq. (5.28);
12         while $|r_{k+1}^{(i)}|_\infty \geq E_{rr}$ and $i \leq Iters$ do
13             Use Algorithm 6 $InvKrylovMatEx$ to compute $(H_{m^{(i)}}, V_{m^{(i)}})$ for $\phi_1$ with
               residue vector $\widetilde{r}_{k+1}^{(i)}$;
14             Compute the solution $x_{k+1}^{(i+1)}$ as Eq. (5.31);
15             Update the derivative $\frac{dx}{dt}\Big|_{x=x_{k+1}^{(i+1)}}$ as Eq. (5.32);
16             Compute the residue vector $r_{k+1}^{(i+1)}$ as Eq. (5.28);
17             Increase the iteration number $i = i + 1$;
18         end
19         if $|r_{k+1}^{(i)}|_\infty \geq E_{rr}$ then
20             $i = 0$; $h = \mu h$;
21         end
22         else
23             $x(t+h) = x_{k+1}^{(i)}$; $t = t + h$; $k = k + 1$;
24             if $i \leq Iters_{small}$ then
25                 $h = min(\alpha h, h_{max})$;
                   /* $i$ is small, $h$ is increased by $\alpha > 1$ to accelerate the
                      process. $h_{max}$ is set for the maximum time step in
                      order to maintain the waveform resolution.          */
26             end
27             break; // Break the while loop.
28         end
29     end
30 end
```

Note that in terms of Krylov subspace generation, invert Krylov subspace method is suitable for the post-layout simulation with strong coupled C, because its matrix factorization target is only matrix G. Simulation of circuits with post-layout extraction or strong parasitics is very critical. The parasitic effects are sometimes ignored in fast circuit analysis. The semiconductor device modeling, even in the most advanced models, a lot of simplifications and approximations are utilized. Such approximation might lead to discrepancy between the actual and simulated results. Post-layout analysis with detailed capacitance extraction [2, 24, 26] is usually important to make sure the functionality of circuits after VLSI Place-and-Route.

We use a design **FreeCPU** [2] as an example to show post-extraction matrices[4] (Fig. 5.1). The sizes of all matrices are 11417×11417. The number *nnz* represents the total number of non-zeros in the matrix. Fig. 5.1 (a) shows non-zero entries distribute widely in the matrix extracted capacitance matrix C, which has the number $nnz = 62,815$. Fig. 5.1 (b) illustrates the extracted conductance matrix G. The number *nnz* of non-zero terms is $34,388$. We use LU to factorize[5] C, and obtain Fig. 5.1 (c), which is the lower triangular matrix L_C and (d) the upper triangular matrix U_C. The number *nnz* are $281,233$ and $281,171$, respectively. Fig. 5.1 (e) shows the matrix L_G and (f) the matrix U_G of $LU\_decompose(G)$. The number *nnz* are $23,049$ and $20,711$, respectively. Fig. 5.1 (g) plots the matrix $L_{C/h+G}$ and (h) matrix $U_{C/h+G}$ of $LU\_decompose(C/h+G)$. The number *nnz* are $521,380$ and $521,379$, respectively. For those extracted matrices, we observe that the conductance/resistance G contains less number of *nnz* than the capacitance/inductance matrix C.

Another important point is the distribution of non-zeros. We notice the bandwidth of G is much smaller than that of C based on the plot of the two matrices in Fig. 5.1 (a) and (b). The number of *nnz* and distribution pattern all play important roles in matrix

[4]Parasitics are extracted by industrial tool Synopsys Star-RCXT.
[5]MATLAB2013a UMFPACK

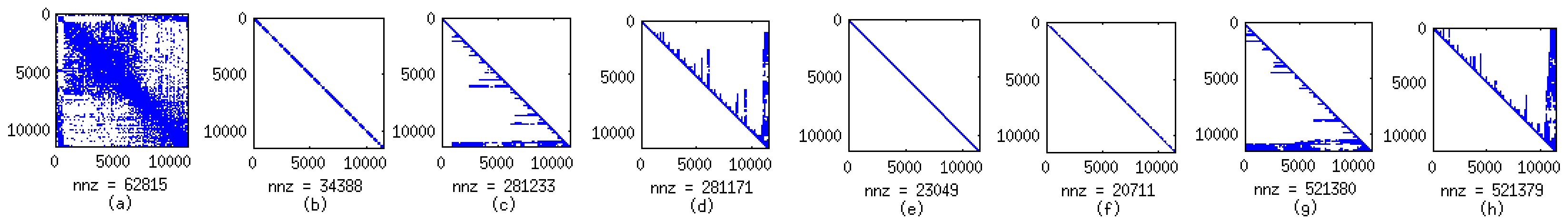


Figure 5.1: Visualization of post-extraction matrices' non-zero elements distributions from a design **FreeCPU** [2], the sizes of matrix are 11417×11417, which are obtained from SPEF extracted by industrial tool *Synopsys Star-RCXT*. *nnz* is the number of non-zeros in the matrix. (a) Extracted capacitance matrix C (non-zero entries distribute widely in the matrix). (b) Extracted conductance matrix G (there are many off-diagonal non-zeros in the matrix, but the bandwidth is much smaller than C). (c) Lower triangular matrix L_C and (d) Upper triangular matrix U_C of $LU\_decompose(C)$; (e) Lower triangular matrix L_G and (f) Upper triangular matrix U_G of $LU\_decompose(G)$; (g) Lower triangular matrix $L_{\frac{C}{h}+G}$ and (h) Upper triangular matrix $U_{\frac{C}{h}+G}$ of $LU\_decompose(\frac{C}{h}+G)$. The function of *LU_decompose* uses MATLAB2013a UMFPACK. L_G and U_G contain much smaller *nnz* than L_C, U_C, $L_{\frac{C}{h}+G}$ and $U_{\frac{C}{h}+G}$. [39]

factorization algorithms [28]. For the factorized matrices from Fig. 5.1 (c) to 5.1 (h), we can observe the effects of distribution and number of *nnz* in matrices of Fig. 5.1 (a) and (b). Factorized L_G and U_G from the matrix G contains less than 10% *nnz* of $L_{C/h+G}$, L_C and $U_{C/h+G}$, U_C. The larger number of *nnz* will increase the runtime of matrix solving via direct solver [28] within BENR, under the same given software packages and hardware resources.

5.5 Numerical Results

In this section, we present the numerical results to compare our proposed circuit simulation framework EI (Exponential Integration Kernel) and conventional SPICE integration method BENR (Backward Euler Time Integration with Newton-Raphson). The numerical algorithms are implemented in MATLAB. Device evaluation and matrix stamping are done in C/C++ with BSIM3 model for MOSFET. The interactions between C/C++ and MATLAB2014a are through MATLAB Executable (MEX) external interface with GCC 4.4.7. We perform our experiments on a Linux server with Intel(R) Xeon(R) CPU E5-2640 v3 2.60GHz and 125 GB memory (except the cases labeled with *).

The test case specification is listed in Table. 5.1, which includes industrial cases. All of algorithms and procedures are tested in the single thread and no JVM mode via launching MATLAB in command line $matlab\ -singleCompThread\ -nojvm$. We use external Python script to frequently access the memory information of the corresponding MATLAB instance and report the peak value during the time domain simulation in Table 5.2.

The results among BENR with EI should maintain the closed accuracy. In Chapter 3 we demonstrate that, in the linear system, the error is already significant by BE. In order to make the performance more comparable and eliminate the degree of freedom

Table 5.1: Test Case Specification

#N: the number of unknowns/the dimension of circuit matrices; **#Dev.**: the number of nonlinear devices. $\mathbf{nnz}_{Cl}$ and $\mathbf{nnz}_{Gl}$: the number of non-zero elements in linear capacitance/inductance matrix C and conductance/resistance matrix G.

ID	#N	#Dev.	nnz_{Cl}	nnz_{Gl}
1	52	98	50	3
2	259	260	302	264
3	2826	253	8700	6286
4	8K	19K	4K	12K
5	11K	24	63K	34K
6	63K	12K	0.32M	0.18M
7	0.12M	0.12M	0.32M	0.18M
8	0.23M	0.24M	0.64M	0.36M
9	1.17M	1.2M	3.19M	1.82M
10	11.54M	0.24M	62.83M	34.40M

from degradation of numerical accuracy, we restrict the SPICE-like option constraints for BENR, such as maximum time step in BENR for the strong dynamics region. To provide more comprehensive information as reference, we generate the simulation results from BENR with three sets of constraints, which under (1) the loose constraint set *lcs*, (2) the medium constraint set *mcs*, and (3) the strict constraint set *scs*.

The constraint sets lead to different total step numbers during the transient simulation processes, roughly $\text{Step}_{mcs}/\ \text{Step}_{lcs} \approx 10$ and $\text{Step}_{scs}/\text{Step}_{mcs} \approx 10$. We observe that when we use constraint set *scs*, the time step size is smaller (total time step number is larger), the waveforms converge to the results obtained by EI, which is with relatively larger time step.

In Table 5.2, we record statistics from the medium constraint *mcs*, i.e., #Step, average Newton-Raphson (NR). To compare the solution differences among all test cases, we use the peak error metric

$$Df^p = \frac{1}{V_{nom}} max_{t=0,\cdots,K}\ |x(t,:) - x_b(t,:)|_\infty, \tag{5.33}$$

Table 5.2: Simulation Performance Comparison Between Numerical integration with BENR and proposed Exponential Integration

	Numerical Integration via BENR						Proposed Exponential Integration (EI)					Df^p/Df^a Comparisons (%/%)			
ID	T_l(s)	Step	NR	M(B)	T_m(s)	T_s(s)	Step	m_a	m_p	Mem	T(s)	Df_l	Df_m	Df_s	SPD
1	2	10K	2.4	165M	21	187	2K	6.7	25	166M	18	8.08/1.99	1.11/0.25	0.31/0.07	10×
2	38	100K	2.6	168M	350	3452	12K	6.8	75	171M	126	2,16/0.19	0.31/0.03	0.19/0.02	27×
3	20	20K	1.1	175M	197	1,908	2K	4.2	14	173M	41	0.16/0.04	0.12/0.03	0.11/0.03	47×
4	149	8.0K	1.4	269M	923	8,754	2K	2.9	51	257M	363	16.7/2.15	4.00/0.48	1.79/0.16	24×
5	264	5.0K	2.6	314M	2.2K	21.2K	948	8.8	20	183M	131	4.91/2.38	3.14/0.66	0.21/0.09	162×
6	2.6K	5.0K	3.9	631M	17.6K	181K	978	17.3	45	493M	1.7K	4.81/1.76	3.05/0.62	0.23/0.08	106×
7	312	503	3.0	1.19G	2.5K	29K	121	9.0	24	0.80G	215	5.08/2.16	0.52/0.22	0.14/0.05	135×
8	621	507	3.4	1.86G	4.6K	60K	121	10.3	31	1.47G	653	5.05/2.15	0.57/0.24	0.08/0.03	92×
9	2.0K	253	3.9	8.29G	13.1K	191K	70	15.3	24	6.10G	1.4K	1.95/0.97	0.20/0.08	0.05/0.01	136×
10*	27K	253	4.3	52.2G	175K	–	70	13.3	29	16.9G	14.2K	1.95/0.97	0.19/0.09	–	NA

* Program is run in the Linux Server with CPU Intel Xeon E-2620 v3 @ 2.40GHz and 252GB Memory. NA is caused by the runtime of corresponding task is beyond the time budget.

Numerical Integration via BENR:

Step: the number of steps for transient simulation with the constraint *mcs*. NR: the average number of Newton-Raphson iterations for each time step. Mem: the peak memory consumption observed during the transient simulation with the constraint *mcs* in byte. T_m: the runtime of BENR with the constraint *mcs* in second. T_l: the runtime of BENR with the constraint *lcs* in second. T_s: the runtime of BENR with the constraint *scs* in second.

Proposed Exponential Integration (EI):

Step: the number of steps for transient simulation with *mcs* constraints. m_a: the average number of invert Krylov subspace for each time step, including the parts within the solution correction and the time step shrinking. m_p: the peak dimension number of invert Krylov subspace for each time step, including the parts within the solution correction and the time step shrinking. T(s): the runtime of transient simulation via EI kernel in second. Mem: the peak memory consumption during the transient simulation with EI in byte.

Df^p/Df^a Comparisons (%/%): Compare the metrics the peak solution difference Df^p from Eq. (5.33) and the average solution difference Df^a Eq. (5.34). Df_l (%/%): Df^p and Df^a under the constraint $_{lcs}$. Df_m (%/%): Df^p and Df^a under the constraint $_{mcs}$. Df_s (%/%): Df^p and Df^a under the constraint $_{scs}$. SPD: the runtime speedup of proposed method with EI over BENR. SPD = T_s(s)/T(s).

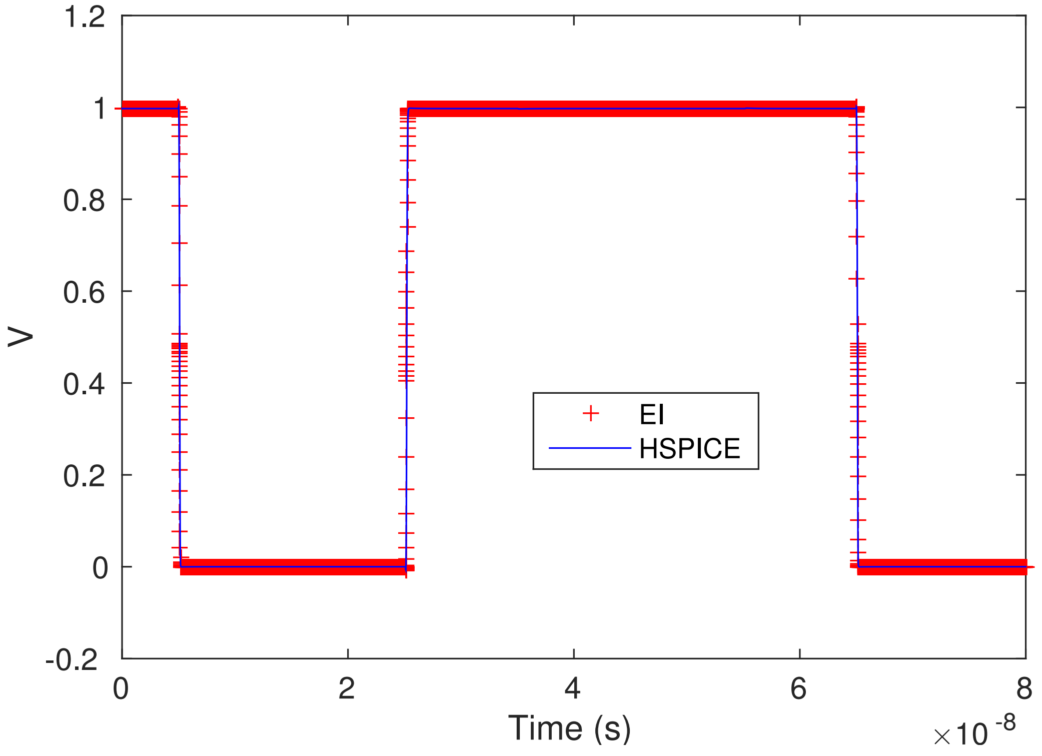


Figure 5.2: Accuracy reference between EI and HSPICE by industrial SRAM design (Case ID 4).

where x is the observed nodes computed by proposed method, x_b is the observed nodes computed by corresponding BENR method; K is number of time step; $x(t=0)$ is the DC solution, and V_{nom} is a scaling factor, which is the nominal voltage from each test case. The purpose of this normalization is for comparisons among all the test cases.

Furthermore, to get more sense of the whole transient simulation results, we define a statistical metric via root-mean-square error (RMSE),

$$Df^a = \frac{1}{V_{nom}} \sqrt{\frac{\sum_{t=0,\cdots,K} |(x(t,:) - x_b(t,:)|_\infty^2}{K+1}}. \tag{5.34}$$

For proposed EI, we also use m_a and m_p to record the average and peak dimension of Krylov subspace basis generated for one time step, which is from Line 6 to Line 29.

Regarding the runtime performance in Table 5.2, proposed EI achieves over one

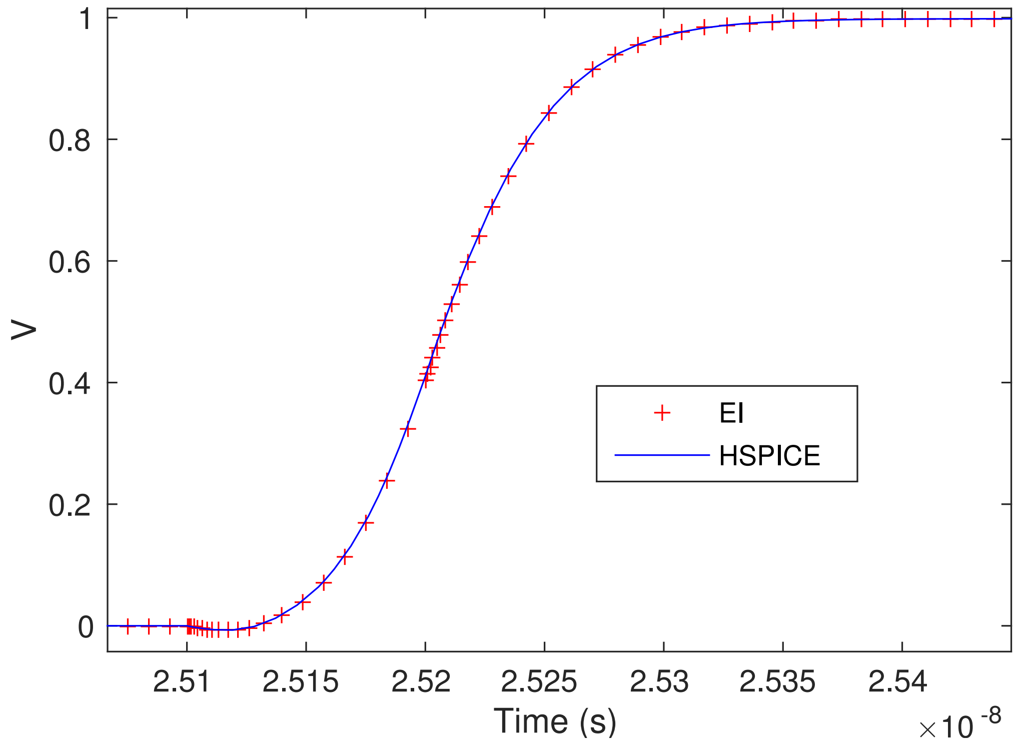


Figure 5.3: Zoom-in figure of Fig. 5.2 for the accuracy comparison between EI and HSPICE by industrial SRAM design (Case ID 4).

hundred speedup with closed accuracy compared to BENR method with constraint *scs*. If we chose *lcs* constraint, we observe smaller number of time step, smaller runtime number, but worse accuracy metrics Df^p and Df^a. The industrial SRAM design (Case ID 4) is used in our test cases, which has 19K MOSFETs. The extracted parasitics contribute to 4K *nnz* in C_l and 12K *nnz* in G_l of the matrices. Fig. 5.2 shows the accuracy comparison between EI and the industrial tool HSPICE. Fig. 5.3 is the zoom-in figure of Fig. 5.2.

In terms of memory, EI has lower consumption than BENR, especially for the cases with complicated C. For example, Case 8 has large amount *nnz* in matrix C_l, which is 62.83M and larger than 32.40M *nnz* in G_l. The dominate memory consumption is still from $LU_{Decompose}$. Therefore, we have memory performance gain for those test cases.

5.6 Limitations and Possible Solutions

Note that the test cases in this section have dominate eigenvalues of $G^{-1}C$ closed to the input transitions. EI performs well since invert Krylov subspace is able to capture the response of interest in efficient manner. However, when the range of dominate eigenvalues is far away from the response time of interest, the performance of EI degrades. It is because that the large dimension of Krylov subspace is required for capturing the eigenvalues for the region of interest. One possible solution is to replace the invert Krylov basis with rational Krylov basis and set the γ to the time step of interest.

5.7 Summary

In this chapter, we propose an efficient algorithmic framework (EI) for time domain large-scale nonlinear circuit simulation using exponential integration. The product of matrix function and vector is computed by efficient invert Krylov subspace. The numerically error from nonlinearity is controlled by measuring the system residue against KCL/KVL. We also devise a residue based compensation iteration to maintain the accuracy through the refinement.

Compared to conventional methods, our new framework has several distinguished features. By virtue of the stable explicit nature of our formulation, we remove Newton-Raphson iterations and reduce the number of LU decomposition operations. In addition, this approach can keep capacitance/inductance matrix C from matrix factorization. Moreover, within one time step integration, EI does not need to repeat LU decompositions when the length of time step is adjusted for error constraint. The proposed EI method can handle the test cases with the matrices C, which contain many parasitics. We test the proposed EI against BENR (standard Backward Euler method with Newton-Raphson iterations) and achieve runtime improvement.

Chapter 5, in part, is currently being prepared for submission for publication of the material by Hao Zhuang, Wenjian Yu, Deokseong Kim, Xinyuan Wang, and Chung-Kuan Cheng. The thesis author was the primary investigator and author of this material. This chapter also contains the content from “Dynamic Analysis of Power Delivery Network with Nonlinear Components Using Matrix Exponential Method” by Hao Zhuang, Xinan Wang, Ilgweon Kang, Jeng-Hau Lin, and Chung-Kuan Cheng in *Proceedings of IEEE International Symposium on Electromagnetic Compatibility 2015*, and “An Algorithmic Framework of Large-Scale Circuit Simulation Using Exponential Integrators” by Hao Zhuang, Wenjian Yu, Ilgweon Kang, Xinan Wang, and Chung-Kuan Cheng in *Proceedings of IEEE/ACM Design Automation Conference 2015*. The thesis author was the primary investigator and author of the papers.

Chapter 6

Conclusions

6.1 Summary of Contributions

In this thesis, we study the exponential time integration for transient analysis of large-scale circuits. The contributions of this study are listed as follows.

Chapter 3 presents the formulation of exponential integration. We also illustrate the error distribution via exponential integration based approaches with standard, invert, and rational Krylov subspace methods, and compare with traditional integration methods, such as Forward Euler, Backward Euler, and Trapezoidal methods. The different trends show the scope of application among those approaches.

Chapter 4 investigates the exponential based integration formulation for transient analysis of linear circuits. We target the challenging large-scale VLSI power network simulation problem. In numerical results, rational Krylov subspace method for the computation of matrix and vector product can achieve 14.4X speedups over conventional approach via trapezoidal integration with fixed time step on our benchmarks. Furthermore, we leverage the distributed computation framework and accelerate the simulation up to 98.0X with high level of accuracy.

Chapter 5 presents an explicit integration framework for nonlinear dynamical systems. Since the nonlinear system is treated in explicit format, we remove Newton-Raphson iteration during the whole transient analysis and greatly reduce the times of matrix factorization. In order to control the error, we use residue checking to maintain KCL/KVL laws numerically and control the time step when strong nonlinearity is encountered. In our numerical results, when invert Krylov subspace can capture the dynamics in the time step of interest, we can achieve over hundred speedup on the test cases with strong capacitive couplings compared to BENR with closed accuracy level. The efficiency of proposed EI on different types of circuits is worthwhile further investigating, especially for the test case with the response time of interest far away from dominate eigenvalues of $G^{-1}C$.

6.2 Future Work and Possible Directions

Since the application of exponential integration and matrix exponentials in circuit simulation society is still quite new, there are many aspects to be explored in the future.

- The nature of explicit formulation exposes more parallel processing opportunity than traditional implicit based circuit simulation algorithms. Sparse matrix and vector multiplication (SpMV) plays an important role in this framework, so that parallel processing SpMV using advanced multi-core and many-core architectures could be beneficial to enhance the runtime performance.
- To extend Chapter 4, we can use more advanced schedule technique and parallel computational power to trade the runtime performance.
- For the model of power network analysis in Chapter 4, we can add the current models with voltage-dependent sources in order to get solutions more accurate to

actual results than the modeling part of current academic benchmarks. Based on the new formulation, it is very interesting to see the performance gain by exponential time integration based approaches.

- For simulation of nonlinear systems, smarter error and step size controlling schemes are also important to further accelerate the whole simulation process.
- The high order ϕ_i computation is favored to be further investigated in order to extend the approximation theory in Eq. (5.13). The research on high order effects and the nonlinearity is also interesting, which might provide deeper insight for the error and step size control in our explicit formulation.
- In Arnoldi algorithm that generates Krylov subspace, the iterative linear matrix solver [40, 67] could be one possible solution for extremely larger dynamical systems.
- Gather more test cases and test the application scope of the framework EI, such as analog circuits and RF circuits. Besides, we can replace the Krylov subspace basis of EI with rational, or standard version for more thorough comparisons.

Bibliography

[1] C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later," *SIAM review*, vol. 45, no. 1, pp. 3–49, 2003.

[2] C. Zhang and W. Yu, "Efficient space management techniques for large-scale interconnect capacitance extraction with floating random walks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 10, pp. 1633–1637, 2013.

[3] F. N. Najm, *Circuit simulation*. Wiley, 2010.

[4] Y. Cheng, M. Chan, K. Hui, M.-C. Jeng, Z. Liu, J. Huang, K. Chen, J. Chen, R. Tu, P. K. Ko, and C. Hu, "BSIM3v3 manual," *University of California, Berkeley*, 1996.

[5] L. W. Nagel and D. O. Pederson, *SPICE: Simulation program with integrated circuit emphasis*. Electronics Research Laboratory, College of Engineering, University of California, 1973.

[6] L. Nagel and R. Rohrer, "Computer analysis of nonlinear circuits, excluding radiation (CANCER)," *IEEE Journal of Solid-State Circuits*, vol. 6, no. 4, pp. 166–182, 1971.

[7] L. O. Chua and P.-M. Lin, *Computer Aided Analysis of Electric Circuits: Algorithms and Computational Techniques*. Prentice-Hall, 1975.

[8] L. T. Pillage, R. A. Rohrer, and C. Visweswariah, *Electronic circuit and system simulation methods*. McGraw-Hill New York, 1995.

[9] Z. Zhu, H. Peng, C. K. Cheng, K. Rouz, M. Borah, and E. S. Kuh, "Two-stage Newton-Raphson method for transistor-level simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 5, pp. 881–895, 2007.

[10] P. Li, "Parallel circuit simulation: A historical perspective and recent developments," *Foundations and Trends in Electronic Design Automation*, vol. 5, no. 4, pp. 211–318, 2012.

[11] X. Ye, W. Dong, P. Li, and S. Nassif, “Maps: Multi-algorithm parallel circuit simulation,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 73–78, 2008.

[12] W. Dong and P. Li, “Parallelizable stable explicit numerical integration for efficient circuit simulation,” in *Proceedings of IEEE/ACM Design Automation Conference*, 2009.

[13] W. Dong, P. Li, and X. Ye, “Wavepipe: Parallel transient simulation of analog and digital circuits on multi-core shared-memory machines,” in *Proceedings of IEEE/ACM Design Automation Conference*, pp. 238–243, 2008.

[14] S.-H. Weng, Q. Chen, and C. K. Cheng, “Time-domain analysis of large-scale circuits by matrix exponential method with adaptive control,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 8, pp. 1180–1193, 2012.

[15] C. Moler and C. Van Loan, “Nineteen dubious ways to compute the exponential of a matrix,” *SIAM review*, vol. 20, no. 4, pp. 801–836, 1978.

[16] Y. Saad, “Analysis of some krylov subspace approximations to the matrix exponential operator,” *SIAM Journal on Numerical Analysis*, vol. 29, no. 1, pp. 209–228, 1992.

[17] M. Hochbruck and A. Ostermann, “Exponential integrators,” *Acta Numerica*, vol. 19, pp. 209–286, 2010.

[18] J. Loffeld and M. Tokman, “Comparative performance of exponential, implicit, and explicit integrators for stiff systems of ODEs,” *Journal of Computational and Applied Mathematics*, vol. 241, pp. 45–67, 2013.

[19] M. Caliari and A. Ostermann, “Implementation of exponential rosenbrock-type integrators,” *Applied Numerical Mathematics*, vol. 59, no. 3, pp. 568–581, 2009.

[20] C. Ho, A. Ruehli, and P. Brennan, “The modified nodal approach to network analysis,” *IEEE Transactions on Circuits and Systems*, vol. 22, no. 6, pp. 504–509, 1975.

[21] K. Nichols, T. Kazmierski, M. Zwolinski, and A. Brown, “Overview of spice-like circuit simulation algorithms,” *IEE Proceedings-Circuits, Devices and Systems*, vol. 141, no. 4, pp. 242–250, 1994.

[22] G. Wanner, “Dahlquist’s classical papers on stability theory,” *BIT Numerical Mathematics*, vol. 46, no. 3, pp. 671–683, 2006.

[23] O. Nastov, R. Telichevesky, K. Kundert, and J. White, “Fundamentals of fast simulation algorithms for RF circuits,” *Proceedings of the IEEE*, vol. 95, no. 3, pp. 600–621, 2007.

[24] H. Zhuang, W. Yu, G. Hu, Z. Liu, and Z. Ye, “Fast floating random walk algorithm for multi-dielectric capacitance extraction with numerical characterization of Green’s functions,” in *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference*, pp. 377–382, 2012.

[25] W. Yu, H. Zhuang, C. Zhang, G. Hu, and Z. Liu, “RWCap: A floating random walk solver for 3-D capacitance extraction of very-large-scale integration interconnects,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 3, pp. 353–366, 2013.

[26] W. Yu and X. Wang, *Advanced Field-Solver Techniques for RC Extraction of Integrated Circuits*. Springer, 2014.

[27] R. Ionutiu, J. Rommes, and W. H. Schilders, “SparseRC: Sparsity preserving model reduction for RC circuits with many terminals,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 12, pp. 1828–1841, 2011.

[28] T. A. Davis, *Direct Method for Sparse Linear Systems*. SIAM, 2006.

[29] G. Karypis and V. Kumar, “A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices,” *University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN*, 1998.

[30] X. Chen, Y. Wang, and H. Yang, “NICSLU: An adaptive sparse matrix solver for parallel circuit simulation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 2, pp. 261–274, 2013.

[31] X. Chen, L. Xia, Y. Wang, and H. Yang, “Sparsity-oriented sparse solver design for circuit simulation,” in *Proceedings of IEEE Design, Automation, and Test in Europe Conference & Exhibition*, pp. 1580–1585, 2016.

[32] T. A. Davis and E. Palamadai Natarajan, “Algorithm 907: Klu, a direct sparse solver for circuit simulation problems,” *ACM Transactions on Mathematical Software*, vol. 37, no. 3, p. 36, 2010.

[33] K. He, S. X.-D. Tan, H. Wang, and G. Shi, “Gpu-accelerated parallel sparse lu factorization method for fast circuit analysis,” *IEEE Transactions on Very Large Scale Integration Systems*, vol. 24, no. 3, pp. 1140–1150, 2016.

[34] L. Nagel, *SPICE2: A computer program to simulate semiconductor circuits*. Ph.D. dissertation, 1975.

[35] M. Hochbruck, A. Ostermann, and J. Schweitzer, “Exponential Rosenbrock-type methods,” *SIAM Journal of Numerical Analysis*, vol. 47, no. 1, pp. 786–803, 2009.

[36] H. Zhuang, S.-H. Weng, and C. K. Cheng, "Power grid simulation using matrix exponential method with rational krylov subspaces," in *Proceedings of IEEE International Conference on ASIC*, 2013.

[37] H. Zhuang, S.-H. Weng, J.-H. Lin, and C. K. Cheng, "MATEX: A distributed framework of transient simulation of power distribution networks," in *Proceedings of IEEE/ACM Design Automation Conference*, 2014.

[38] H. Zhuang, W. Yu, S.-H. Weng, I. Kang, J.-H. Lin, X. Zhang, R. Coutts, J. Lu, and C. K. Cheng, "Simulation algorithms with exponential integration for time-domain analysis of large-scale power delivery networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016.

[39] H. Zhuang, W. Yu, I. Kang, X. Wang, and C. K. Cheng, "An algorithmic framework for efficient large-scale circuit simulation using exponential integrators," in *Proceedings of IEEE/ACM Design Automation Conference*, 2015.

[40] L. Orecchia, S. Sachdeva, and N. K. Vishnoi, "Approximating the exponential, the lanczos method and an o (m)-time spectral algorithm for balanced separator," in *ACM Symposium on Theory of Computing*, pp. 1141–1160, 2012.

[41] M. A. Botchev, "A short guide to exponential krylov subspace time integration for maxwell's equations," Department of Applied Mathematics, University of Twente, 2012.

[42] J. van den Eshof and M. Hochbruck, "Preconditioning Lanczos approximations to the matrix exponential," *SIAM Journal on Scientific Computing*, vol. 27, no. 4, pp. 1438–1457, 2006.

[43] Q. Chen, S.-H. Weng, and C. K. Cheng, "A practical regularization technique for modified nodal analysis in large-scale time-domain circuit simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 1031–1040, 2012.

[44] J. Wilkinson, "Kronecker's canonical form and the QZ algorithm," *Linear Algebra and its Applications*, vol. 28, pp. 285–303, 1979.

[45] D. Kouroussis and F. N. Najm, "A static pattern-independent technique for power grid voltage integrity verification," in *Proceedings of IEEE/ACM Design Automation Conference*, pp. 99–104, 2003.

[46] S. R. Nassif and J. N. Kozhaya, "Fast power grid simulation," in *Proceedings of IEEE/ACM Design Automation Conference*, pp. 156–161, 2000.

[47] M. S. Gupta, J. L. Oatley, R. Joseph, G.-Y. Wei, and D. M. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Proceedings of IEEE Design, Automation, and Test in Europe Conference & Exhibition*, pp. 1–6, 2007.

[48] S. Lin, M. Nagata, K. Shimazake, K. Satoh, M. Sumita, H. Tsujikawa, and A. T. Yang, "Full-chip vectorless dynamic power integrity analysis and verification against 100uv/100ps-resolution measurement," in *Proc. IEEE CICC*, pp. 509–512, 2004.

[49] S. Lin and N. Chang, "Challenges in power-ground integrity," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 651–654, 2001.

[50] R. Zhang, B. H. Meyer, W. Huang, K. Skadron, and M. R. Stan, "Some limits of power delivery in the multicore era," *Proceedings of WEED*, 2012.

[51] R. Zhang, K. Wang, B. H. Meyer, M. R. Stan, and K. Skadron, "Architecture implications of pads as a scarce resource," in *Proceedings of International Symposium on Computer Architecture*, pp. 373–384, 2014.

[52] K. Wang, B. H. Meyer, R. Zhang, K. Skadron, and M. R. Stan, "Walking pads: Fast power-supply pad-placement optimization.," in *Proceedings of IEEE/ACM Asia and South Pacific Design Automation Conference*, vol. 20, p. 4, 2014.

[53] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. Huang, C.-C. Teng, and C.-K. Cheng, "ePlace: Electrostatics based placement using Nesterov's method," in *Proceedings of IEEE/ACM Design Automation Conference*, pp. 1–6, 2014.

[54] M. Pan, N. Viswanathan, and C. Chu, "An efficient and effective detailed placement algorithm," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 48–55, 2005.

[55] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng, and C. K. Cheng, "ePlace-MS: Electrostatics based placement for mixed-size circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 5, pp. 685–698, 2015.

[56] L. Xiao, Z. Xiao, Z. Qian, Y. Jiang, T. Huang, H. Tian, and E. F. Y. Young, "Local clock skew minimization using blockage-aware mixed tree-mesh clock network," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 458–462, 2010.

[57] Y. Zhang and C. Chu, "GDRouter: Interleaved global routing and detailed routing for ultimate routability," in *Proceedings of IEEE/ACM Design Automation Conference*, pp. 597–602, 2012.

[58] A. B. Kahng, S. Kang, H. Lee, I. L. Markov, and P. Thapar, "High-performance gate sizing with a signoff timer," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 450–457, 2013.

[59] C. Zhuo, G. Wilke, R. Chakraborty, A. Aydiner, S. Chakravarty, and W.-K. Shih, "A silicon-validated methodology for power delivery modeling and simulation," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 255–262, 2012.

[60] Z. Zeng, X. Ye, Z. Feng, and P. Li, "Tradeoff analysis and optimization of power delivery networks with on-chip voltage regulation," in *Proceedings of IEEE/ACM Design Automation Conference*, pp. 831–836, 2010.

[61] H. Zhuang, J. Lu, K. Samadi, Y. Du, and C. K. Cheng, "Performance-driven placement for design of rotation and right arithmetic shifters in monolithic 3D ICs," in *Proceedings of IEEE International Conference on Communications, Circuits and Systems*, vol. 2, pp. 509–513, 2013.

[62] S. K. Samal, K. Samadi, P. Kamal, Y. Du, and S. K. Lim, "Full chip impact study of power delivery network designs in monolithic 3D ICs," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 565–572, 2014.

[63] H. Zhuang, X. Wang, I. Kang, J.-H. Lin, and C. K. Cheng, "Dynamic analysis of power delivery network with nonlinear components using matrix exponential method," in *Proceedings of IEEE Symposium on Electromagnetic Compatibility and Signal Integrity*, 2015.

[64] S. R. Nassif, "Power grid analysis benchmarks," in *Proceedings of Asia and South Pacific Design Automation Conference*, pp. 376–381, 2008.

[65] Z. Li, R. Balasubramanian, F. Liu, and S. Nassif, "2012 tau power grid simulation contest: benchmark suite and results," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 643–646, 2012.

[66] C. Zhuo, H. Gan, and W.-K. Shih, "Early-stage power grid design: Extraction, modeling and optimization," in *Proceedings of IEEE/ACM Design Automation Conference*, pp. 1–6, 2014.

[67] Y. Saad, *Iteravite Methods for Sparse Linear Systems*. SIAM, 2003.

[68] T. Yu and M. D.-F. Wong, "PGT_SOLVER: An efficient solver for power grid transient analysis," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 647–652, 2012.

[69] J. Yang, Z. Li, Y. Cai, and Q. Zhou, "Powerrush: Efficient transient simulation for power grid analysis," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 653–659, 2012.

[70] X. Xiong and J. Wang, "Parallel forward and back substitution for efficient power grid simulation," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 660–663, 2012.

[71] M. Zhao, R. V. Panda, S. S. Sapatnekar, and D. Blaauw, "Hierarchical analysis of power distribution networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 2, pp. 159–168, 2002.

[72] X. Ye, M. Zhao, R. Panda, P. Li, and J. Hu, "Accelerating clock mesh simulation using matrix-level macromodels and dynamic time step rounding," in *Proceedings of IEEE International Symposium on Quality Electronic Design*, pp. 627–632, 2008.

[73] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center.," in *USENIX Networked Systems Design and Implementation*, vol. 11, pp. 22–22, 2011.

[74] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[75] Z. Li and C.-J. Shi, "SILCA: SPICE-accurate iterative linear-centric analysis for efficient time-domain simulation of vlsi circuits with strong parasitic couplings," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 1087–1103, 2006.

[76] J. R. Phillips and L. M. Silveira, "Simulation approaches for strongly coupled interconnect systems," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 430–437, 2001.

[77] S. Lin, E. S. Kuh, and M. Marek-Sadowska, "Stepwise equivalent conductance circuit simulation technique," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 5, pp. 672–683, 1993.

[78] A. Devgan and R. A. Rohrer, "Adaptively controlled explicit simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 6, pp. 746–762, 1994.

[79] Q. He, H. Gan, and D. Jiao, "Explicit time-domain finite-element method stabilized for an arbitrarily large time step," *IEEE Transactions on Antennas and Propagation*, vol. 60, no. 11, pp. 5240–5250, 2012.

[80] V. T. Luan and A. Ostermann, "Parallel exponential rosenbrock methods," *Computers & Mathematics with Applications*, 2016.

[81] J. Niesen and W. M. Wright, "Algorithm 919: A krylov subspace algorithm for evaluating the ϕ-functions appearing in exponential integrators," *ACM Transactions on Mathematical Software*, vol. 38, no. 3, p. 22, 2012.

[82] N. J. Higham, *Functions of matrices: theory and computation*. SIAM, 2008.

[83] A. H. Al-Mohy and N. J. Higham, "Computing the action of the matrix exponential, with an application to exponential integrators," *SIAM Journal on Scientific Computing*, vol. 33, no. 2, pp. 488–511, 2011.