MULTIPHYSICS AND LARGE-SCALE MODELING AND SIMULATION METHODS FOR ADVANCED INTEGRATED CIRCUIT DESIGN

by

Shuzhan Sun

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Electrical and Computer Engineering

West Lafayette, Indiana

December 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL STATEMENT OF COMMITTEE APPROVAL

Dr. Dan Jiao, Chair

School of Electrical and Computer Engineering

Dr. Alexander V. Kildishev

School of Electrical and Computer Engineering

Dr. Peter Bermel

School of Electrical and Computer Engineering

Dr. Zhihong Chen

School of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

To my family and loved ones.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my advisor Professor Dan Jiao to have a chance to do this work with her in Purdue University. She introduced me into the area of computational electromagnetics and imparts her knowledge to me step by step. Doing research sometimes feels like searching in the ocean and it is easy to get lost. Professor Jiao is like a lighthouse and a dose of encouragement, giving me the clear direction of my research and helping me get out of the mist. She is always there whenever I meet any difficulty or get stuck. Without her help, this work can never be complete. I feel lucky that Professor Jiao is my PhD advisor.

Also, I would like to thank to my other committee members: Professor Alexander V. Kildishev, Professor Peter Bermel, and Professor Zhihong Chen for their precious time and insightful help to my research work.

Thanks to my labmates who have worked with me in On-Chip Electromagnetic group: Dr. Jin Yan, Dr. Kaiyuan Zeng, Dr. Miaomiao Ma, Dr. Li Xue, Michael R. Hayashi, Yifan Wang, Chang Yang, Daniel Wei, and Vinicius Cabral Do Nascimento for their professional discussion and friendly support.

Last but not least, I would like to thank my parents, my family, and all my friends for their love and support during these years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The design of advanced integrated circuits (ICs) and systems calls for multiphysics and large-scale modeling and simulation methods. On the one hand, novel devices and materials are emerging in next-generation IC technology, which requires multiphysics modeling and simulation. On the other hand, the ever-increasing complexity of ICs requires more efficient numerical solvers.

In this work, we propose a multiphysics modeling and simulation algorithm to co-simulate Maxwell's equations, dispersion relation of materials, and Boltzmann equation to characterize emerging new devices in IC technology such as Cu-Graphene (Cu-G) hybrid nano-interconnects. We also develop an unconditionally stable time marching scheme to remove the dependence of time step on space step for an efficient simulation of the multiscaled and multiphysics system. Extensive numerical experiments and comparisons with measurements have validated the accuracy and efficiency of the proposed algorithm. Compared to simplified steady-state-models based analysis, a significant difference is observed when the frequency is high or/and the dimension of the Cu-G structure is small, which necessitates our proposed multiphysics modeling and simulation for the design of advanced Cu-G interconnects.

To address the large-scale simulation challenge, we develop a new split-field domain-decomposition algorithm amenable for parallelization for solving Maxwell's equations, which minimizes the communication between subdomains, while having a fast convergence of the global solution. Meanwhile, the algorithm is unconditionally stable in time domain. In this algorithm, unlike prevailing domain decomposition methods that treat the interface unknown as a whole and let it be shared across subdomains, we partition the interface unknown into multiple components, and solve each of them from one subdomain. In this way, we transform the original coupled system to fully decoupled subsystems to solve. Only one addition (communication) of the interface unknown needs to be performed after the computation in each subdomain is finished at each time step. More importantly, the algorithm has a fast convergence and permits the use of a large time step irrespective of space step. Numerical experiments on large-scale on-chip and package layout analysis have demonstrated the capability of the new domain decomposition algorithm.

To tackle the challenge of efficient simulation of irregular structures, in the last part of the thesis, we develop a method for the stability analysis of unsymmetrical numerical systems in time domain. An unsymmetrical system is traditionally avoided in numerical formulation since a traditional explicit simulation is absolutely unstable, and how to control the stability is unknown. However, an unsymmetrical system is frequently encountered in modeling and simulating of unstructured meshes and nonreciprocal electromagnetic and circuit devices. In our method, we reduce stability analysis of a large system into the analysis of dissembled single element, therefore provides a feasible way to control the stability of large-scale systems regardless of whether the system is symmetrical or unsymmetrical. We then apply the proposed method to prove and control the stability of an unsymmetrical matrix-free method that solves Maxwell's equations in general unstructured meshes while not requiring a matrix solution.

1. INTRODUCTION

1.1 Background and Motivation

The design of advanced integrated circuits (ICs) and systems calls for multiphysics and large-scale modeling and simulation methods. On the one hand, novel devices and materials are emerging in next-generation IC technology, which requires multiphysics modeling and simulation. On the other hand, the ever-increasing complexity of ICs requires more efficient numerical solvers.

As ICs have progressed to nanometer technology nodes and higher levels of integration, existing Cu-based interconnect solution has become increasingly difficult in sustaining the continued evolution of IC technology. Due to side wall and grain boundary scatterings, the resistivity of Cu at small dimensions increases rapidly [1], which leads to, for aggressively scaled Cu interconnects, an increased resistor–capacitor (RC) delay, a lower current-driving capacity, more heat generations, a reduced interconnect bandwidth, a larger crosstalk noise, and other negative effects [2]. As a result, the overall performance and reliability of an IC can degrade significantly.

Among emerging alternatives to replace existing Cu-based interconnect technology, graphene has attracted a lot of attentions in the design of advanced ICs. Due to graphene's 2-D nature, μm-long mean free path, high physical strength, and large electrical and thermal conductivity, graphene has a potential to push the next-generation ICs to an era of ultra-scaled dimension, lower signal delay, faster data transferring speed, reduced energy consumption and heat generation, and better reliability [3]. Over the last decade, utilizing these advantages of graphene, researchers have proposed many graphene-based on-chip designs, including pure-graphene based interconnects and nanoscale functional devices, and Cu-graphene hybrid structures like graphene encapsulated Cu interconnects. However, the electrical performance of general on-chip Cu-graphene hybrid interconnects remains unclear at high operating frequencies ($\sim$ 100 GHz or higher) and in the sub-10 nm regime. On the one hand, the fabrication and measurement capability is limited. In most laboratories, only single Cu-graphene hybrid structure can be measured at either DC or an optical frequency. Thus, existing experimental capabilities are not sufficient to measure the performance of complicated on-chip Cu-graphene hybrid systems at desired frequencies. On the other hand, most of existing simulation methods, which separately model the graphene layers [4]–[6] and the hybrid Cu-graphene structure [7]–[10], may have accuracy problems in high frequency and sub-nm simulations.

The limited accuracy of existing simplified-model based simulation methods is mainly due to three factors. First, most of simplified graphene models are inaccurate at high frequencies. For example, the model in [4] gives the conductivity of graphene layer by counting the number of conduction channels. However, it only considers a graphene length shorter than the mean free path (MFP) and it fails to model the frequency dependence. Another widely used model of graphene is Kubo formula [5], which has considered both the intraband and interband transition. The intraband transition of Kubo formula corresponds to the Drude model [11]–[14] in this work and will be discussed in detail in Chapter 2. Second, the Ohm's law itself can be inaccurate because the skin depth becomes comparable to the MFP of graphene at high frequencies [11]. When the skin depth becomes comparable to the MFP of graphene, the carriers can no longer be considered to move under the influence of a constant field between collisions, and the current at any point is also influenced by the electric fields at other points. This renders the assumption of Ohm's law invalid and necessitates a more generic approach using the Boltzmann equation. Thus, any conductivity model extracted with Ohm's law is not accurate. Third, existing decoupled electrical conductivity models of graphene assume graphene's steady state responses to an external stimulus. This assumption can be valid for many low frequency applications, but it is unlikely to hold true in high-frequency settings. The main reason to the failure at high frequencies is the low back scattering frequency (BSF) of graphene ($\sim$ 100 GHz) [15], [16]. When the signal frequency in Cu-G interconnects becomes high enough to reach the relatively low BSF of graphene, graphene layers may not have enough scatterings to re-equilibrate themselves. As a result, it may not give the physical steady state response predicted by the steady-state conductivity models. Since the decoupled steady-state models can miss graphene's dynamic responses at high frequencies, a first-principles based dynamic modeling and simulation in time domain is needed.

Apart from the modeling challenges arising from new materials and new structures, another bottleneck for the design of large-scale ICs is the limited simulation capacity. To date, the fastest partial-differential-equation (PDE) based solvers scale as $\mathcal{O}(N)$ in both memory complexity and computational complexity. This performance is generally regarded as the limit that one can achieve in computational electromagnetics (CEM). However, since the number of unknowns N is huge in IC analysis even for a circuitry of a modest size, the current performance of CEM techniques is still insufficient when tackling a realistic IC design problem that can involve billions of unknowns. To further increase the simulation capacity, a parallel domain decomposition (DD) method can be deployed utilizing distributed computational resources.

A parallel domain decomposition method (DDM) [17] is popular in solving large-scale problems. In open literature, a DDM-based parallel algorithm has been developed for not only conventional Yee's finite-difference time-domain (FDTD) method, but also many unconditionally stable FDTD methods such as alternating direction implicit FDTD (ADI-FDTD) method, locally 1-D FDTD method [18], Laguerre-FDTD method [19], and Crank–Nicolson FDTD method. The same is observed in the finite element method (FEM) in both frequency and time domain. Among existing DDM-based methods, the parallelization is achieved mainly in two approaches. The first approach is a direct solver using Schur complement. The major computational cost of this approach is the dense system of interface unknowns. The second approach is an iterative solver that applies transmission conditions on the interfaces to exchange solutions between domains. However, the convergence of this approach is problem dependent.

One common feature in existing DD methods is that the interface field is treated as a whole, and shared in common by adjacent domains. The transmission condition can be formulated using tangential **E**, tangential **H**, or their weighted sum on the interface. The interface fields can be viewed as the sources of each domain. Given an interface field, the unknowns inside the domains are solved, which is essentially how the overall work is partitioned into each domain to solve.

Another direction to increase the simulation capacity is to avoid the solution of large matrices in simulation algorithms. For example, the matrix-free method developed in [20] has a naturally diagonal mass matrix, hence the need for numerically finding the matrix solution is completely eliminated. Then, much larger problems can be solved on the same computer platform. However, because of the unsymmetrical system matrix, new theory is needed to guide and control the stability of the matrix-free method. Prevailing numerical methods formulate a symmetrical system to solve, due to the nice mathematical property of a symmetrical matrix and the theory on symmetrical systems is rich and abundant. An unsymmetrical numerical method has a potential to significantly enlarge our solution space for solving numerical problems. However, it is rarely studied. The theoretical understanding of the unsymmetrical systems and the new analysis method being pursued in this work are important to the development of generic unsymmetrical numerical methods.

1.2 Contributions of This Work

In this work, we propose a multiphysics modeling and simulation algorithm to co-simulate Maxwell's equations, dispersion relation of materials, and Boltzmann equation to characterize emerging new devices in IC technology such as Cu-Graphene (Cu-G) hybrid nano-interconnects. We also develop an unconditionally stable time marching scheme to remove the dependence of time step on space step for an efficient simulation of the multiscaled and multiphysics system. Extensive numerical experiments and comparisons with measurements have validated the accuracy and efficiency of the proposed algorithm. Compared to simplified steady-state-models based analysis, a significant difference is observed when the frequency is high or/and the dimension of the Cu-G structure is small, which necessitates our proposed multiphysics modeling and simulation for the design of advanced Cu-G interconnects.

To address the large-scale simulation challenge, we develop a new split-field domain-decomposition algorithm amenable for parallelization for solving Maxwell's equations, which minimizes the communication between subdomains, while having a fast convergence of the global solution. Meanwhile, the algorithm is unconditionally stable in time domain. In this algorithm, unlike prevailing domain decomposition methods that treat the interface field as a whole and let it be shared across subdomains as a transmission condition, we split the interface field into multiple components, and solve each of them from one subdomain. In this way, we transform the original coupled system to fully decoupled subsystems to solve, and with the coupling between subdomains captured via an iteration process that is guaranteed to converge. Only one addition (communication) of the interface unknown needs to be performed after the computation in each subdomain is finished at each time step. Numerical experiments on large-scale on-chip and package layout analysis have demonstrated the capability of the new domain decomposition algorithm.

To tackle the challenge of efficient simulation of irregular structures, in the last part of the thesis, we develop a method for the stability analysis of unsymmetrical numerical systems in time domain. An unsymmetrical system is traditionally avoided in numerical formulation since a traditional explicit simulation is absolutely unstable, and how to control the stability is unknown. However, an unsymmetrical system becomes more and more important in modeling and simulating of unstructured meshes and nonreciprocal electromagnetic and circuit devices. In our method, we reduce the stability analysis of a large system into the analysis of dissembled elements, therefore providing a feasible way to control the stability of large-scale systems regardless of whether the system is symmetrical or unsymmetrical. We then apply the proposed method to prove and control the stability of an unsymmetrical matrix-free method that solves Maxwell's equations in general unstructured meshes while not requiring a matrix solution.

1.3 Dissertation Outline

The remainder of this dissertation is organized as follows.

In Chap. 2, we develop a multiphysics-based model and an efficient simulation algorithm to co-simulate directly in time domain Maxwell's equations, equations characterizing graphene materials, and Boltzmann equation from direct current (DC) to high frequencies. To enable the simulation of nano-interconnects within a feasible run time, the entire numerical system is further made unconditionally stable in time marching. We show the multiphysics modeling and simulation algorithm for analyzing Cu-G interconnects, prove the time-domain stability of the coupled simulation, validate the proposed work against measured data, and also apply it to predict the crosstalk and propagation delay of Cu-G interconnects.

We also study the difference between the first-principles based method of and commonly used simplified models such as the Drude-model based approach for analyzing general on-chip Cu-graphene hybrid systems, from both theoretical and numerical perspectives. To do so, we first develop a Drude model based simulation algorithm, including the derivation of the Drude model from the Boltzmann transport equation, the numerical representation of the Drude model, and an efficient algorithm for simulating the Drude model in conjunction with the FDTD to analyze a Cu-graphene interconnect. We then compare it with the first-principles based multiphysics model and the resulting simulation algorithm both theoretically and numerically through extensive numerical experiments performed on the Cu-G nano-interconnects. We find that the first-principles based analysis is necessary to capture the physical process happening in a Cu-G interconnect at microwave frequencies and in the sub-nm regime.

In Chap. 3, we propose a new non-overlapping parallel DDM for finite difference method (FDM), which partitions both the system matrix and the interface field according to contributions from each subdomain. The proposed DDM is theoretically proved to preserve the coupling among subdomains with fast convergence. Parallel iteration scheme based on the partitioning is developed for nun-uniform FDM in both frequency and time domain. In time domain, the parallel iteration scheme is unconditionally stable, allowing for a large time step irrespective of space step. Extensive numerical examples are tested to demonstrate the capability of the proposed parallel DD solver.

In Chap. 4, we further develop the split-field DD parallel solver in time domain. We also develop a new algorithm to solve the relative residual in parallel, which makes it easier to control the overall accuracy of the solution during the iteration. We analyze the convergence of the parallel solver by deriving its overall amplification matrix in the iteration, then theoretically and numerically check its spectral radius and eigenvalue spectrum. Both on-chip and package examples are simulated, which are bench-marked with not only a direct solver but also a state-of-the-art iterative solver. We also numerically show the run time complexity, memory complexity, and parallel efficiency of the new DD solver.

In Chap. 5, we develop a theory on the stability analysis of unsymmetrical numerical systems in time domain. We show its application to prove and control the stability of a matrix-free method that solves Maxwell's equations in general unstructured meshes while not requiring a matrix solution.

In Chap. 6, we summarize the work that has been done and present the future work.

2. MULTIPHYSICS MODELING AND SIMULATION OF 3-D CU-GRAPHENE HYBRID NANO-INTERCONNECTS

2.1 Introduction

As integrated circuits (ICs) have progressed to nm technology nodes and higher levels of integration, existing Cu-based interconnect solution has become increasingly difficult in sustaining the continued evolution of IC technology. Due to side wall and grain boundary scatterings, the resistivity of Cu at small dimensions increases rapidly [1], which leads to, for aggressively scaled Cu interconnects, an increased resistor–capacitor (RC) delay, a lower current-driving capacity, more heat generations, a reduced interconnect bandwidth, a larger crosstalk noise, and other negative effects [2]. As a result, the overall performance and reliability of an IC can degrade significantly.

Cu-Graphene (Cu-G) hybrid nano-interconnect solutions, such as graphene encapsulated Cu interconnects, are promising alternatives to Cu-based interconnects. The hybrid can benefit from the combined properties of both materials, and hence can be superior to either of them in terms of electrical and thermal performance. Compared with Cu-based interconnects, Cu-G interconnects exhibit an enhanced electrical conductivity and current driving capacity [3], [21], a faster data transferring speed [3], a larger thermal conductivity [22], and the resistance to electromigration therefore a better long-term reliability [23]. However, as far as the modeling of Cu-G interconnect is concerned, most of existing methods separately model the graphene layers [4], [6] and the hybrid interconnect structure [7]–[10]. Such a decoupled approach may cause accuracy problems in high frequency simulations for two main reasons. First, as shown in [11], most of these graphene models are no longer sufficient at high frequencies since skin depth becomes comparable to the mean free path. Second, these decoupled electrical conductivity models of graphene [4], [6] assume graphene's steady state responses to external stimulus. This assumption can be valid for many low frequency measurements and single-frequency stimuli, but is unlikely to hold in emerging high-frequency IC scenarios. The main reason to the failure at high frequencies is the low back scattering frequency (BSF) of graphene ($\sim$ 100 GHz) [15], [16]. When the signal frequency in Cu-G interconnects becomes high enough to reach the relatively low BSF of graphene, graphene layers may not have enough scatterings to re-equilibrate themselves, thus may not give the physical steady state response as predicted by steady state conductivity models. Since the decoupled steady state models can miss graphene's dynamic electronic responses in high frequency simulations, a full-wave dynamic modeling and simulation in time domain is needed.

To successfully develop Cu-G new interconnect solutions for high frequency IC technology, it is necessary to understand the entire physical process that takes place in a Cu-G interconnect. Under a voltage or current source excitation, the electric and magnetic fields are generated in the physical layout of a graphene interconnect. These fields drive the movement of charge carriers in the graphene material. The resultant change in conduction current, in turn, modifies the electric and magnetic field distributions. At high frequencies (e.g. 50 GHz), the graphene layer may never reach a steady state, resulting in a nonlinear conduction current response. To the best of our knowledge, none of existing models have sufficiently captured the dynamic physics present in the Cu-G interconnects. Hence, they may lose their predictive power when applied to the design of new Cu-G interconnects.

In this work, we develop a multiphysics-based model and an efficient simulation algorithm to co-simulate directly in time domain Maxwell's equations, equations characterizing graphene materials, and Boltzmann equation from direct current (DC) to high frequencies. To enable the simulation of nano-interconnects within a feasible run time, the entire numerical system is further made unconditionally stable in time marching. We show the multiphysics modeling and simulation algorithm for analyzing Cu-G interconnects, prove the time-domain stability of the coupled simulation, validate the proposed work against measured data, and also apply it to predict the crosstalk and propagation delay of Cu-G interconnects which has not been reported in open literature. A version of this chapter was previously published in [24], [25].

2.2 Simplified Drude Model Based Simulation Method

Graphene has been extensively simulated via various models in last decade. Among these models, Drude model [11]–[14], within the framework of Boltzmann transport theories, is a widely used model and has shown good accuracy in a linear regime. In this section, we derive Drude model from the Boltzmann Transport Equation. We then present a numerical algorithm to apply the Drude model to simulate Cu-graphene hybrid interconnects in time domain.

2.2.1 Analytical Derivation of Drude Model from the Boltzmann Transport Equation

The distribution function of charge carriers in graphene, denoted by $f(\mathbf{r}, \mathbf{k}, t)$ in phase space (real $\mathbf{r}$-space and momentum $\mathbf{k}$-space), is governed by the following Boltzmann transport equation:

$$\mathbf{v} \cdot \nabla_{\mathbf{r}} f + \frac{q}{\hbar} \mathbf{E} \cdot \nabla_{\mathbf{k}} f + \frac{\partial f}{\partial t} = -\frac{f - f_0}{\tau}, \tag{2.1}$$

where $\mathbf{v} = d\mathbf{r}/dt$ is the velocity vector, $\mathbf{k} = \mathbf{p}/\hbar$ is the wave vector of Bloch wave in momentum space, q is the amount of charge in each carrier, $\mathbf{E}$ is the electric field intensity, and $\hbar$ is the Planck constant. The scattering term on the right hand side of (2.1) is approximated by the relaxation time approximation [26], where τ is the relaxation time, and f_0 is the Fermi-Dirac distribution at the equilibrium state as the following

$$f_0 = \left[1 + \exp\left(\frac{\xi - \xi_{\mathrm{F}}}{k_{\mathrm{B}} T}\right)\right]^{-1}, \tag{2.2}$$

in which ξ is the carrier's energy, ξ_{F} is the Fermi energy (also called Fermi level or chemical potential), k_{B} is the Boltzmann constant, and T is the temperature.

Given $f(\mathbf{r}, \mathbf{k}, t)$, the conduction current density $\mathbf{j}_g$ in graphene can be evaluated from an integration over $\mathbf{k}$-space as:

$$\mathbf{j}_g = \frac{g_s g_v q}{(2\pi)^d} \int_{\mathbf{k}} f \mathbf{v} d\mathbf{k}, \tag{2.3}$$

where g_s and g_v are spin, and valley degeneracy respectively, and d denotes the problem dimension which is 2 and 3 in a 2-, and 3-D analysis respectively. In order to calculate $\mathbf{j}_g$ from (2.1) and (2.3), the velocity vector $\mathbf{v}$ needs to be expressed as a function of $\mathbf{k}$. Semi-classically, by treating the Bloch waves as wave packets, the classical velocity $\mathbf{v}$ is defined as the group velocity $d\omega/d\mathbf{k}$ of such wave packets [26]. The frequency ω is associated with a wave function of energy ξ by quantum theory, $\omega = \xi/\hbar$, and hence $\mathbf{v} = \nabla_{\mathbf{k}}\xi/\hbar$. After substituting the following linear dispersion relation of graphene [27],

$$\xi = v_{\mathrm{F}}\hbar k, \tag{2.4}$$

where $v_{\mathrm{F}} = 10^6$ m/s is the Fermi velocity and $k = \sqrt{k_x^2 + k_y^2}$, we can express the velocity vector $\mathbf{v}$ as the following function of $\mathbf{k}$

$$\mathbf{v}(\mathbf{k}) = \nabla_{\mathbf{k}}\xi/\hbar = v_{\mathrm{F}}\hat{\mathbf{k}}. \tag{2.5}$$

To obtain the analytical Drude model of graphene from Boltzmann transport equation (2.1), three assumptions are made as follows:

1. The spatial variance of f is small, thus $\mathbf{v} \cdot \nabla_{\mathbf{r}} f \sim 0$;
2. The external fields are small, leading to a small change of the Fermi sea. Thus, $f \sim f_0$ in $\mathbf{k}$-space and $\frac{q}{\hbar}\mathbf{E} \cdot \nabla_{\mathbf{k}} f \sim \frac{q}{\hbar}\mathbf{E} \cdot \nabla_{\mathbf{k}} f_0$;
3. The nonlinear effect is negligible, thus, $\frac{\partial f}{\partial t}$ can be analyzed in frequency domain using $\mathrm{j}\omega\tilde{f}$.

Denoting $\tilde{f}$ and $\tilde{\mathbf{E}}$ as the frequency domain counterparts of $f(t)$ and $\mathbf{E}(t)$, the aforementioned three assumptions lead to a simplified Boltzmann transport equation in frequency domain as follows

$$\frac{q}{\hbar}\tilde{\mathbf{E}} \cdot \nabla_{\mathbf{k}} f_0 + \mathrm{j}\omega\tilde{f} = -\frac{\tilde{f} - f_0}{\tau}, \tag{2.6}$$

from which we can get an analytical expression of charge carrier distribution

$$\tilde{f} = \frac{1}{1 + \mathrm{j}\omega\tau}\left(f_0 - \frac{\tau q}{\hbar}\tilde{\mathbf{E}} \cdot \nabla_{\mathbf{k}} f_0\right). \tag{2.7}$$

Substituting the analytical $\tilde{f}$ in (2.7) and velocity $\mathbf{v}$ in (2.5) into (2.3), we obtain an analytical expression of conduction current density $\tilde{\mathbf{j}}_g$ in graphene

$$\tilde{\mathbf{j}}_g = \frac{\tilde{\mathbf{E}}}{1 + \mathrm{j}\omega\tau} \cdot \frac{\tau q}{\hbar}\frac{g_s g_v q}{(2\boldsymbol{\pi})^d} v_{\mathrm{F}} \int_{\mathbf{k}} (-\nabla_{\mathbf{k}} f_0)\hat{\mathbf{k}} d\mathbf{k} = \frac{\sigma_{dc}\tilde{\mathbf{E}}}{1 + \mathrm{j}\omega\tau}, \tag{2.8}$$

with

$$\sigma_{dc}\mathbf{I} = \frac{\tau q}{\hbar}\frac{g_s g_v q}{(2\pi)^d} v_{\mathrm{F}} \int_{\mathbf{k}} (-\nabla_{\mathbf{k}} f_0)\hat{\mathbf{k}} d\mathbf{k}. \tag{2.9}$$

Here, we have used the symmetry of the linear dispersion and the spherical Fermi sphere to simplify the integration. First, the integration of $f_0\mathbf{v}$ is 0 because f_0 is an even function in $\mathbf{k}$-space while $\mathbf{v}$ is an odd function. Second, the tensor $\int_{\mathbf{k}}(-\nabla_{\mathbf{k}} f_0)\hat{\mathbf{k}} d\mathbf{k}$ is isotropic due to the symmetry, thus can be written as a scaled identity matrix. As a result of the three assumptions and the use of special symmetry, graphene follows the Ohm's Law as shown in (2.8).

The analytical Drude model derived in the above yields the conductivity of graphene in frequency domain

$$\tilde{\sigma}_g(\omega) = \frac{\sigma_{dc}}{1 + \mathrm{j}\omega\tau}, \tag{2.10}$$

where ω is angular frequency, τ is the relaxation time as that in Boltzmann equation (2.1), and σ_{dc} is the DC conductivity of graphene. From open literature, it can be seen that σ_{dc} can be obtained in many ways, such as represented as (2.9) by utilizing the dispersion relation, represented by other parameters like the carrier density [14], and directly measured at low frequencies [13]. For the comparison concerned in this work between a first-principles based simulation and the Drude model based simulation, in order to use the same assumptions and parameters, σ_{dc} in Drude model is extracted from the numerical Boltzmann solver developed in this work.

Drude model (2.10) agrees with the intraband transition part of Kubo formula [5], which is a more accurate conductivity model of graphene accounting for both the intraband and interband transition. The intraband transition, as described by the above Boltzmann transport equation (2.1) and Drude model (2.10), is the transition of electron states near the Fermi surface in $\mathbf{k}$-space. The interband transition, corresponding to the electrons poping up from an inner band to an upper conduction band, can be obtained by employing the kramers-kronig relation or Fermi's golden rule [28]. For ICs, the intraband transition is the dominant effect because both thermal excitation energy ($T = 300$ K, $k_{\mathrm{B}}T \sim 25$ meV) and photon energy ($\omega = 10$ GHz, $\hbar\omega \sim 6.6 \times 10^{-6}$ eV) are much smaller than the Fermi energy of graphene (typically 0.21 eV). The electrons at an inner band can hardly find enough excitation energy to pop up to an upper band, thus, the interband transition is suppressed. In this work, focusing on simulating on-chip Cu-graphene interconnects, we only consider the intraband transition using either the Boltzmann equation or the Drude model.

2.2.2 Accounting for Drude Model in Time Domain Analysis

Based on the Drude model,

$$\tilde{\mathbf{j}}_g(\omega) = \tilde{\sigma}_g(\omega)\tilde{\mathbf{E}}(\omega) = \frac{\sigma_{dc}}{1 + \mathrm{j}\omega\tau}\tilde{\mathbf{E}}(\omega). \tag{2.11}$$

By multiplying $1 + \mathrm{j}\omega\tau$ to both sides and replacing the $\mathrm{j}\omega$ with $\partial/\partial t$, the equation for $\mathbf{j}_g(t)$ in time domain can be found as

$$\mathbf{j}_g(t) + \tau\frac{\partial \mathbf{j}_g(t)}{\partial t} = \sigma_{dc}\mathbf{E}(t). \tag{2.12}$$

Using a backward difference to discretize the time derivative, we obtain

$$\{\mathrm{j}_g\}^{n+1} + \tau\frac{\{\mathrm{j}_g\}^{n+1} - \{\mathrm{j}_g\}^{n}}{\Delta t} = \sigma_{dc}\{\mathrm{e}\}^n, \tag{2.13}$$

from which we have the following time-domain update equation for the current density in graphene

$$\{\mathrm{j}_g\}^{n+1} = \left(\sigma_{dc}\{\mathrm{e}\}^n + \frac{\tau}{\Delta t}\{\mathrm{j}_g\}^n\right) / \left(\frac{\tau}{\Delta t} + 1\right), \tag{2.14}$$

which is then used in conjunction with the FDTD to simulate Cu-graphene interconnects.

2.2.3 Drude Model in Conjunction with the FDTD Algorithm for Simulating On-Chip Cu-Graphene Hybrid Interconnects

The electrical performance of a Cu-G interconnect is governed by the following Maxwell's equations from DC to high frequencies:

$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t}, \tag{2.15a}$$

$$\nabla \times \mathbf{H} = \epsilon \frac{\partial \mathbf{E}}{\partial t} + \sigma \mathbf{E} + \mathbf{j}_{\mathrm{i}}, \tag{2.15b}$$

where $\mathbf{E}$ is electric field intensity, $\mathbf{H}$ is magnetic field intensity, $\mathbf{j}_{\mathrm{i}}$ is input (supply) current density, μ, ϵ, and σ are permeability, permittivity, and conductivity respectively.

In this work, we apply an implicit unconditionally stable time domain scheme developed in [20] to an FDTD-based discretization of Maxwell's equations. In this method, we discretize Maxwell's equations (2.17) as:

$$\mathbf{S}_{\mathrm{e}}\{\mathrm{e}\}^{n+1} = -\,\mathbf{D}_{\mu}\frac{\{h\}^{n+\frac{1}{2}} - \{h\}^{n-\frac{1}{2}}}{\Delta t}, \tag{2.16a}$$

$$\begin{aligned}\mathbf{S}_{h}\{h\}^{n+\frac{1}{2}} =& \mathbf{D}_{\epsilon}\frac{\{\mathrm{e}\}^{n+1} - \{\mathrm{e}\}^{n}}{\Delta t} + \mathbf{D}_{\sigma}\{\mathrm{e}\}^{n+1} + \\ & \{\mathrm{j}_{g}\}^{n+1} + \{\mathrm{j}_{\mathrm{i}}\}^{n+1},\end{aligned} \tag{2.16b}$$

where $\{\mathrm{e}\}^{n}$ represents the vector of electric fields at the n-th time instant, $\{h\}^{n+\frac{1}{2}}$ represents the vector of magnetic fields at the $n + \frac{1}{2}$ time instant, and $\{\mathrm{j}_g\}^{n+1}$ represents the vector of conduction current densities in graphene layers, which is obtained from (2.14). In (2.16), $\{\mathrm{j}_{\mathrm{i}}\}$ denotes a vector of input current densities, $\mathbf{D}_{\mu}$, and $\mathbf{D}_{\epsilon}$, and $\mathbf{D}_{\sigma}$ are diagonal matrices of permeability, permittivity, and conductivity (for the non-graphene region) respectively. The matrix-vector products $\mathbf{S}_{\mathrm{e}}\{\mathrm{e}\}$ and $\mathbf{S}_{h}\{h\}$ represent discretized $\nabla \times \mathbf{E}$ and $\nabla \times \mathbf{H}$. The $\mathbf{S}_{\mathrm{e}}$ and $\mathbf{S}_{h}$ can be readily constructed using a single-grid patch based FDTD formulation developed in [29].

The Drude model based simulation algorithm is realized by substituting the current density $\{\mathrm{j}_g\}^{n+1}$ in (2.14) into the right hand side of Maxwell solver (2.16). The procedure, written in a pseudo-code, is shown in Algorithm 1.

Algorithm 1 Drude Model + FDTD

```
1: Set excitation and boundary conditions
2: Initialize electromagnetic fields ${\rm e}\}^1$ & $\{h\}^{\frac{1}{2}}$
3: for time step $n := 1$ to $n_{max}$ do
4:     Update $\{{\rm j}_g\}^{n+1}$ with $\{{\rm j}_g\}^n$ & $\{{\rm e}\}^n$
5:     Update $\begin{bmatrix}\{{\rm e}\}^{n+1}\\ \{h\}^{n+\frac{1}{2}}\end{bmatrix}$ with $\begin{bmatrix}\{{\rm e}\}^{n}\\ \{h\}^{n-\frac{1}{2}}\end{bmatrix}$ & $\{{\rm j}_g\}^{n+1}$
6: end for
```

2.3 Proposed Multiphysics Modeling of Cu-G Hybrid Nano-Interconnects

As shown in previous section, a Drude model based simulation relies on a few simplifications, which can miss the dynamic nonlinear physics at high frequencies, including both the nonlinear buildup of the conduction current in graphene and the nonlinear coupling between the electric field and electrons inside graphene layers. Therefore, an accurate model requires a direct observation of the charge carriers in graphene, thereby requires a direct solution of the charge carrier distribution function $f(\mathbf{r}, \mathbf{k}, t)$ through Boltzmann transport equation. In this section, we present a first-principles based multiphysics modeling and simulation algorithm, co-simulating directly in time domain Maxwell's equations, equations characterizing graphene materials, and Boltzmann equation from DC to high frequencies.

The electromagnetic performance of a Cu-G interconnect is governed by Maxwell's equations from DC to high frequencies:

$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t}, \tag{2.17a}$$

$$\nabla \times \mathbf{H} = \epsilon \frac{\partial \mathbf{E}}{\partial t} + \sigma \mathbf{E} + \mathbf{j}_{\rm i}, \tag{2.17b}$$

where $\mathbf{E}$ is electric field intensity, $\mathbf{H}$ is magnetic field intensity, $\mathbf{j}_{\rm i}$ is input (supply) current density, μ, ϵ, and σ are permeability, permittivity, and conductivity respectively.

When considering the existence of graphene layers, especially their conduction current density $\mathbf{j}_g = \sigma \mathbf{E}$ in changing the entire electromagnetic response, conventional simplified steady-state σ models [4], [6] can miss the dynamic nonlinear physics at high frequencies, including both the nonlinear buildup of the conduction current in graphene and the nonlinear coupling between the external field and electron behavior inside graphene layers. Therefore, an accurate model requires a direct observation of the charge carriers in graphene, which is described by the distribution function $f(\mathbf{r}, \mathbf{k}, t)$ in phase space (real $\mathbf{r}$-space and momentum $\mathbf{k}$-space). Based on first principles, $f(\mathbf{r}, \mathbf{k}, t)$ is governed by the following Boltzmann equation:

$$\mathbf{v} \cdot \nabla_{\mathbf{r}} f + \frac{q}{\hbar} \mathbf{E} \cdot \nabla_{\mathbf{k}} f + \frac{\partial f}{\partial t} = -\frac{f - f_0}{\tau}, \tag{2.18}$$

where $\mathbf{v} = d\mathbf{r}/dt$ is the velocity vector, $\mathbf{k} = \mathbf{p}/\hbar$ is the wave vector of Bloch wave in momentum space, q is the amount of charge in each carrier, and $\hbar$ is the Planck constant. The magnetic effects in Boltzmann equation are not considered here as they are much smaller than electric effects in IC interconnects. The scattering term on the right hand side of Boltzmann equation is approximated by the relaxation time approximation [26], where τ is the relaxation time, and f_0 is the Fermi-Dirac distribution at the equilibrium state

$$f_0 = \left[1 + \exp\left(\frac{\xi - \xi_{\mathrm{F}}}{k_{\mathrm{B}} T}\right)\right]^{-1}, \tag{2.19}$$

in which ξ is the carrier's energy, ξ_{F} is the Fermi energy (also called Fermi level or chemical potential), k_{B} is the Boltzmann constant, and T is the temperature.

Given $f(\mathbf{r}, \mathbf{k}, t)$, the conduction current density $\mathbf{j}_g$ in graphene can be evaluated from an integration over $\mathbf{k}$-space as:

$$\mathbf{j}_g = \frac{g_s g_v q}{(2\pi)^d} \int_{\mathbf{k}} f \mathbf{v} d\mathbf{k}, \tag{2.20}$$

where g_s and g_v are spin and valley degeneracy, respectively, and d denotes the problem dimension which is 2 and 3 in a 2- and 3-D analysis respectively. In order to calculate $\mathbf{j}_g$ from (2.18) and (2.20), the velocity vector $\mathbf{v}$ needs to be expressed as a function of $\mathbf{k}$. Semi-classically, by treating the Bloch waves as wave packets, the classical velocity $\mathbf{v}$ is defined as the group velocity $d\omega/d\mathbf{k}$ of such wave packets [26]. The frequency ω is associated with a wave function of energy ξ by quantum theory, $\omega = \xi/\hbar$, and hence

$$\mathbf{v} = \nabla_{\mathbf{k}} \xi / \hbar. \tag{2.21}$$

After substituting the following linear dispersion relation of graphene [27], which is illustrated in Fig. 2.1(b),

$$\xi = v_{\mathrm{F}}\hbar k, \tag{2.22}$$

where $v_{\mathrm{F}} = 10^6$ m/s is the Fermi velocity and $k = \sqrt{k_x^2 + k_y^2}$, we can express the velocity vector $\mathbf{v}$ as the following function of $\mathbf{k}$

$$\mathbf{v}(\mathbf{k}) = \nabla_{\mathbf{k}}\xi/\hbar = v_{\mathrm{F}}\hat{\mathbf{k}}, \tag{2.23}$$

with $\mathbf{k} = k_x\hat{x} + k_y\hat{y}$, and $\hat{\mathbf{k}} = \mathbf{k}/|k|$ being the unit vector along the direction of $\mathbf{k}$.

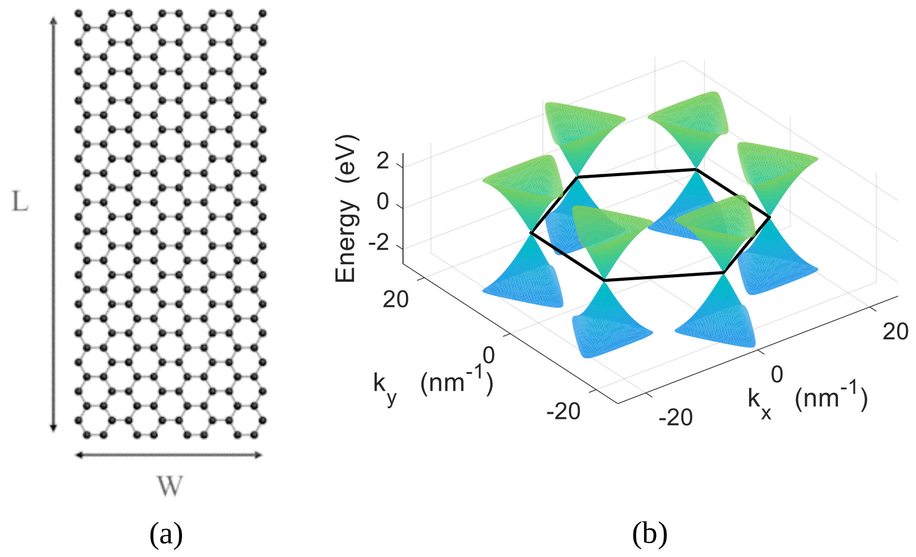


Figure 2.1. (a) Structure of a single layer graphene with length L and width W. (b) Linear dispersion of graphene. Dirac cones are located at the six corners of the hexagonal Brillouin zone. Therefore, the valley degeneracy $g_v = 2$.

The proposed system of equations, which governs the electromagnetic performance of Cu-G interconnects, consists of three sets of first-principle equations, namely Maxwell's equations (2.17), Boltzmann equation (2.18), and the dispersion relation of graphene (2.22). Because the carrier distribution function f is a function of $\mathbf{r}$, $\mathbf{k}$, and t, the computational domain for this model has seven dimensions in 3-D analyses and five dimensions in 2-D

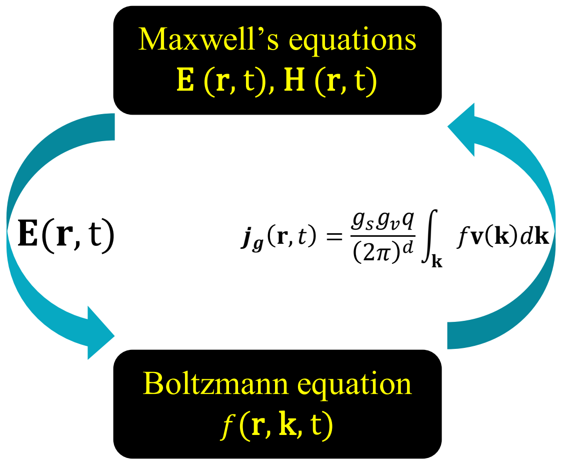

Maxwell's equations
$\mathbf{E}(\mathbf{r}, t), \mathbf{H}(\mathbf{r}, t)$
$\mathbf{E}(\mathbf{r}, t)$
$\boldsymbol{j_g}(\mathbf{r}, t) = \frac{g_s g_v q}{(2\pi)^d} \int_{\mathbf{k}} f\mathbf{v}(\mathbf{k}) d\mathbf{k}$
Boltzmann equation
$f(\mathbf{r}, \mathbf{k}, t)$

Figure 2.2. Illustration of the co-simulation flow.

analyses. A flow of the co-simulation of these equations in time domain is illustrated in Fig. 2.2. Given an external source and initial conditions, Maxwell's equations (2.17) are solved to obtain electric field $\mathbf{E}(\mathbf{r}, t)$, using which Boltzmann equation (2.18) can be solved to obtain charge carrier distribution $f(\mathbf{r}, \mathbf{k}, t)$. From integrating $f(\mathbf{r}, \mathbf{k}, t)$ over $\mathbf{k}$-space as shown in (2.20), the conduction current density $\mathbf{j}_g(\mathbf{r}, t)$ in graphene layers is calculated at each space point. At next time instant, graphene's conduction current density term $\sigma\mathbf{E}$ in Maxwell's equations (2.17) is replaced by latest $\mathbf{j}_g(\mathbf{r}, t)$, while the conduction current density in other conducting materials is still updated using $\sigma\mathbf{E}$. Now, with all the current updated, Maxwell's equations (2.17) are ready to be solved again. The whole process continues until a desired time is reached or until the physical phenomenon happening in a Cu-G interconnect has reached its steady state.

2.4 Multiphysics Co-Simulation in Time Domain and Stability Analysis

There are two major challenges in the multiphysics simulation of Cu-G interconnects. The first challenge is that Boltzmann equation (2.18) is a seven-dimensional equation in a 3-D analysis, which is computationally expensive. To reduce the computational cost, we utilize the fact that graphene is a 2-D material, hence we can solve a 2-D version of Boltzmann equation (2.18) in conjunction with the 3-D Maxwell's equations. However, even using a 2-D Boltzmann equation, there are five dimensions involved, making the simulation of Boltzmann subsystem much slower than that of the Maxwell subsystem. The second challenge arises from the small size of nano-interconnects, which results in a large number of time steps to finish one simulation using most explicit solvers. To address this problem, we develop an unconditionally stable co-simulation algorithm to remove the dependence of time step on space step.

2.4.1 Unconditionally Stable Time-Marching Scheme of the Maxwell Subsystem

In this work, we apply an implicit unconditionally stable time domain scheme developed in [20] to an Finite Difference Time Domain (FDTD)-based discretization of Maxwell's equations. This scheme is theoretically proved to be unconditionally stable for general problem settings having arbitrary structures and inhomogeneous materials. In this method, we discretize Maxwell's equations (2.17) as:

$$\mathbf{S}_{\mathrm{e}}\{\mathrm{e}\}^{n+1} = -\,\mathbf{D}_{\mu}\frac{\{h\}^{n+\frac{1}{2}}-\{h\}^{n-\frac{1}{2}}}{\Delta t}, \tag{2.24a}$$

$$\begin{aligned}\mathbf{S}_{h}\{h\}^{n+\frac{1}{2}} =& \mathbf{D}_{\epsilon}\frac{\{\mathrm{e}\}^{n+1}-\{\mathrm{e}\}^{n}}{\Delta t}+\mathbf{D}_{\sigma}\{\mathrm{e}\}^{n+1} \\ &+\{\mathrm{j}_g\}^{n}+\{\mathrm{j}_{\mathrm{i}}\}^{n+1},\end{aligned} \tag{2.24b}$$

where $\{\mathrm{e}\}^n$ represents the vector of electric field intensities at the n-th time instant, $\{h\}^{n+\frac{1}{2}}$ represents the vector of magnetic field intensities at the $n+\frac{1}{2}$ time instant, $\{\mathrm{j}_g\}$ represents the vector of conduction current densities in graphene layers, $\{\mathrm{j}_{\mathrm{i}}\}$ represents input current densities, $\mathbf{D}_{\mu}$, $\mathbf{D}_{\epsilon}$, and $\mathbf{D}_{\sigma}$ are diagonal matrices of permeability, permittivity, and conductivity respectively. The matrix-vector products $\mathbf{S}_{\mathrm{e}}\{\mathrm{e}\}$ and $\mathbf{S}_{h}\{h\}$ represent discretized $\nabla\times\mathbf{E}$ and $\nabla\times\mathbf{H}$. The $\mathbf{S}_{\mathrm{e}}$ and $\mathbf{S}_{h}$ can be readily constructed using a single-grid patch based FDTD formulation developed in [29].

If we eliminate the $\{h\}$ in (2.24), we will end up with the following backward-difference based discretization of the second order vector wave equation for $\mathbf{E}$ if $\{\mathrm{j}_g\}$ is not considered

$$\begin{aligned}\{\mathrm{e}\}^{n+1} - 2\{\mathrm{e}\}^{n} + \{\mathrm{e}\}^{n-1} + \Delta t \mathbf{D}_\epsilon^{-1}\mathbf{D}_\sigma(\{\mathrm{e}\}^{n+1} - \{\mathrm{e}\}^{n}) \\ +\Delta t^2 \mathbf{D}_\epsilon^{-1}\mathbf{S}_h\mathbf{D}_\mu^{-1}\mathbf{S}_\mathrm{e}\{\mathrm{e}\}^{n+1} = -\Delta t^2 \mathbf{D}_\epsilon^{-1}\left(\frac{\partial\{\mathrm{j}\}}{\partial t}\right)^{n+1}.\end{aligned} \tag{2.25}$$

Discarding the source term since it has nothing to do with the stability, and performing a z-transform of the above time marching equation, we can find

$$|z| = \frac{1}{\sqrt{1+\Delta t^2\lambda}}, \tag{2.26}$$

where λ is the eigenvalue of $\mathbf{D}_\epsilon^{-1}\mathbf{S}_h\mathbf{D}_\mu^{-1}\mathbf{S}_\mathrm{e}$. Since in an FDTD method, $\mathbf{S}_h = \mathbf{S}_\mathrm{e}^T$ is satisfied in a uniform grid [29], the eigenvalues of $\mathbf{D}_\epsilon^{-1}\mathbf{S}_h\mathbf{D}_\mu^{-1}\mathbf{S}_\mathrm{e}$ are always nonnegative. Substituting $\lambda \geq 0$ into (2.26), it can be readily found that z's modulus is always bounded by 1 regardless of Δt. Hence, the time marching of (2.25) is ensured to be unconditionally stable. Although it appears that we have to solve a matrix in the time marching, using the scheme developed in [20], this matrix's inverse can be explicitly found, thus avoiding a matrix solution.

The updating from one time step to the next in (2.24) can also be rewritten as

$$\mathbf{M}_A\{x\}^{n+1} = \mathbf{M}_B\{x\}^{n} + \{b_\mathrm{j}\}^{n+1}, \tag{2.27}$$

where

$$\{x\}^n = \begin{bmatrix}\{\mathrm{e}\}^n \\ \{h\}^{n-\frac{1}{2}}\end{bmatrix} \text{ and } \{b_\mathrm{j}\}^{n+1} = \begin{bmatrix}-\{\mathrm{j}_g\}^n - \{\mathrm{j}_\mathrm{i}\}^{n+1} \\ 0\end{bmatrix},$$

and

$$\mathbf{M}_A = \begin{bmatrix}\frac{\mathbf{D}_\epsilon}{\Delta t} + \mathbf{D}_\sigma & -\mathbf{S}_h \\ \mathbf{S}_\mathrm{e} & \frac{\mathbf{D}_\mu}{\Delta t}\end{bmatrix} \text{ and } \mathbf{M}_B = \begin{bmatrix}\frac{\mathbf{D}_\epsilon}{\Delta t} & 0 \\ 0 & \frac{\mathbf{D}_\mu}{\Delta t}\end{bmatrix}.$$

2.4.2 Unconditionally Stable Time-Marching Scheme of the Boltzmann Subsystem

The high dimensionality of the phase space makes solving the Boltzmann equation (2.18) a challenging task. One of the biggest obstacles for a deterministic Boltzmann solver is the requirement of huge memory. To resolve the memory issue, the last few decades have seen many efforts along two major directions for solving the Boltzmann equation. One direction is the Monte Carlo approach, where the Boltzmann equation is solved by simulating a stochastic process [30]–[32]. Another direction is to expand the distribution function f with basis functions in $\mathbf{k}$-space, and then truncate the expansion to the first few terms according to the accuracy. The commonly used expansions are spherical harmonics expansion [33] and Fourier harmonics expansion in quantized $\mathbf{k}$-space [34]. Both of the two directions can reduce the required memory by a few orders. But the disadvantages are 1) the simplification of the original Boltzmann equation, and 2) the requirement of a self-iterative solver to determine a few key parameters like the expansion coefficient. These existing Boltzmann solvers can hardly provide the dynamic time-domain nonlinear transition we want to capture from the original Boltzmann equation (2.18). Therefore, in this work, we develop a direct deterministic Boltzmann solver for the Cu-G system. The advancement of Dynamic Random-Access Memory (DRAM) technology and today's computers has greatly alleviated the limitation from huge memory requirement. On the other hand, the 2-D nature of graphene reduces the dimension of phase space from six to four. These two factors make a direct deterministic Boltzmann solver feasible for the Cu-G hybrid nano-interconnects.

However, the direct Boltzmann solver still needs to be carefully developed to resolve two challenges. First, the extremely small space step could require an extremely small time step due to the stability requirement. For example, in one Cu-G hybrid nano-interconnect to be shown later, a conditionally stable Boltzmann solver can require millions of time steps to finish one run. Because of the expensive computational cost for the Boltzmann subsystem, the need for simulating many time steps can significantly degrade the efficiency of the simulation. Second, the coupling with the Maxwell subsystem should not ruin the stability, or should even maintain the global unconditional stability of the entire system. Both of the challenges are solved with the following unconditionally stable Boltzmann solver.

Unconditionally Stable Boltzmann Solver

Substituting (2.23) into (2.18), the 2-D Boltzmann equation for graphene in the 4-D phase space $(x - y - k_x - k_y)$ becomes

$$\frac{v_\mathrm{F}}{k}(k_x\frac{\partial f}{\partial x} + k_y\frac{\partial f}{\partial y}) + \frac{q}{\hbar}(E_x\frac{\partial f}{\partial k_x} + E_y\frac{\partial f}{\partial k_y}) + \frac{\partial f}{\partial t} = -\frac{f - f_0}{\tau}. \tag{2.28}$$

In terms of the discretization of the derivatives, the independence among $\mathbf{r}$, $\mathbf{k}$, and t allows us to consider each first order derivative independently. First of all, $\nabla_\mathbf{r}$ is discretized with a central difference to maintain the same accuracy $\sim \mathcal{O}(\Delta r^2)$ as that in FDTD. Second, $\nabla_\mathbf{k}$ is also discretized with a central difference to align with the $\nabla_\mathbf{r}$. Mathematically, the role of $\mathbf{r}$ and $\mathbf{k}$ in Boltzmann equation (2.28) can be exchanged without changing the equation much. Thus, aligning the numerical treatment of $\mathbf{r}$ and $\mathbf{k}$ can simplify the system matrices and thereby the solution of the Boltzmann subsystem. Having $\nabla_\mathbf{r}$ and $\nabla_\mathbf{k}$ discretized with the central difference in phase space, the remaining $\partial/\partial t$ could be discretized in time with a backward difference to guarantee the unconditional stability. As a result, we obtain

$$(\mathbf{S}_r + \mathbf{S}_k^n)\{f\}^{n+1} + \frac{\{f\}^{n+1} - \{f\}^n}{\Delta t} = \frac{\{f_0\} - \{f\}^{n+1}}{\tau}, \tag{2.29}$$

where $\{f\}^n$ is the vector of carrier distribution function at the n-th time instant, $\mathbf{S}_r\{f\}$ and $\mathbf{S}_k^n\{f\}$ represent discretized $\mathbf{v} \cdot \nabla_\mathbf{r} f$ and $\frac{q}{\hbar}\mathbf{E} \cdot \nabla_\mathbf{k} f$, respectively. Here, the superscript n of $\mathbf{S}_k^n$ denotes the time instant of $\mathbf{E}$ used to obtain $\mathbf{S}_k^n$. The grid used for discretizing the Maxwell's equations is also used for solving the Boltzmann subsystem, and the f is assigned at the $\mathbf{H}$'s points. The electric field $\mathbf{E}$ used in the Boltzmann equation is center-averaged by neighboring $\mathbf{E}$ fields in the grid. The matrix-based expression here follows a similar logic as that in the Maxwell subsystem. All local $f(\mathrm{i},\mathrm{j},\mathrm{i}_k,\mathrm{j}_k)$ in the 4-D phase space are reorganized and labeled with a global index

$$m_{\mathrm{i},\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k} = \mathrm{j}_k N_x N_y N_{k_x} + \mathrm{i}_k N_x N_y + \mathrm{j} N_x + \mathrm{i}, \tag{2.30}$$

in which N_x, N_y, and N_{k_x} are the number of nodes along the x-, y-, and k_x-directions, respectively. The local index $(\mathrm{i},\mathrm{j},\mathrm{i}_k,\mathrm{j}_k)$ means the position in phase space is at $(x = \mathrm{i}\Delta x + x_0, y = \mathrm{j}\Delta y + y_0, k_x = \mathrm{i}_k \Delta k_x + k_{x0}, k_y = \mathrm{j}_k \Delta k_y + k_{y0})$, where $(\Delta x, \Delta y, \Delta k_x, \Delta k_y)$ and $(x_0, y_0, k_{x0}, k_{y0})$ denote the cell size, and starting point along each dimension. Thus, each local $f(\mathrm{i},\mathrm{j},\mathrm{i}_k,\mathrm{j}_k)$ becomes the $m_{\mathrm{i},\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k}$-th element $f_{m_{\mathrm{i},\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k}}$ in vector $\{f\}$.

The entries of two matrices $\mathbf{S}_r$ and $\mathbf{S}_k^n$, using a central difference in a uniform grid, can be analytically extracted as the following. Take $\mathbf{S}_r\{f\} \sim \mathbf{v}\cdot\nabla_{\mathbf{r}} f$ as an example. Since in $\mathbf{v}\cdot\nabla_{\mathbf{r}} f$, we use nearby f values to generate a value $\tilde{f}$ at a local position $(\mathrm{i},\mathrm{j},\mathrm{i}_k,\mathrm{j}_k)$, the central-difference formula written in local indices is

$$\begin{aligned}\tilde{f}(\mathrm{i},\mathrm{j},\mathrm{i}_k,\mathrm{j}_k) = v_x(\mathrm{i}_k,\mathrm{j}_k)&\frac{f(\mathrm{i}+1,\mathrm{j},\mathrm{i}_k,\mathrm{j}_k) - f(\mathrm{i}-1,\mathrm{j},\mathrm{i}_k,\mathrm{j}_k)}{2\Delta x}\\ +v_y(\mathrm{i}_k,\mathrm{j}_k)&\frac{f(\mathrm{i},\mathrm{j}+1,\mathrm{i}_k,\mathrm{j}_k) - f(\mathrm{i},\mathrm{j}-1,\mathrm{i}_k,\mathrm{j}_k)}{2\Delta y}.\end{aligned}$$

This formula, if written in global indices (2.30), becomes

$$\begin{aligned}\tilde{f}_{m_{\mathrm{i},\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k}} =& v_{x_{m_{0,0}^{\mathrm{i}_k,\mathrm{j}_k}}} \frac{f_{m_{\mathrm{i}+1,\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k}} - f_{m_{\mathrm{i}-1,\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k}}}{2\Delta x}\\ &+ v_{y_{m_{0,0}^{\mathrm{i}_k,\mathrm{j}_k}}} \frac{f_{m_{\mathrm{i},\mathrm{j}+1}^{\mathrm{i}_k,\mathrm{j}_k}} - f_{m_{\mathrm{i},\mathrm{j}-1}^{\mathrm{i}_k,\mathrm{j}_k}}}{2\Delta y},\end{aligned} \tag{2.31}$$

which is simply a row of the matrix-based expression $\{\tilde{f}\} = \mathbf{S}_r\{f\} \sim \mathbf{v}\cdot\nabla_{\mathbf{r}} f$. Since $\mathbf{v}$ is independent of $\mathbf{r}$ for graphene here, the (i,j) indices for $\mathbf{v}$ are denoted as $(0,0)$. The elements of matrix $\mathbf{S}_r$ hence can be directly extracted from (2.31) as

$$\begin{aligned}\mathbf{S}_{r_{m_{\mathrm{i},\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k}, m_{\mathrm{i}+1,\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k}}} &= v_{x_{m_{0,0}^{\mathrm{i}_k,\mathrm{j}_k}}}/(2\Delta x) = -\mathbf{S}_{r_{m_{\mathrm{i},\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k}, m_{\mathrm{i}-1,\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k}}},\\ \mathbf{S}_{r_{m_{\mathrm{i},\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k}, m_{\mathrm{i},\mathrm{j}+1}^{\mathrm{i}_k,\mathrm{j}_k}}} &= v_{y_{m_{0,0}^{\mathrm{i}_k,\mathrm{j}_k}}}/(2\Delta y) = -\mathbf{S}_{r_{m_{\mathrm{i},\mathrm{j}}^{\mathrm{i}_k,\mathrm{j}_k}, m_{\mathrm{i},\mathrm{j}-1}^{\mathrm{i}_k,\mathrm{j}_k}}}.\end{aligned} \tag{2.32}$$

For the other matrix $\mathbf{S}_k^n\{f\} \sim \frac{q}{\hbar}\mathbf{E} \cdot \nabla_{\mathbf{k}} f$, we can find its elements similarly by exchanging $\mathbf{E}$ to $\mathbf{v}$ and $\mathbf{k}$ to $\mathbf{r}$.

The proposed time-marching formula for the Boltzmann subsystem (2.29) is

$$\mathbf{B}^n\{f\}^{n+1} = \{f\}^n + \{\tilde{f}_0\}, \tag{2.33}$$

where the constant term $\{\tilde{f}_0\} = \{f_0\}\Delta t/\tau$ and the system matrix

$$\mathbf{B}^n = (1 + \Delta t/\tau)\mathbf{I} + \Delta t(\mathbf{S}_r + \mathbf{S}_k^n). \tag{2.34}$$

Proof on the Unconditional Stability of the Boltzmann Solver and the Choice of Time Step

To analyze the stability, the eigenvalues of matrix $\mathbf{B}^n$, thereby the eigenvalues of $\mathbf{S}_r$ and $\mathbf{S}_k^n$, should be studied. Here, we first prove that both $\mathbf{S}_r$ and $\mathbf{S}_k^n$, with a central difference in a uniform grid, are skew-symmetric. Still take $\mathbf{S}_r\{f\} \sim \mathbf{v} \cdot \nabla_{\mathbf{r}} f$ as an example. From the matrix elements in (2.32), the rotated counterpart of matrix element $\mathbf{S}_{r_{m_{\mathrm{i,j}}^{\mathrm{i}_k,\mathrm{j}_k}, m_{\mathrm{i+1,j}}^{\mathrm{i}_k,\mathrm{j}_k}}}$ is $\mathbf{S}_{r_{m_{\mathrm{i+1,j}}^{\mathrm{i}_k,\mathrm{j}_k}, m_{\mathrm{i,j}}^{\mathrm{i}_k,\mathrm{j}_k}}}$, whose value (by shifting i to i + 1 in the right hand side of (2.32)) is the opposite of $\mathbf{S}_{r_{m_{\mathrm{i,j}}^{\mathrm{i}_k,\mathrm{j}_k}, m_{\mathrm{i+1,j}}^{\mathrm{i}_k,\mathrm{j}_k}}}$. The same procedure can be applied to j. Thus, the skew-symmetry of $\mathbf{S}_r$ is proved. The key factors to the skew-symmetry are 1) the velocity $\mathbf{v}$ is independent of $\mathbf{r}$-space and 2) the $\mathbf{r}$-space is discretized uniformly along each direction. These two together guarantee the matrix elements in (2.32) to be the same regardless of the choice of (i, j). For the other matrix $\mathbf{S}_k^n\{f\} \sim \frac{q}{\hbar}\mathbf{E} \cdot \nabla_{\mathbf{k}} f$, we can exchange $\mathbf{E}$ to $\mathbf{v}$ and $\mathbf{k}$ to $\mathbf{r}$, and end up with a similar proof.

As a result of skew-symmetry, the eigenvalues of $\mathbf{S}_r + \mathbf{S}_k^n$ are purely imaginary [35]. From the expression of matrix $\mathbf{B}^n$ in (2.34), we can see clearly that its eigenvalues are

$$\lambda(\mathbf{B}^n) = 1 + \Delta t/\tau + \Delta t \lambda(\mathbf{S}_r + \mathbf{S}_k^n).$$

Since $\lambda(\mathbf{S}_r + \mathbf{S}_k^n)$ are purely imaginary, we have

$$|\lambda(\mathbf{B}^n)| = \sqrt{(1 + \Delta t/\tau)^2 + \Delta t^2 |\lambda(\mathbf{S}_r + \mathbf{S}_k^n)|^2} \geq 1. \tag{2.35}$$

Hence, the amplification factor of Boltzmann subsystem $\mathbf{B}^n\{f\}^{n+1} = \{f\}^n$ is bounded by 1, regardless of the choice of time step. As a result, we prove the proposed time marching of Boltzmann subsystem is unconditionally stable.

The unconditional stability allows for a choice of any large time step without affecting stability. Hence, in real simulations, the time step can be solely chosen according to the accuracy requirement. The relaxation time approximation in Boltzmann equation (2.18) assumes an exponential decay with a relaxation time τ. Therefore, the physical process gives the Boltzmann subsystem a characteristic time constant τ. According to the sampling theorem, an accurate time step to capture the time constant τ in Boltzmann subsystem would be

$$\Delta t \leq \tau/10. \tag{2.36}$$

2.4.3 Unconditionally Stable Time-Marching Scheme of the Coupled System

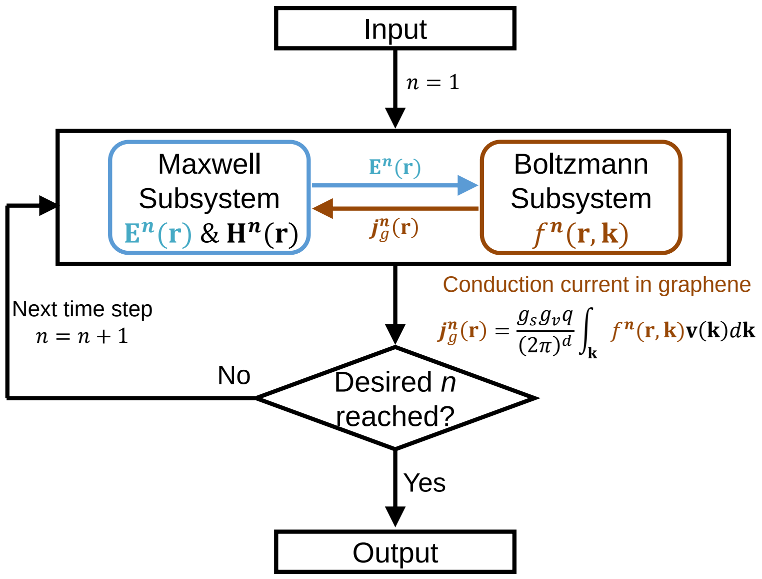

Input
$n = 1$
Maxwell Subsystem
$\mathbf{E}^n(\mathbf{r})$ & $\mathbf{H}^n(\mathbf{r})$
$\mathbf{E}^n(\mathbf{r})$
$j_g^n(\mathbf{r})$
Boltzmann Subsystem
$f^n(\mathbf{r},\mathbf{k})$
Conduction current in graphene
$j_g^n(\mathbf{r}) = \frac{g_s g_v q}{(2\pi)^d}\int_{\mathbf{k}} f^n(\mathbf{r},\mathbf{k})\mathbf{v}(\mathbf{k})d\mathbf{k}$
Next time step
$n = n + 1$
No
Desired n reached?
Yes
Output

Figure 2.3. Flowchart of the proposed multiphysics simulation algorithm, where the electromagnetic fields **E** & **H** and dynamic charge distribution f are updated at every time step.

As for the coupling between the Maxwell subsystem and the Boltzmann subsystem as seen in Fig. 2.3, the Boltzmann subsystem directly uses the electric field intensity $\mathbf{E}$ from the Maxwell subsystem, whereas the Maxwell subsystem uses, indirectly from the Boltzmann subsystem, the conduction current density $\{\mathrm{j}_g\}^n$ in graphene layers. The $\{\mathrm{j}_g\}^n$ is evaluated from $\{f\}^n$ through the integration of (2.20), which is numerically evaluated from a trapezoidal integration rule to maintain the second-order accuracy in the truncated $\mathbf{k}$-space. For a surface conduction current density, the x-component of (2.20) can be written as

$$\mathrm{j}^n_{gx\_2D} = \frac{g_s g_v q}{(2\pi)^2} \int_{k_x} \int_{k_y} f^n v_x dk_x dk_y, \tag{2.37}$$

a 2-D trapezoidal integration of which yields

$$\begin{aligned}\mathrm{j}^n_{gx\_2D}(\mathrm{i},\mathrm{j}) = \frac{g_s g_v q}{(2\pi)^2} \frac{\Delta k_x \Delta k_y}{4} \sum_{\mathrm{i}_k=0}^{N_{k_x}-1} \sum_{\mathrm{j}_k=0}^{N_{k_y}-1} \\ \alpha(\mathrm{i}_k,\mathrm{j}_k) f^n(\mathrm{i},\mathrm{j},\mathrm{i}_k,\mathrm{j}_k) v_x(\mathrm{i}_k,\mathrm{j}_k),\end{aligned} \tag{2.38}$$

where the coefficient $\alpha(\mathrm{i}_k,\mathrm{j}_k) = 4$ inside the k_x-k_y grid, $\alpha(\mathrm{i}_k,\mathrm{j}_k) = 2$ on the four outermost boundaries of the grid, and $\alpha(\mathrm{i}_k,\mathrm{j}_k) = 1$ at four corners of the grid. The y-component of $\mathrm{j}^n_{g\_2D}$ can be obtained by changing the v_x in (2.38) to v_y. After replacing the local index of $f^n(\mathrm{i},\mathrm{j},\mathrm{i}_k,\mathrm{j}_k)$ with global index (2.30), the numerical trapezoidal integration (2.38) could be expressed by a matrix-vector product of $\{\mathrm{j}_g\}^n_{2D} = \mathbf{S}_{\mathrm{j}\_2D}\{f\}^n$. The $\{\mathrm{j}_g\}^n_{2D}$ here is a surface current density, which agrees with the fact that graphene is a 2-D material whose current flow is a sheet current flow. However, Maxwell's equations require a volume current density $\{\mathrm{j}_g\}^n$. Here, we can treat a graphene layer as a thin sheet [9] and obtain an equivalent volume current density $\{\mathrm{j}_g\}^n = \{\mathrm{j}_g\}^n_{2D}/dz$ [36], where dz is the grid size perpendicular to the graphene sheet. Thus, by using $\mathbf{S}_\mathrm{j} = \mathbf{S}_{\mathrm{j}\_2D}/dz$, we obtain

$$\{\mathrm{j}_g\}^n = \mathbf{S}_\mathrm{j}\{f\}^n. \tag{2.39}$$

The coupled systems of equations, including the Maxwell subsystem (2.27), the Boltzmann subsystem (2.33), and the coupling mechanism through conduction current density (2.39), constitute a nonlinear system of equations, as shown in the following

$$\begin{bmatrix} \mathbf{M}_A & 0 \\ 0 & \mathbf{B}^n \end{bmatrix} \begin{bmatrix} \{x\}^{n+1} \\ \{f\}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_B & \mathbf{M}_{\mathrm{j}} \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \{x\}^{n} \\ \{f\}^{n} \end{bmatrix} + \begin{bmatrix} \{x_0\}^{n+1} \\ \{\tilde{f}_0\} \end{bmatrix}, \tag{2.40}$$

where

$$\mathbf{M}_{\mathrm{j}} = \begin{bmatrix} -\mathbf{S}_{\mathrm{j}} \\ 0 \end{bmatrix} \text{ and } \{x_0\}^{n+1} = \begin{bmatrix} -\{\mathrm{j}_{\mathrm{i}}\}^{n+1} \\ 0 \end{bmatrix}.$$

Given an initial condition $\{x\}^0$ and $\{f\}^0$, and the excitation $\{x_0\}$, we can update the system in time based on (2.40), and finally obtain the full-wave response of General 3-D Cu-G hybrid nano-interconnects.

Next, we prove that the proposed time marching of the co-simulation system shown in (2.40) is unconditionally stable. Since the constant terms and excitation are irrelevant to stability, they are ignored in the following stability analysis. For the coupled nonlinear system of equations (2.40), at every time step, we have

$$\begin{bmatrix} \{x\}^{n+1} \\ \{f\}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_A^{-1}\mathbf{M}_B & \mathbf{M}_A^{-1}\mathbf{M}_{\mathrm{j}} \\ 0 & (\mathbf{B}^n)^{-1} \end{bmatrix} \begin{bmatrix} \{x\}^{n} \\ \{f\}^{n} \end{bmatrix} = \mathbf{G}^n \begin{bmatrix} \{x\}^{n} \\ \{f\}^{n} \end{bmatrix}. \tag{2.41}$$

As can be seen, the amplification matrix $\mathbf{G}^n$ is a block upper triangular matrix, whose eigenvalues $\{\lambda(\mathbf{G}^n)\}$ consist of the eigenvalues of the two diagonal block matrices $\mathbf{M}_A^{-1}\mathbf{M}_B$, and $(\mathbf{B}^n)^{-1}$, namely

$$\{\lambda(\mathbf{G}^n)\} = \{\lambda(\mathbf{M}_A^{-1}\mathbf{M}_B)\} \oplus \{\lambda((\mathbf{B}^n)^{-1})\}.$$

In other words, the overall stability of the coupled nonlinear system (2.40) is decoupled and determined by the stability of each subsystem (2.27) and (2.33). Because both $|\lambda(\mathbf{M}_A^{-1}\mathbf{M}_B)|$ and $|\lambda((\mathbf{B}^n)^{-1})|$ are bounded by 1, all the $|\lambda(\mathbf{G}^n)|$, thereby $\rho(\mathbf{G}^n)$ are bounded by 1, hence we prove the co-simulation algorithm (2.40) is unconditionally stable for an arbitrary choice of time step. Notice that, in this co-simulation scheme (2.40), neither of the two physical coupling flows determines the overall time marching stability. The first coupling flow from the Maxwell part, manifested by the electric field $\mathbf{E}$ in Boltzmann subsystem, enters system matrix $\mathbf{S}_k^n$ but cannot change its skew-symmetry, thus cannot determine the stability of the Boltzmann subsystem. The second coupling flow from the Boltzmann part, the conduction current density of graphene, becomes the off-diagonal block in (2.40), thus cannot determine the eigenvalues thereby the stability of the Maxwell subsystem.

The unconditional stability of the entire system allows both Maxwell and Boltzmann subsystems to use the same arbitrary time step, despite their different characteristic time constants. Hence, the time step can be chosen solely based on accuracy. For the Boltzmann subsystem, the sampling theorem sets an upper limit of the accurate time step. The characteristic time constant of Maxwell subsystem is usually determined by the main signal frequency ν_{sig}, which requires a $\Delta t \leq 1/(10\nu_{\text{sig}})$. Taking into account Boltzmann's time step requirement (2.36), an accurate time step for the entire coupled system would be

$$\Delta t \leq \min\{\tau/10, 1/(10\nu_{\text{sig}})\}. \tag{2.42}$$

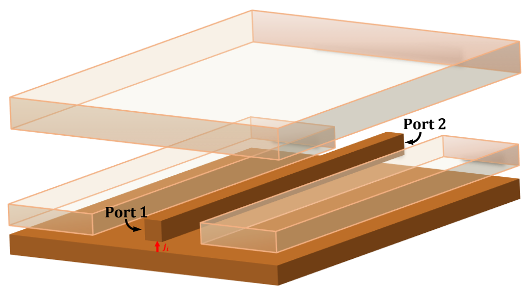
Since both Maxwell and Boltzmann subsystems use backward difference in time, the overall accuracy in time for each subsystem as well as for the whole coupled system is $\mathcal{O}(\Delta t)$, while the accuracy in space is of second-order.

2.5 Numerical Results from Proposed Solver

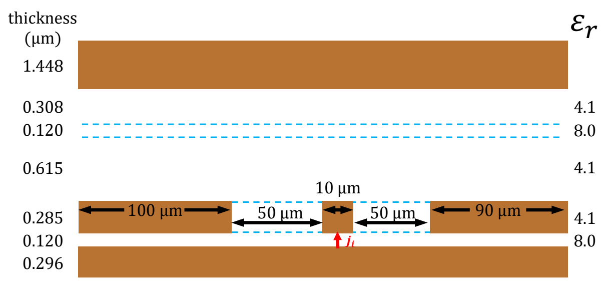
In this section, we first validate the accuracy of the proposed multiphysics solvers by comparing our numerical results with measurements. After validating both Maxwell and Boltzmann solvers, we proceed to simulate realistic graphene-encapsulated Cu nano-interconnects [3] and analyze their DC conductivity, crosstalk effect, and propagation delay.

2.5.1 Validation of the Maxwell Solver

We first validate the accuracy of the proposed work by simulating a realistic test-chip interconnect structure, which is fabricated using a silicon processing technology [37]. This 100 μm-long test-chip interconnect comprises 3 metal layers and 5 inhomogeneous dielectric

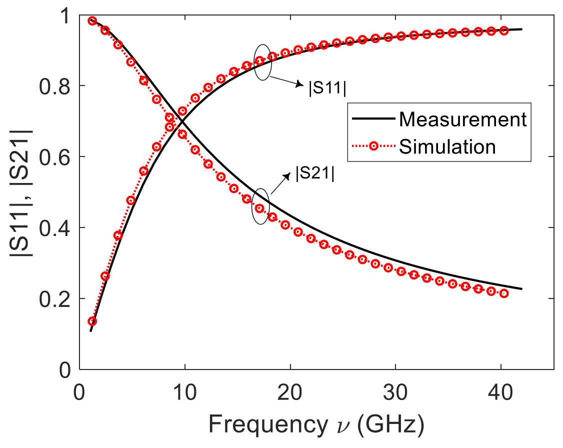


(a)

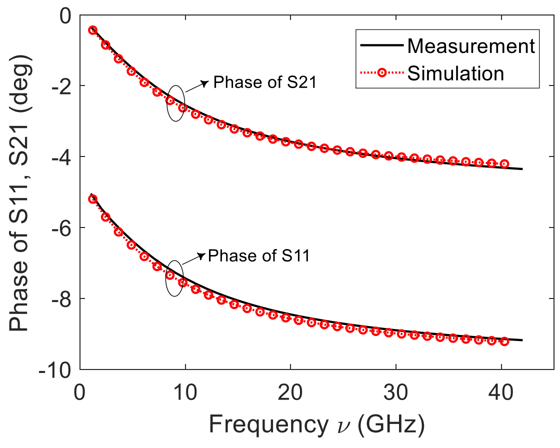


(b)

Figure 2.4. Geometry of a test-chip interconnect. (a) 3-D view of three metal layers, where the current source is supplied from bottom metal layer to the center wire at port 1. (b) Front view of the test-chip interconnect.



(a)



(b)

Figure 2.5. Simulated S-parameters of a test-chip interconnect in comparison with measurements. (a) Magnitude of S_{11} and S_{21}. (b) Phase of S_{11} and S_{21}.

stacks, whose cross-sectional view is illustrated in Fig. 2.4. Fig. 2.4 also shows all geometrical dimensions and the relative permittivity ϵ_r of each layer. A current source of a time derivative Gaussian pulse $\mathrm{j_i} = -(t-t_0)\exp[-(\frac{t-t_0}{\tau_s})^2]$ A/m^2 ($t_0 = 4\tau_s$, $\tau_s = 2\times 10^{-11}$ s) is placed right in the middle at the near-end of the center interconnect. The 100 μm-long interconnect is sandwiched between two 20 μm-long air layers in the front and at the back. The smallest mesh size used in the simulation is 0.04 μm, and the time step for time marching is 4×10^{-13} s owing to the proposed unconditionally stable method. After performing a Fast Fourier Transform (FFT) on the current source and the simulated time-domain port voltages, we directly obtain the Z-parameters of the structure, which are then converted to S-parameters with a 50 Ω reference impedance. The S-parameters of this test-chip interconnect are measured in the frequency range 45 MHz-40 GHz using an HP8510 system, where the undesirable signals from cables and probes are further removed following the short-open-load-thru (SOLT) technique, meanwhile the remaining noises generated by the bondpads, vias, and access lines are de-embedded using a YZ-matrix technique [37]. The simulated S-parameters and measured ones, as shown in Fig. 2.5, agree very well with each other.

2.5.2 Validation of the Boltzmann Solver

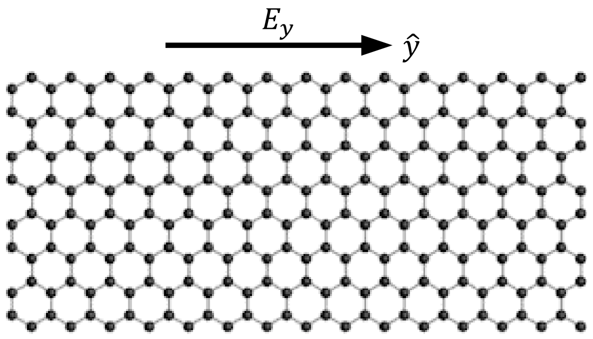


Figure 2.6. Structure of a single layer graphene ribbon with length L and width W. The electric field E_y is applied along **y** direction.

When numerically solving Boltzmann equation (2.18) for nm-scale structures such as a graphene ribbon in Fig. 2.6, we choose the backward difference method (2.29) because of its unconditional stability as proved in Section 2.4. The other common difference methods for discretizing a first-order time derivative equation (2.18) are either unstable (e.g. central difference method) or conditionally stable (e.g. forward difference method and Crank-Nicholson method). Fig. 2.7 (a) shows that the backward difference method (2.29) allows for the use of a large time step irrespective of the extremely small space step. In this example, initial f is Fermi-Dirac distribution (2.19), Fermi energy $\xi_{\mathrm{F}} = 0.21$ eV, relaxation time $\tau = 4 \times 10^{-11}$ s, the electric field $E_y = 2 \times 10^4$ V/m, $dx = 0.018$ μm, and $dy = 0.5$ μm. All three meth-

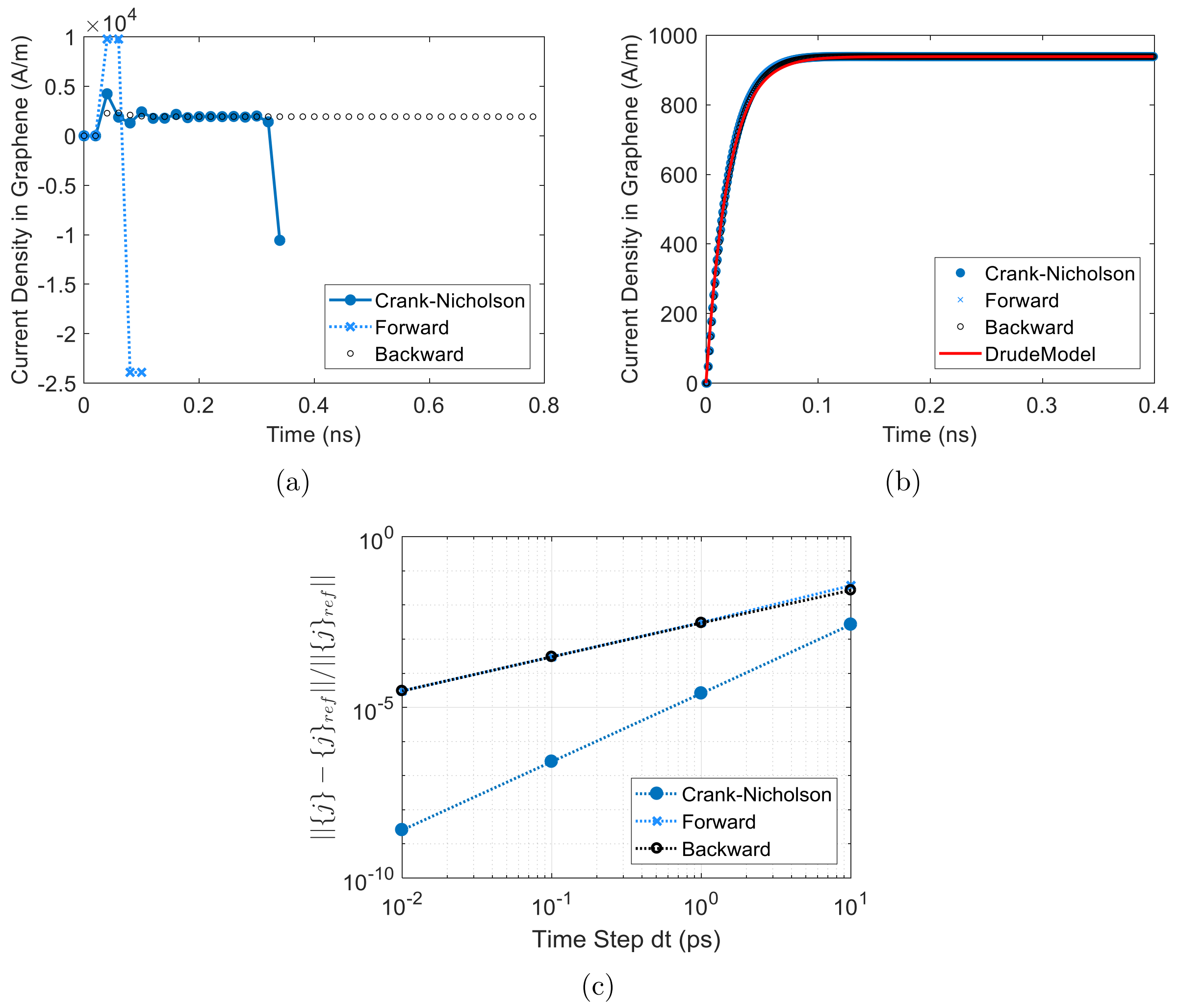


Figure 2.7. Surface current density in a graphene layer under a constant electric field E_y. (a) Extremely small grid and large electric field: $dx = 0.018$ μm, $dy = 0.5$ μm, and $E_y = 2 \times 10^4$ V/m. The $dt = 2 \times 10^{-11}$ s. (b) Larger grid and smaller electric field: $dx = 5.4$ μm, $dy = 5$ μm, and $E_y = 2 \times 10^3$ V/m. The $dt = 1 \times 10^{-12}$ s. (c) Error as a function of time step.

ods use the same $dt = 2 \times 10^{-11}$ s. After performing the time marching for a long time, only backward difference method remains stable. When approaching the steady state, the backward difference method also exhibits a much smaller un-physical oscillation caused by numerical error.

As for the accuracy, we find all three methods, backward-, forward-, and Crank-Nichoison methods, give similar results as long as the time step dt is accurately chosen based on the sampling theorem. Taking Boltzmann equation (2.18) as an example, the characteristic time length is the relaxation time τ, thus a time step $dt = \tau/20$ would be an accurate choice. Within the interval of such a time step, the time dependence of physical quantities does not go beyond linear, therefore backward-, forward-, and central-differences should produce the same result in terms of approximating the time derivative. To further investigate the convergence rate, a new example is specifically designed and illustrated in Fig. 2.7 (b). To make all three methods stable when using the same time step $dt = \tau/20$, we adopt a large grid $dx = 5.4$ μm and $dy = 5$ μm, and a smaller electric field $E_y = 2 \times 10^3$ V/m. The other parameters remain the same as in Fig. 2.7 (a). The current density solved from the backward difference method, as shown in Fig. 2.7 (b), agrees very well with those from other methods, including the Drude model. In Fig. 2.7 (c), we plot the error as a function of time step, where the error is assessed by $\|\{\mathrm{j}\} - \{\mathrm{j}_{ref}\}\|/\|\{\mathrm{j}_{ref}\}\|$, in which the Crank-Nicholson method with $dt = 1 \times 10^{-15}$ s is employed as the reference solution $\{\mathrm{j}_{ref}\}$, and norm-2 is used. The $\{\mathrm{j}\}$ is from either the backward or the forward difference, which includes j at all of the simulated time instants. As can be seen, the backward difference can produce accurate results, and its convergence rate is of first order as theoretically expected.

Another feasible validation is the surface DC conductivity $\sigma_{\mathrm{dc\_2d}}$ of a graphene sheet. Although the model developed in this work aims at the high-frequency and non-linear responses of graphene, the solver can also accurately reproduce the measured $\sigma_{\mathrm{dc\_2d}}$. One measurement, using well-known four-point measurements by injecting an excitation current through graphene ribbon and measuring the voltage drop, reports a $\sigma_{\mathrm{dc\_2d}} = 0.015$ S [15] for a graphene sheet of Fermi energy $\xi_{\mathrm{F}} = 0.21$ eV, mean free path $l = 600$ nm therefore relaxation time $\tau = l/v_{\mathrm{F}} = 6 \times 10^{-13}$ s. After substituting these parameters into the simulations as in Fig. 2.7, and dividing the steady-state surface current density j_y by the constant electric field E_y, the proposed Boltzmann solver gives a simulated $\sigma_{\mathrm{dc\_2d}} = 0.0147$ S, which agrees very well with the measurements since the relative error is only 2.0%.

2.5.3 Enhanced Electrical Conduction in Cu-G Nanowires Predicted by the Coupled Solver for Multiphysics Simulation

With both the Maxwell solver and the Boltzmann solver validated, next, we employ the proposed coupled Maxwell-Boltzmann solver to simulate a Cu nanowire encapsulated by a

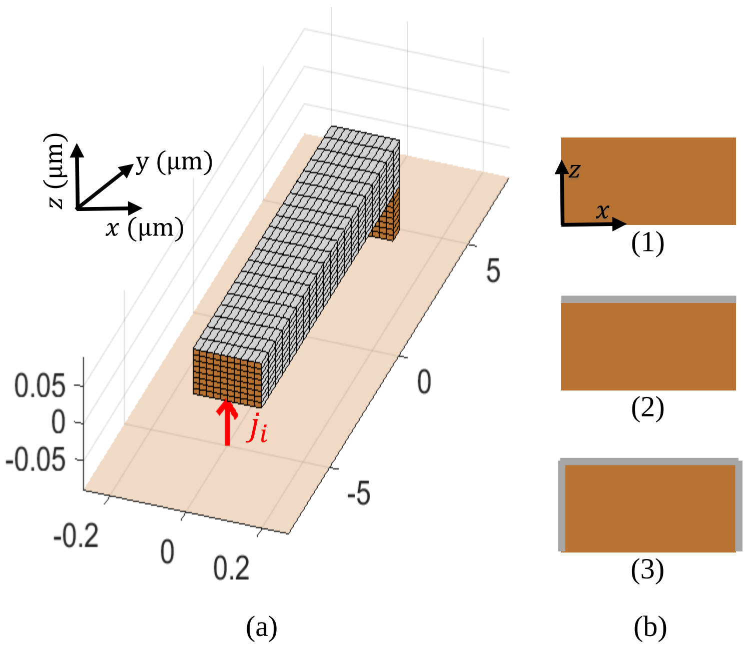


Figure 2.8. (a) Geometry and discretization of a Cu-G nanowire whose far-end is shorted to the ground PEC. The graphene layers, in gray color, are coated on top, left, and right surfaces. (b) Front view of the near-end. (1) is bare Cu without graphene coating, (2) has a single graphene layer coated on the top surface, (3) corresponds to the structure in (a).

single graphene layer on the top, left and right sides as illustrated in Fig. 2.8. The size of the Cu stripline, $W = 180$ nm, $H = 60$ nm, and $L = 10$ μm, is similar to that of a measured Cu-G nanowire [3], whose conductance is measured with standard four-point techniques. We use a uniform regular grid to discretize the Cu into $10 \times 8 \times 20$ grid cells. Graphene layers have a relaxation time $\tau = 2 \times 10^{-11}$ s and a Fermi energy $\xi_{\mathrm{F}} = 0.21$ eV, based on which we truncate the effective **k**-space into a energy range from 0 to $2\xi_{\mathrm{F}}$. Then, we discretize the truncated 2-D **k**-space with 10×20 grid cells. The Maxwell computation domain is a box with perfect electric conductor (PEC) boundaries at the top and the bottom, and perfect magnetic conductor (PMC) boundaries at the other four sides. To see the full-wave response of such Cu-G nanowires, we inject into the structure a current source whose waveform is a Gaussian derivative in time, $\mathrm{j_i} = -10^{16}(t - t_0)\exp[-(\frac{t-t_0}{\tau_s})^2]$ A/m^2, where $t_0 = 4\tau_s$ and $\tau_s = 2 \times 10^{-9}$s, indicating a maximal signal frequency of approximately 0.5 GHz. For the aforementioned real space grid, conventional FDTD requires a small Δt therefore about 10^8 time steps to

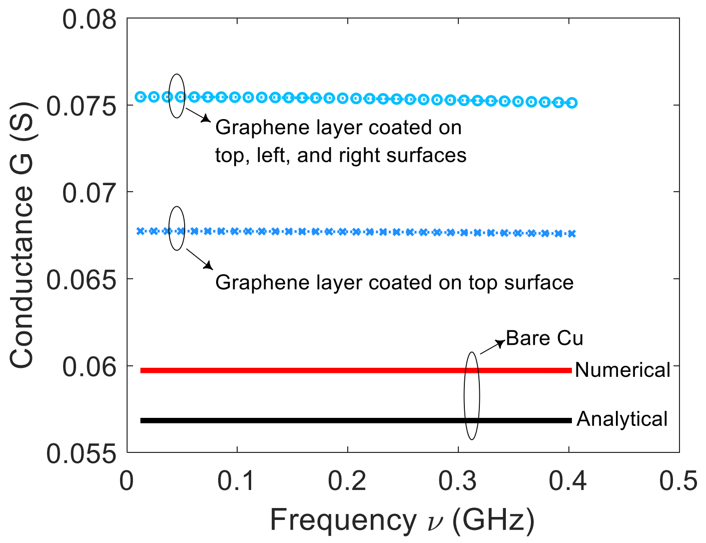


Figure 2.9. Simulated conductance G of the three interconnect structures in Fig. 2.8

finish the simulation in the width of a full pulse. However, in our unconditionally stable algorithm (2.40), only 200 time steps are simulated, where time step is solely determined by the accuracy requirement.

We perform a Fourier transform of the time domain data and calculate the admittance $Y(\omega) = I_{input}(\omega)/V_{drop}(\omega)$, whose real part, the conductance G, is plotted in Fig. 2.9. The numerical and analytical conductance G of the bare Cu case, plotted in solid lines in Fig. 2.9, shows a fairly good correlation with an error of 5.07%. Compared with the numerical G in bare Cu structure, a single graphene layer's coating on the top surface enhances the conductance G by 13.4%, whereas the coating on three sides enhances G by 26.4%. For the structure in Fig. 2.8(a), measurement [3] reports a 22% enhancement on G, which is very close to the simulated 26.4% here.

2.5.4 Increased Crosstalk Effect and Decreased Propagation Delay in Graphene-Encapsulated Cu Nano-Interconnects Predicted by the Proposed Multiphysics Solver

Next, to study the effect of coating graphene layers on the crosstalk, especially for cutting-edge 10 nm technology node, we analyze two parallel Cu-G nano-interconnect wires, whose geometry and discretization are illustrated in Fig. 2.10. We adopt similar settings as the

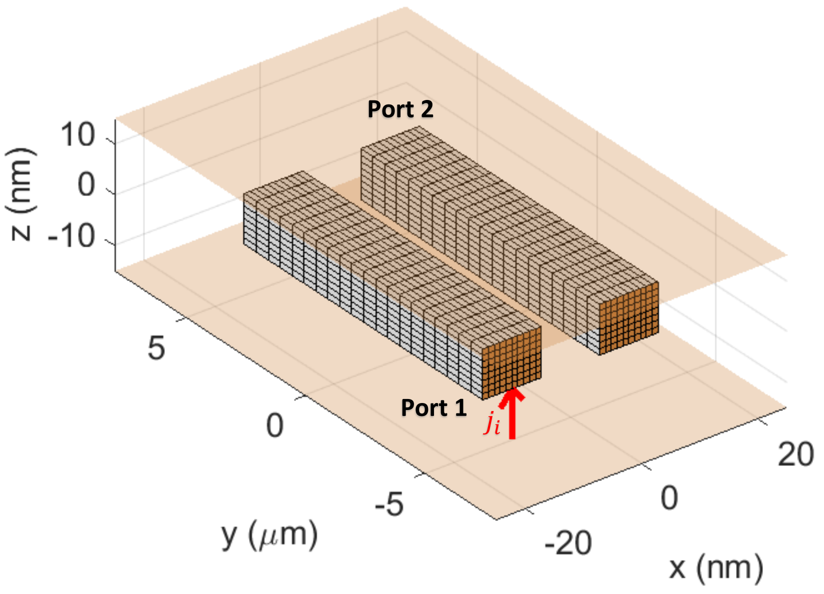
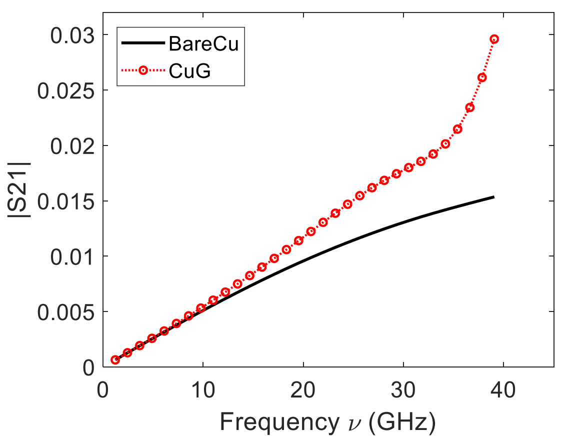
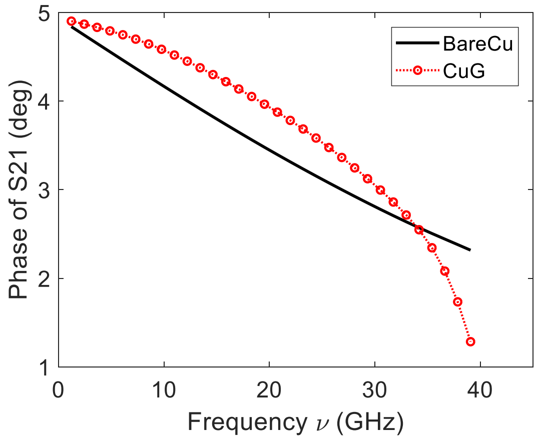


Figure 2.10. Geometry and discretization of two parallel Cu-G nano-interconnects. The cross section of each nano-interconnect is 10 nm × 10 nm, much smaller than that in Fig. 2.8. Port voltages on port 1 and port 2 are detected for analyzing the crosstalk S_{21}.

one in Fig. 2.8. Each Cu interconnect, whose $W = 10$ nm, $H = 10$ nm, and $L = 10$ μm, is discretized into a uniform $10 \times 8 \times 20$ grid. In this example, we inject a current source at port 1 and port 2 in turn, whose waveform is $\mathrm{j_i} = -(t - t_0)\exp[-(\frac{t-t_0}{\tau_s})^2]$ A/m^2, where $t_0 = 4\tau_s$ and $\tau_s = 2 \times 10^{-11}$s. The pulse has a maximal signal frequency of approximately 50 GHz. Due to the small spatial feature, conventional conditionally stable methods require about 10^7 time steps to finish the simulation of Maxwell subsystem, and 10^9 time steps to simulate the Boltzmann subsystem in the window of a full pulse [38]. However, using the proposed unconditionally stable algorithm, only 200 time steps are required, where time step is solely determined by accuracy. Furthermore, the same time step is used for simulating both Maxwell and Boltzmann subsystems. We then do an FFT on the simulated time-domain responses, from which we extract the crosstalk $|S_{21}|$ between the two ports. As can be seen from Fig. 2.11, the graphene coating clearly increases the crosstalk effect as compared to Cu-based counterparts.



(a)



(b)

Figure 2.11. Crosstalk S_{21} of the Cu-G nano-interconnects in Fig. 2.10. (a) Magnitude of S_{21}. (b) Phase of S_{21}.

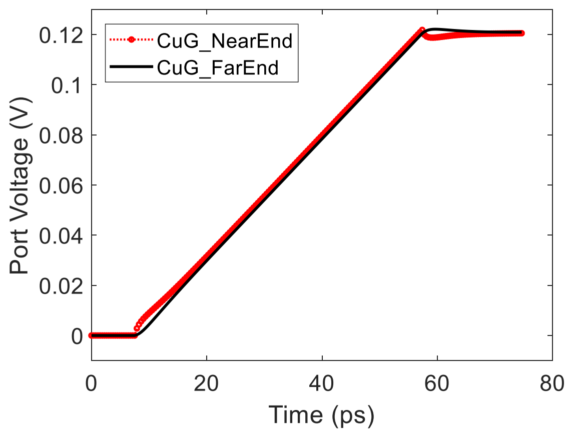
Table 2.1. Propagation Delay vs. Length L

	Bare Cu	Cu-Graphene
$L = 20\ \mu$m	9.0011 ps	2.2760 ps
$L = 10\ \mu$m	2.2513 ps	0.5864 ps
$L = 5\ \mu$m	0.5629 ps	0.1495 ps

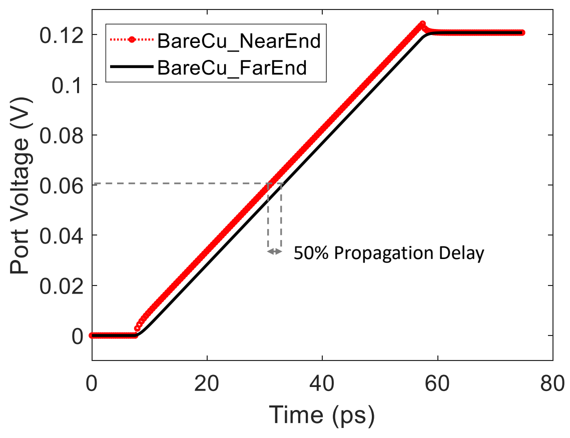
For analyzing the propagation delay in the Cu-G nano-interconnects, we use the same structure as in Fig. 2.10. This time, we inject a current source of

$$\mathrm{j_i}(t) = \begin{cases} 1.09 \times 10^{10}\ \mathrm{A/m^2} & 7.5\ \mathrm{ps} < t < 57.5\ \mathrm{ps} \\ 0 & \text{otherwise} \end{cases}.$$

The resulting port voltages, given in Fig. 2.12, have a ramp waveform of 50 ps transient time and 0.12 V maximum voltage, which is compatible with current Complementary Metal–Oxide–Semiconductor (CMOS) technology. The 50% propagation delay between the near-end and far-end of a single nanowire, and their dependence on the length L of nanowires, are listed in Table 2.1. For the 10 nm $\times$ 10 nm-thick graphene-encapsulated Cu nano-interconnects, from length $L = 5\ \mu$m to $L = 20\ \mu$m, the propagation delay is only 26% of the bare Cu



(a)



(b)

Figure 2.12. Near end and far end port voltages of a single nano-interconnect. (a) Single Graphene-encapsulated Cu nano-interconnect in Fig. 2.10. (b) Bare Cu counterparts of (a) without graphene coating.

counterparts. The result shows that Cu-G nano-interconnects have a faster data-transferring speed than that of bare Cu interconnects.

2.6 Comparisons between Proposed Multiphysics Solver and Drude Model Based Simulation

The key difference between the first-principles based simulation and the Drude-model based one is their way to deal with the conduction current in graphene. The first-principles model utilizes the dynamic time-domain response by directly solving Boltzmann equation (2.1), while the Drude model simplifies the Boltzmann transport theories to a steady-state conductivity model. The three major assumptions made in the Drude model may no longer be valid in simulating realistic ultra-scaled Cu-G hybrid nano-interconnects in a high-frequency setting, as revealed by the numerical examples shown in this Section. Through extensive numerical experiments, we find that the spatial size and the signal frequency can determine the difference between the two simulations. First, a spatial size smaller than 100 nm can significantly increase the spatial variance of σ_{dc} extracted from a direct Boltzmann solver, thus can decrease the reliability of ignoring the spatial variation term $\mathbf{v} \cdot \nabla_{\mathbf{r}} f$ in Boltzmann equation. Second, a high signal frequency, which is comparable to the back-scattering frequency of graphene, can lead to a non-linear response, thus making the linear response based Drude model less accurate. The effect of simplifying f in $\mathbf{k}$-space is hard to distinguish in the comparison made here, because part of the effect is already included in σ_{dc}, and σ_{dc} in Drude model is provided from our direct Boltzmann solver.

In this section, we simulate a suite of examples to make a comparison between the first-principles model based simulation and the Drude model based one in analyzing Cu-G hybrid nano-interconnects. Numerical results indicate that two factors, spatial size and signal frequency, can determine the difference between the two simulations.

2.6.1 Validation of Both Simulations at DC

We re-simulate the example in Fig. 2.6 that is a graphene ribbon subject to a constant electric field. In this case, the time domain current density, from equilibrium state to steady state, has an analytical expression using the Drude model, as derived as follows. From the inverse Fourier transform, the time-domain counterpart of Drude model $\tilde{\sigma}_g(\omega) = \sigma_{dc}/(1+\mathrm{j}\omega\tau)$ can be found as

$$\sigma_g(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty}\frac{\sigma_{dc}}{1+\mathrm{j}\omega\tau}\mathrm{e}^{\mathrm{j}\omega t}d\omega = \frac{\sigma_{dc}}{\tau}\mathrm{e}^{-t/\tau}. \tag{2.43}$$

Given a constant electric field $\mathbf{E}$, the time domain current density in graphene layer is

$$\begin{aligned}\mathbf{j}_g(t) &= \sigma_g(t) * \mathbf{E}(t) = \int_{-\infty}^{+\infty}\sigma_g(t-t)\mathbf{E}(t)dt \\ &= \int_0^t \frac{\sigma_{dc}}{\tau}\mathrm{e}^{-(t-t)/\tau}\mathbf{E}(t)dt = \sigma_{dc}\mathbf{E}(1-\mathrm{e}^{-t/\tau}),\end{aligned} \tag{2.44}$$

where $*$ denotes convolution. The above can be used as a benchmark to validate both the numerical Drude model based simulator and the first-principles based simulator developed in this work.

In this example, initial f is Fermi-Dirac distribution (2.19), Fermi energy $\xi_{\mathrm{F}} = 0.21$ eV, relaxation time $\tau = 4\times10^{-11}$ s, the electric field $E_y = 2\times10^3$ V/m, $dx = 5.4$ μm, and $dy = 5$ μm. Both simulators use the same $dt = 1\times10^{-12}$ s. After performing the time marching, it is found that the current densities obtained from both simulators, as shown in Fig. 2.13, agree very well with each other, and also with the analytical data.

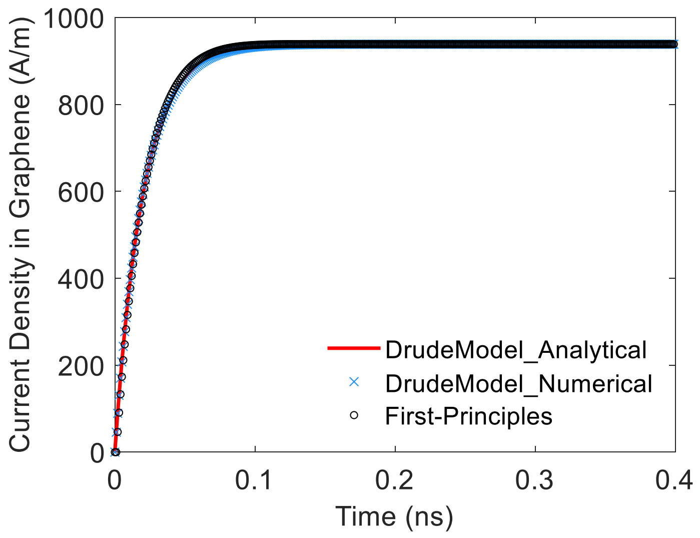


Figure 2.13. Surface current density in a graphene layer subject to a constant electric field E_y.

2.6.2 Simulating Graphene-Encapsulated Cu Nano-Interconnects and Two Determining Factors: Feature Size & Signal Frequency

A Cu-Graphene nano-interconnect encapsulated by a single graphene layer on the top, left and right sides [3] is simulated, whose geometry and discretization are illustrated in Fig. 2.14. The Cu interconnect, with $W = 10$ nm, $H = 10$ nm, and $L = 10$ μm, is discretized into a uniform $10 \times 8 \times 20$ grid. Graphene layers have a relaxation time $\tau = 2 \times 10^{-11}$ s and a Fermi energy $\xi_\mathrm{F} = 0.21$ eV, based on which we truncate the effective **k**-space into an energy range from 0 to $2\xi_\mathrm{F}$. Then, we discretize the truncated 2-D **k**-space with 10×20 grid cells. The extracted surface conductivity of graphene is $\sigma_{\mathrm{dc\_2d}} = 0.2286$ S. The Maxwell computation domain is a box with PEC boundaries at the top and the bottom, and PMC boundaries at the other four sides. For the aforementioned real space grid, due to the small spatial feature, conventional conditionally stable methods require about 10^7 time steps to finish the simulation of the Maxwell subsystem, and 10^9 time steps to simulate the Boltzmann subsystem in the window of a full pulse [38]. However, using the proposed unconditionally stable algorithm, only 200 time steps are required, where time step is solely determined by

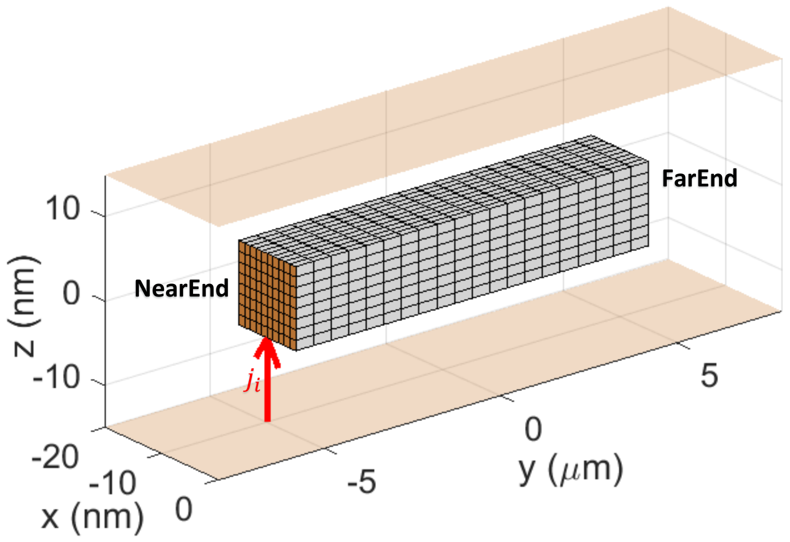


Figure 2.14. Geometry and discretization of a Cu-G nano-interconnect. The cross section of the nano-interconnect is 10 nm × 10 nm. Port voltages at the near and far end are sampled for analysis.

accuracy as given by (2.42). Furthermore, the same time step is used for simulating both Maxwell and Boltzmann subsystems.

Determining Factor 1 - Feature Size

Table 2.2. Propagation Delay Predicted by the First-Principles Based Simulation and the Drude Model Based One

Feature Size	Bare Cu	Cu-G, Drude	Cu-G, Proposed
10 nm ×10 nm ×10 μm	2.2513 ps	1.5830 ps	0.5864 ps
100 nm×100 nm×100 μm	2.2513 ps	2.1201 ps	1.8476 ps

We first analyze the propagation delay in the Cu-G nano-interconnect by injecting a current source of

$$\mathrm{j_i}(t) = \begin{cases} 1.09 \times 10^{10}\ \mathrm{A/m^2} & 7.5\ \mathrm{ps} < t < 57.5\ \mathrm{ps} \\ 0 & \text{otherwise} \end{cases} .$$

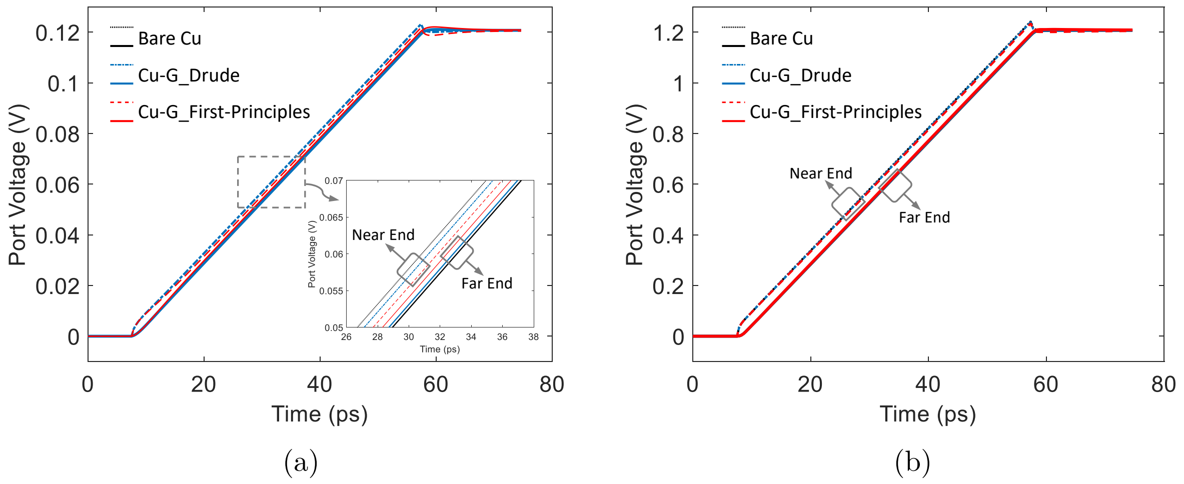


Figure 2.15. Near- and far-end port voltages of a Cu-G nano-interconnect. The red color is from the first-principles based multi-physics simulation, the blue color is from the Drude model based simulation, and the black color is from the simulation of Bare Cu counterparts without graphene coating. All solid lines represent far-end voltages while dashed and dotted lines represent near-end voltages. (a) Simulation of the Cu-G nano-interconnect in Fig. 2.14. (b) Simulation of a 10 times larger version of the Cu-G interconnect in Fig. 2.14.

The resulting port voltages, given in Fig. 2.15, has a rising time of 50 ps. To show the effect of spatial size, a small example of 10 nm×10 nm×10 μm and a larger example of 100 nm×100 nm×100 μm are simulated. The small example has a structure shown in Fig. 2.14. Its port voltages are plotted in Fig. 2.15 (a). For the larger example in Fig. 2.15 (b), only the spatial size is enlarged by 10 times and all the other settings are the same as in the small example. The simulated propagation delays from two simulations are listed in Table 2.2. For the small example, the Drude model gives a propagation delay three times larger than the first-principles based multiphysics simulation. For the larger example, the difference between two simulations becomes much smaller. More experiments show that 100 nm is a good separation criterion, above which two simulations give almost the same results, whereas below 100 nm two simulations can be very different. Direct observations on the extracted σ_{dc} of the graphene plane show a large spatial variation when the spatial size is smaller than 100 nm. However, when the spatial size is larger than 100 nm, the extracted σ_{dc} is identical everywhere in the graphene plane. As analyzed above, a major difference between two simulations is the Drude model's ignoring the spatial variation term $\mathbf{v} \cdot \nabla_{\mathbf{r}} f$ in Boltzmann equation (2.18).

Determining Factor 2 - Signal Frequency

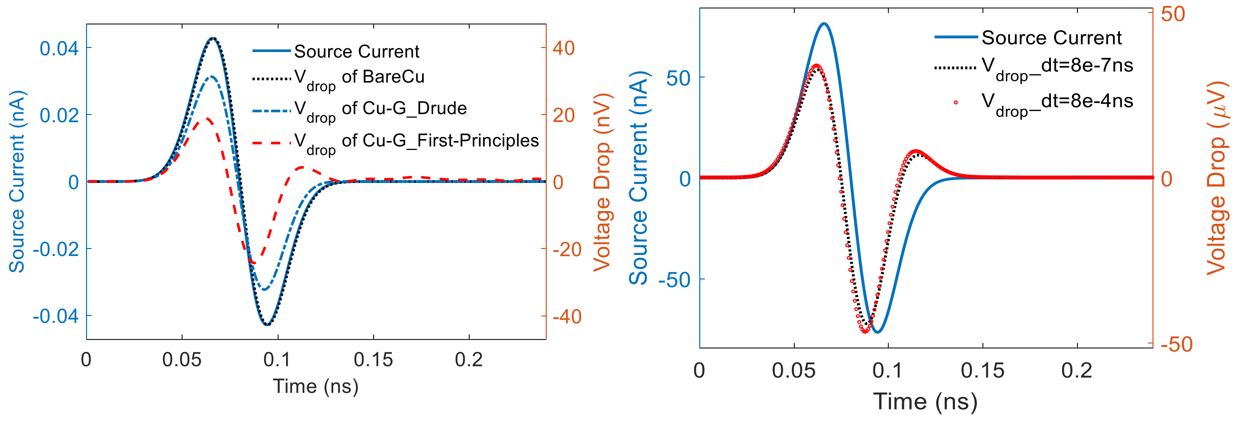


Figure 2.16. Time-domain voltage drop (labeled to the right) along the single Graphene-encapsulated Cu nano-interconnect in Fig. 2.14. The Gaussian derivative source current is plotted in solid line and labeled to the left. (a) Comparison between different models. (b) Comparison between different choice of time step in the proposed first-principles modeling.

To show the effect of a high signal frequency, a Gaussian derivative source current density is injected from the near end of the structure in Fig. 2.14. The Gaussian derivative pulse has a relatively narrow frequency band, which helps the analysis of the difference between the two simulations. For the injected source current, it has a waveform of $\mathrm{j_i} = -(t - t_0)\exp[-(\frac{t-t_0}{\tau_s})^2] \times 10^{16}$ A/m^2, where $t_0 = 4\tau_s$ and $\tau_s = 2 \times 10^{-11}$s, the maximal signal frequency of which is approximately 50 GHz. Here, the relaxation time of graphene is still $\tau = 2 \times 10^{-11}$s, yielding a back scattering frequency of 50 GHz and a surface DC conductivity $\sigma_{\mathrm{dc\_2d}} = 0.2286$ S. The source current and the voltage drop along the single graphene-encapsulated Cu nano-interconnect are plotted in Fig. 2.16a.

As can be seen from the waveform of voltage drops in Fig. 2.16a, Drude model based simulation predicts a Gaussian derivative voltage drop that is the same as the waveform of the source current, therefore indicates a pure resistor-like performance. However, the first-principles model shows an oscillating voltage drop, therefore predicts a full-wave effect for the graphene-encapsulated Cu nano-interconnect. As for the low frequency comparison, one example at DC is already shown in Fig. 2.7, where the Drude model agrees very well with the first-principles solver.

The reason to the difference caused by a high signal frequency is much more complicated than that caused by the spatial factor. First, Drude model assumes a linear coupling between the Maxwell and Boltzmann subsystems by using Ohm's Law (2.11). However, this is unlikely to be the case. For example, the nonlinear operation (2.20), an integration of f over $\mathbf{k}$-space to obtain the current, can directly cause the non-linear coupling between the Maxwell and Boltzmann subsystems. Second, Drdue model assumes a linear response in Boltzmann equation by replacing $\partial/\partial t$ with $\mathrm{j}\omega$. The Boltzmann equation has its own characteristic frequency ω_{BE} determined by the relaxation time τ, which tells how f attenuates to its steady state. When the signal frequency ω_{sig}, namely the characteristic frequency of Maxwell subsystem and the electric field, is comparable to the ω_{BE}, the electric field changes so fast that the f can hardly attenuate to its steady state. As a result, the response of f, governed by Boltzmann equation (2.18), is not linear. At a high signal frequency, both of the two reasons given in the above can be important, making the Drude model less accurate. However, at a low frequency, the two factors are less important as the electric field changes very slowly. Thus, the Boltzmann equation is subject to an almost constant electric field, giving a linear response and linear coupling. That's why we see a good match between two simulations at a low frequency. The same analysis can be applied to many other steady-state models of graphene.

We also simulated this example using the time step permitted by a conditionally stable scheme, and investigated whether our unconditionally stable method is able to produce the same accurate result while using an orders-of-magnitude larger time step. As can be seen from Fig. 2.16b, they do agree well with each other. In this figure, the conventional method uses a time step of 8×10^{-7} ns, while the proposed one uses 8×10^{-4} ns.

It has been shown that the relaxation time of graphene predicted by theory can vary in a wide range between 0.1 ps and 100 ps. However, as a modeling and simulation method,

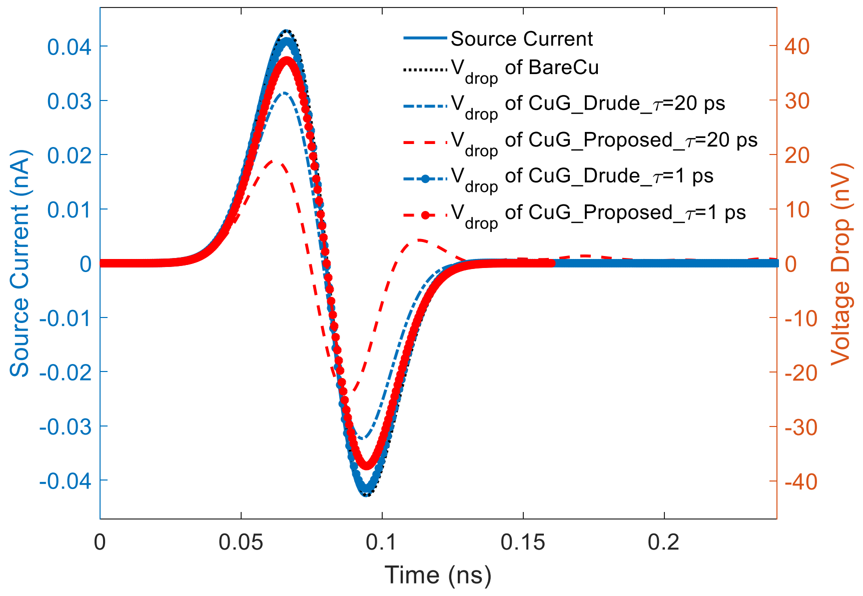


Figure 2.17. Time-domain voltage drop (labeled to the right) along the single Graphene-encapsulated Cu nano-interconnect in Fig. 2.10. The Gaussian derivative source current is plotted in solid line and labeled to the left.

the proposed work has no restriction on the choice of material parameters. To demonstrate this point, we also simulated the same example using $\tau = 1$ ps, which is the relaxation time observed in many experiments. In Fig. 2.17, we show voltage drops predicted by the proposed multiphysics solver and the Drude model based simulation in graphene-encapsulated Cu nano-interconnects. The structure and parameter settings are the same as those in Fig. 2.10. The injected source current has a waveform of $\mathrm{j_i} = -(t-t_0)\exp[-(\frac{t-t_0}{\tau_s})^2] \times 10^{16}$ A/m^2, where $t_0 = 4\tau_s$ and $\tau_s = 20$ ps, the maximal signal frequency of which is approximately 50 GHz. As can be seen from Fig. 2.17, when the relaxation time of graphene $\tau = 20$ ps, the two are very different, and the proposed solver captures physics that cannot be captured in Drude model based simulation. When $\tau = 1$ ps, there still exists a noticeable difference between a simplified model based analysis and the proposed multiphysics analysis, although the difference is smaller.

2.7 Conclusion

In this work, we propose a multiphysics model for general 3-D Cu-G hybrid nano-interconnects via co-simulating in time domain Maxwell's equations, Boltzmann equation under relaxation time approximation, and the linear dispersion of graphene. We also develop an unconditionally stable simulation algorithm for the proposed multiphysics model, allowing for the use of an arbitrarily large time step irrespective of the extremely small space step for simulating nano-interconnects. Numerical experiments and their comparisons with measurements validate the accuracy and efficiency of the proposed multiphysics modeling algorithm. From the simulated full-wave response in time domain, many essential parameters of Cu-G nano-interconnects, including electrical conductivity, crosstalk effect, and propagation delay, can be easily evaluated.

To compare with proposed multiphysics simulation algorithm, another algorithm using a simplified Drude model together with the FDTD is also developed. The differences between two algorithms are theoretically analyzed by examining the assumptions and simplifications made in the Drude model. Moreover, we also study the differences via numerical experiments performed on the graphene-encapsulated Cu nano-interconnects. From our analysis, an on-chip Cu-graphene hybrid system, featuring a high operating frequency and a sub - 10 nm dimension, requires a more accurate first-principles based modeling and simulation algorithm so that the dynamic coupling between graphene and electromagnetic fields can be accurately captured. In the future, more physics effects, such as the intraband transition in graphene and the surface scattering at the Cu-graphene interface, can be included to further enrich the multiphysics model and enhance the prediction power.

3. A NON-OVERLAPPING DOMAIN DECOMPOSITION PARALLEL ITERATION SCHEME OF NONUNIFORM FINITE DIFFERENCE METHOD FOR LARGE-SCALE ON-CHIP SIMULATION

3.1 Introduction

High-frequency integrated circuit (IC) design imposes many modeling challenges to electromagnetic analysis, including conductor loss, large numbers of dielectric stacks, strong non-uniformity, the presence of substrate, large numbers of conductors, large aspect ratio, broadband, and 3-D complexity. Almost every challenge increases the number of unknowns, and hence the problem size one needs to solve when tackling an IC problem. To date, the fastest partial-differential-equation (PDE) based solvers scale as $\mathcal{O}(N)$ in both memory complexity and computational complexity. This performance is generally regarded as the limit that one can achieve in computational electromagnetics (CEM). However, since the number of unknowns N is big in IC analysis even for a circuitry of a modest size, the current performance of CEM techniques is still insufficient when tackling a realistic IC design problem that can involve billions of unknowns.

To further increase the simulation capacity, a parallel domain decomposition method (DDM) can be deployed utilizing distributed computational resources. The idea of DDM, since it was first proposed to solve PDEs by Schwarz in 1869, has evolved into an important computational approach in numerical simulation [17]. Essentially, the DDM is a divide-and-conquer algorithm. To solve a PDE, the DDM decomposes the whole computational domain into smaller overlapping or non-overlapping subdomains as shown in Fig. 3.1, solves each subdomain individually, then recombines local solutions to provide a global solution. The DDM is primarily used 1) to simplify mesh generation for complicated geometries, such as performing a local mesh refinement and following moving components, and 2) as an indispensable parallel scheme especially for multiscale and multiphysics problems.

Looking back the history, the DDM-based parallelization has been widely used in PDE-based CEM solvers. The conventional Yee's finite-difference time-domain (FDTD) method has a simple iterative scheme and naturally high parallelism, therefore, can be parallelized with DDM for various applications [39], [40]. However, the time step of Yee's FDTD is limited by the Courant–Friedrichs–Lewy (CFL) stability condition, which requires way too many time steps to finish an on-chip simulation. To eliminate the restriction on time step, several implicit temporal difference schemes have been developed in recent years, such as alternating direction implicit FDTD (ADI-FDTD) method [41], locally 1-D FDTD (LOD-FDTD) method [42], Laguerre-FDTD method [43], and Crank–Nicolson FDTD (CN-FDTD) method [44]. Both the ADI-FDTD method and LOD-FDTD method involve the solution of tri-diagonal systems along the three directions of Cartesian coordinate, and their parallel implementations with DDM have been introduced in [45] and [18]. Despite higher computational efficiency, the ADI-FDTD method and LOD-FDTD method suffer from worse accuracy as the time step increases [46], [47]. The Laguerre-FDTD method can be parallelized with conformal DDM [48] and nonconformal DDM [19]. The CN-FDTD method has a second order accuracy in both time and space, and has been parallelized with DDM in [49]. At every time step, both the Laguerre-FDTD method and CN-FDTD method need to solve a large sparse matrix equation. Corresponding parallel DDMs [19], [48], [49] solve this large matrix with block Gaussian elimination, where Schur complements can be computed in parallel. DDMs have also been developed in the finite-element method (FEM) [50]–[53].

The parallelization in DDMs is achieved mainly in two approaches. The first approach is the Schur complement and block Gaussian elimination, which preserves the exact coupling among subdomains in a mathematically rigorous manner. In this approach, the global system matrix is rearranged to separate 1) interior-interior coupling blocks, 2) interior-interface coupling blocks, and 3) interface-interface coupling blocks. Then, Schur complements relative to interface unknowns can be computed in parallel. After a block Gaussian elimination, the original computational system can be reduced to a much smaller size only containing interface unknowns, therefore can be solved more easily. However, when many subdomains and higher levels of eliminations are needed for on-chip simulation, second and higher levels of eliminations require Schur complements on a dense matrix [51], which can be very expensive when the interface is large. The second approach is to use transmission conditions to break the coupling among subdomains meanwhile maintaining the continuity of solution and flux (normal derivative of solution). As a result, subdomains can be solved in parallel. The transmission conditions are boundary conditions on the interfaces that exchange solutions between subdomains. Dirichlet conditions, Neumann conditions, and Robin conditions are commonly employed transmission conditions [17]. However, many transmission conditions may not be enforced strictly but approximately, e.g., via minimization of the residual between the different subdomains [54]. Also, the convergence of this approach depends on the equation parameters, the interface conditions, the overlap size, and the shapes and sizes of the subdomains, therefore can be difficult to predict and control in practice. As a consequence, the second approach used as a solver can suffer from slow or even a lack of convergence [55].

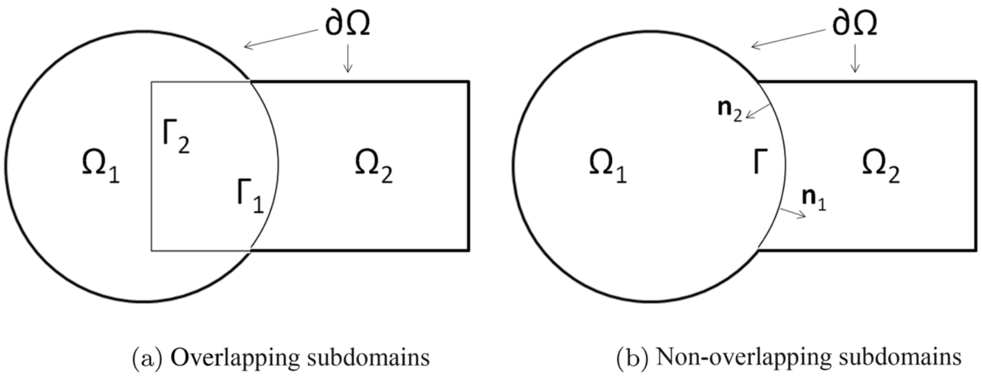


Figure 3.1. Space domain partition of two subdomains Ω_1 and Ω_2. Two subdomains can overlap as in (a) or connect with each other only at their interfaces as in (b).

In this work, we propose a new non-overlapping parallel DDM for finite difference method (FDM). Different from conventional DDMs where only the system matrix is partitioned to decouple subdomains, our proposed DDM partitions both the system matrix and the interface field according to contributions from each subdomain. For example, in a two-subdomain system in Fig. 3.1 (b), two subdomains Ω_1 and Ω_2 overlap only at the interface Γ. Let corresponding solutions from each subdomain at the interface be $x_{1,\Gamma}$ and $x_{2,\Gamma}$, and the global solution at the interface be x_Γ. Then, conventional DDMs use

$$x_\Gamma = x_{1,\Gamma} = x_{2,\Gamma}$$

to guarantee the continuity of solution. However, our proposed DDM uses

$$x_\Gamma = x_{1,\Gamma} + x_{2,\Gamma},$$

where the physical meaning of $x_{\mathrm{i},\Gamma}$ here is the contribution from i-th region. In this way, our proposed DDM can preserve the coupling among subdomains while avoiding a Schur complement and block Gaussian elimination. Also, no approximated transmission condition is needed at the interface. The key idea of our proposed DDM is to disassemble the numerical finite difference system and extract each subdomain's contribution to both the system matrix and the interface field. This can be done readily using a path-based single-grid formulation for FDM [29], from which, we can clearly decouple the physical interactions among subdomains and extract a clean subsystem to solve independently. Note that our proposed DDM is also different from pure domain decomposition preconditioner where only a frontal matrix is generated according to DDM. In this work, to reduce the overall unknown number, a nonuniform FDM is used so that fine discretizations are employed only when necessary. Our proposed DDM can be applied to FDM in both frequency domain and time domain. In time domain, to allow for a large time step for on-chip simulation, we use an unconditionally stable backward-difference based FDTD [20]. Extensive numerical examples are simulated to demonstrate the capability of the proposed parallel solver in both frequency domain and time domain.

3.2 Theory of the Parallel Solver

3.2.1 Review of Patch-Based Single-Grid FDTD Formulation

First, we provide a brief review of the patch-based single-grid FDTD formulation, which is developed in [29]. It is used in this work to facilitate the development of a non-overlapping domain decomposition parallel iteration scheme, as this formulation reveals clearly how the submatrices in different regions are assembled in an FDTD to build a global system matrix.

The formulation is valid for both 2- and 3-D grids. Let $\{\mathrm{e}\}$ be a global electric field unknown vector of length N_{e}, and $\{h\}$ being a global magnetic field unknown vector of length N_h. The FDTD can be written into the following form:

$$\mathbf{S}_{\mathrm{e}}\{\mathrm{e}\} = -\mathbf{D}_{\mu}\{\dot{h}\}, \tag{3.1}$$

$$\mathbf{S}_{h}\{h\} = \mathbf{D}_{\epsilon}\{\dot{\mathrm{e}}\} + \mathbf{D}_{\sigma}\{\mathrm{e}\} + \{\mathrm{j}_s\}, \tag{3.2}$$

where a dot above a letter denotes the first-order time derivative, $\{\mathrm{j}_s\}$ represents a current source vector, and $\mathbf{D}_{\mu}$, $\mathbf{D}_{\sigma}$ and $\mathbf{D}_{\epsilon}$ are diagonal matrices of permeability, conductivity, and permittivity respectively.

Based on the patch-based single-grid formulation, each row of $\mathbf{S}_{\mathrm{e}}$ in (3.1) corresponds to one patch in the grid, and when multiplied by $\{\mathrm{e}\}$, it produces the magnetic field located at the patch center and normal to the patch. Take the i-th row of $\mathbf{S}_{\mathrm{e}}$ as an example, it can be written as

$$\mathbf{S}_{\mathrm{e}}^{(\mathrm{i})} = \{-\frac{1}{L_{\mathrm{i}}}, \frac{1}{L_{\mathrm{i}}}, \frac{1}{W_{\mathrm{i}}}, -\frac{1}{W_{\mathrm{i}}}\} \oplus zeros(1, N_{\mathrm{e}}), \tag{3.3}$$

which has only four nonzero elements, and L_{i} and W_{i} are the two side lengths of the i-th patch. A reference normal direction is defined for every patch, which is also $\mathbf{H}$'s reference direction on the patch. Using the right hand rule, with the right thumb pointing to the reference normal direction, if the electric field edge's direction is along the direction encircling the normal direction, then a plus sign is used; otherwise, a negative sign appears in (3.3). The $\oplus$ denotes an extended addition by adding the four nonzero elements upon a zero vector of length N_{e}, based on the global indexes of the four electric field unknowns on the patch. Similarly, for the i-th patch, we generate a column vector

$$\mathbf{S}_{h}^{(\mathrm{i})} = \{-\frac{1}{L_{\mathrm{i}}^{a1}}, \frac{1}{L_{\mathrm{i}}^{a2}}, \frac{1}{W_{\mathrm{i}}^{a1}}, -\frac{1}{W_{\mathrm{i}}^{a2}}\}^{T} \oplus zeros(N_{\mathrm{e}}, 1), \tag{3.4}$$

where L_{i}^{a1} or L_{i}^{a2} are the averaged side length of the i-th patch with its left neighbor patch or right neighbor patch. The plus sign or negative sign is determined by the same right hand rule as that in $\mathbf{S}_{\mathrm{e}}^{(\mathrm{i})}$. Similarly, W_{i}^{a1} or W_{i}^{a2} are the averaged side length along the other direction. When the mesh is uniform, $\mathbf{S}_{h}^{(\mathrm{i})}$ is nothing but the transpose of (3.3), thus $\mathbf{S}_h = \mathbf{S}_{\mathrm{e}}^{T}$. As can be seen, a column i of $\mathbf{S}_h$ has also at most four nonzero entries, located at the rows corresponding to the four electric fields of patch i.

Eliminating $\{h\}$ from (3.1) and (3.2), we obtain

$$\mathbf{D}_{\epsilon}\{\ddot{\mathrm{e}}\} + \mathbf{D}_{\sigma}\{\dot{\mathrm{e}}\} + \mathbf{S}\{\mathrm{e}\} = -\{\dot{\mathrm{j}_s}\}, \tag{3.5}$$

where $\mathbf{S}$ can be represented as

$$\mathbf{S} = \mathbf{S}_h \mathbf{D}_{\mu^{-1}} \mathbf{S}_{\mathrm{e}} = \sum_{\mathrm{i}=1}^{N_h} \mu_{\mathrm{i}}^{-1} \left(\mathbf{S}_{h}^{(\mathrm{i})}\right)_{N_{\mathrm{e}}\times 1} \left(\mathbf{S}_{\mathrm{e}}^{(\mathrm{i})}\right)_{1\times N_{\mathrm{e}}}, \tag{3.6}$$

which is a sum of the rank-1 matrix $\mathbf{S}_{h}^{(\mathrm{i})}\mathbf{S}_{\mathrm{e}}^{(\mathrm{i})}$ over all the patches.

3.2.2 Disassemble Contributions of Subdomains

From (3.5) and (3.6), we can analyze how the equations in different subdomains are assembled in the FDTD to simulate the entire problem. Consider two subdomains, (3.5) can be rewritten as

$$\mathbf{D}_{\epsilon}\{\ddot{\mathrm{e}}\} + \mathbf{D}_{\sigma}\{\dot{\mathrm{e}}\} = -\left[\mathbf{S}_{h,1}\mathbf{D}_{\mu_1^{-1}}\mathbf{S}_{\mathrm{e},1} + \mathbf{S}_{h,2}\mathbf{D}_{\mu_2^{-1}}\mathbf{S}_{\mathrm{e},2}\right]\{\mathrm{e}\}, \tag{3.7}$$

where $\mathbf{S}_{h,1(2)}$ has all the column vectors generated from the patches in subdomain 1 (2), and $\mathbf{S}_{\mathrm{e},1(2)}$ comprises all the row vectors from the patches in subdomain 1 (2). Here, the source term is omitted to focus on the assembling mechanism in the FDTD. As can been seen, the total $\mathbf{S}$ is an addition of each subdomain's $\mathbf{S}$, which is the same as the assembling procedure in a finite-element method (FEM). Each patch's $\mathbf{S}$ is assembled to obtain a global $\mathbf{S}$ based on the index of a global unknown vector. However, the $\mathbf{D}_{\epsilon}$ and $\mathbf{D}_{\sigma}$ are not added up from each subdomain's contribution. They are diagonal matrices, whose entries are the permittivity or conductivity at the corresponding e's location. Shall they be assembled from each patch's contribution like that in a finite-element method, then the diagonal entry would be a multiple of the permittivity or conductivity.

Based on (3.7), we can express the electric filed unknown as the addition of two contributions: one is from subdomain 1, expressed by the first term of the right hand side of (3.7); the other is from subdomain 2, represented by the second term. Hence, we can rewrite (3.7) as a two-row system:

$$\mathbf{D}_\epsilon\{\ddot{\mathrm{e}}\}_1 + \mathbf{D}_\sigma\{\dot{\mathrm{e}}\}_1 = -\mathbf{S}_{h,1}\mathbf{D}_{\mu_1^{-1}}\mathbf{S}_{\mathrm{e},1}\{\mathrm{e}\} \tag{3.8}$$

$$\mathbf{D}_\epsilon\{\ddot{\mathrm{e}}\}_2 + \mathbf{D}_\sigma\{\dot{\mathrm{e}}\}_2 = -\mathbf{S}_{h,2}\mathbf{D}_{\mu_2^{-1}}\mathbf{S}_{\mathrm{e},2}\{\mathrm{e}\}, \tag{3.9}$$

with

$$\{\mathrm{e}\} = \{\mathrm{e}\}_1 + \{\mathrm{e}\}_2, \tag{3.10}$$

which stitches the two subdomains together. Neither $\{\mathrm{e}\}_1$ nor $\{\mathrm{e}\}_2$ provides a complete solution of $\{\mathrm{e}\}$. This is because for an interface e unknown between subdomain 1 and subdomain 2, (3.8) yields the curl of **H** from domain 1 patches, and (3.9) generates the curl of **H** from domain 2 patches, and the addition shown in (3.10) is required to complete the whole curl of **H** operation to produce the electric field on the interface.

Algorithm 2 Disassemble contribution $\{\mathrm{e}\}_\mathrm{i}$ and $\mathbf{S}_\mathrm{i}$ in i-th subdomain

```
1:  for every subdomain, subdomain index i := 1 to $p$ do
2:      Re-index all $N_{\mathrm{e_i}}$ edges (unknowns) in i-th subdomain as a vector $\{\mathrm{e}\}_\mathrm{i}$
3:      for every patch in i-th subdomain, patch index j := 1 to $N_{h_\mathrm{i}}$ do
4:          Generate $\mathbf{S}_\mathrm{e}^{(\mathrm{j})}$ at j-th patch as in (3.3)
5:          Generate $\mathbf{S}_h^{(\mathrm{j})}$ at j-th patch as in (3.4)
6:          if j-th patch is on the interface and shared by another subdomain then
7:              Divide $\mathbf{S}_h^{(\mathrm{j})}$ by 2
8:          end if
9:      end for
10:     Assemble $\mathbf{S}_\mathrm{i} = \sum_{\mathrm{j}=1}^{N_{h_\mathrm{i}}} \mu_\mathrm{j}^{-1} \left(\mathbf{S}_h^{(\mathrm{j})}\right)_{N_{\mathrm{e_i}}\times 1} \left(\mathbf{S}_\mathrm{e}^{(\mathrm{j})}\right)_{1\times N_{\mathrm{e_i}}}$ over all $N_{h_\mathrm{i}}$ patches in i-th subdomain
11: end for
```

Generally, when the entire computation domain is partitioned into p subdomains, we can rewrite (3.5) by contributions from each subdomain

$$\left\{\begin{array}{c}\mathbf{D}_{\epsilon_1}\{\ddot{\mathrm{e}}\}_1+\mathbf{D}_{\sigma_1}\{\dot{\mathrm{e}}\}_1\\ \vdots \\ \mathbf{D}_{\epsilon_p}\{\ddot{\mathrm{e}}\}_p+\mathbf{D}_{\sigma_p}\{\dot{\mathrm{e}}\}_p\end{array}\right\}+\left[\begin{array}{ccc}\mathbf{S}_1 & \cdots & \mathbf{S}_1\\ \vdots & \ddots & \vdots\\ \mathbf{S}_p & \cdots & \mathbf{S}_p\end{array}\right]\left\{\begin{array}{c}\{\mathrm{e}\}_1\\ \vdots \\ \{\mathrm{e}\}_p\end{array}\right\}=-\left\{\begin{array}{c}\{\dot{\mathrm{j}}_s\}_1\\ \vdots \\ \{\dot{\mathrm{j}}_s\}_p\end{array}\right\}, \tag{3.11}$$

with entire domain solution

$$\{\mathrm{e}\}=\{\mathrm{e}\}_1+\cdots+\{\mathrm{e}\}_p. \tag{3.12}$$

Here, $\{\mathrm{e}\}_\mathrm{i}$ is the contribution to solution vector $\{\mathrm{e}\}$ from i-th subdomain and it only includes $N_{\mathrm{e_i}}$ edges (unknowns) residing in this local subdomain. Notice that one edge can be shared by m subdomains, in which case, the solution at this edge should be the sum of those m $\{\mathrm{e}\}_\mathrm{i}$s. Because each subdomain has its own indexing for edges in $\{\mathrm{e}\}_\mathrm{i}$, one need to carefully find the correct index of this shared edge. $\mathbf{D}_{\mu_\mathrm{i}}$, $\mathbf{D}_{\sigma_\mathrm{i}}$ and $\mathbf{D}_{\epsilon_\mathrm{i}}$ are local diagonal matrices of permeability, conductivity, and permittivity respectively in i-th subdomain. $\mathbf{S}_\mathrm{i}=\mathbf{S}_{h,\mathrm{i}}\mathbf{D}_{\mu_\mathrm{i}^{-1}}\mathbf{S}_{\mathrm{e,i}}$, where $\mathbf{S}_{h,\mathrm{i}}$ has all the column vectors generated from the patches in domain i, and $\mathbf{S}_{\mathrm{e,i}}$ comprises all the row vectors from the patches in domain i. Because of the way that stiffness matrix $\mathbf{S}$ is assembled in (3.6) in patch-based single-grid FDTD, we can clearly extract the contribution $\mathbf{S}_\mathrm{i}$ in i-th subdomain.

The pseudo-code to disassemble contributions of subdomains is shown in Algorithm 2. In equation (3.11) and Algorithm 2, we choose to disassemble $\{\mathrm{e}\}_\mathrm{i}$ and $\mathbf{S}_\mathrm{i}$ in i-th subdomain into local size $N_{\mathrm{e_i}}$. Actually, a straight forward way is to keep the size of $\{\mathrm{e}\}_\mathrm{i}$ and $\mathbf{S}_\mathrm{i}$ as the original N_e, which is the number of all unknowns in the entire domain. In keeping the original size N_e, only a few all zeros rows or columns are included and there is no need to give special consideration for interface edges. Also, summing up the rows in (3.11) can directly recover the Maxwell's equations in the entire domain. However, those extra rows and columns can significantly downgrade the performance of LU factorization on the local system in the subsequent parallel solver. Therefore, we still choose to disassemble to local size $N_{\mathrm{e_i}}$, where $\{\mathrm{e}\}_\mathrm{i}$ is a vector of size $N_{\mathrm{e_i}}\times 1$ and $\mathbf{S}_\mathrm{i}$ is a sparse matrix a size $N_{\mathrm{e_i}}\times N_{\mathrm{e_i}}$. Note that the different local sizes immediately mathematically invalid the block matrix-vector product

$\mathbf{S}_\mathrm{i}\{\mathrm{e}\}_\mathrm{j}$ in (3.11) when subdomain index $\mathrm{j} \neq \mathrm{i}$. However, the physical meaning of $\mathbf{S}_\mathrm{i}\{\mathrm{e}\}_\mathrm{j}$ is simply that the solution contribution $\{\mathrm{e}\}_\mathrm{j}$ at j-th subdomain will change the solution at i-th subdomain. Because of the locality of curl-curl operator in finite difference method, only interface edges have contribution in $\mathbf{S}_\mathrm{i}\{\mathrm{e}\}_\mathrm{j}$. We can mathematically obtain the result of $\mathbf{S}_\mathrm{i}\{\mathrm{e}\}_\mathrm{j}$ by reducing the original form of $\mathbf{S}_\mathrm{i}\{\mathrm{e}\}_\mathrm{j}$ when both $\mathbf{S}_\mathrm{i}$ and $\{\mathrm{e}\}_\mathrm{j}$ are defined by entire unknown size N_e. Equivalently, what one need to do with the off-diagonal blocks in (3.11) is to separate

$$\begin{bmatrix} \mathbf{S}_1 & \cdots & \mathbf{S}_1 \\ \vdots & \ddots & \vdots \\ \mathbf{S}_p & \cdots & \mathbf{S}_p \end{bmatrix} \begin{Bmatrix} \{\mathrm{e}\}_1 \\ \vdots \\ \{\mathrm{e}\}_p \end{Bmatrix} = \begin{Bmatrix} \mathbf{S}_1\{\mathrm{e}\}_1 \\ \vdots \\ \mathbf{S}_p\{\mathrm{e}\}_p \end{Bmatrix} + \begin{bmatrix} 0 & \cdots & \mathbf{S}_1 \\ \vdots & \ddots & \vdots \\ \mathbf{S}_p & \cdots & 0 \end{bmatrix} \begin{Bmatrix} \{\mathrm{e}\}_1 \\ \vdots \\ \{\mathrm{e}\}_p \end{Bmatrix}, \tag{3.13}$$

where the first term in right hand side comes from diagonal block matrices and represents the interaction of each subdomain to itself. The second term in right hand side comes from off-diagonal block matrices and represents the interaction with all neighbor subdomains. So, we can define the interaction with all neighbor subdomains as

$$\begin{Bmatrix} \{b_{ns}\}_1 \\ \vdots \\ \{b_{ns}\}_p \end{Bmatrix} = \begin{bmatrix} 0 & \cdots & \mathbf{S}_1 \\ \vdots & \ddots & \vdots \\ \mathbf{S}_p & \cdots & 0 \end{bmatrix} \begin{Bmatrix} \{\mathrm{e}\}_1 \\ \vdots \\ \{\mathrm{e}\}_p \end{Bmatrix} = \begin{Bmatrix} \mathbf{S}_1\{\mathrm{e}_{interface}\}_1 \\ \vdots \\ \mathbf{S}_p\{\mathrm{e}_{interface}\}_p \end{Bmatrix}, \tag{3.14}$$

with

$$\{\mathrm{e}_{interface}\}_\mathrm{i} = \sum_{\mathrm{j}=1,\mathrm{j}\neq\mathrm{i}}^{p} \{\mathrm{e}\}_\mathrm{j}, \tag{3.15}$$

which is later cascaded into corresponding edges in i-th subdomain. The $\{\mathrm{e}_{interface}\}_\mathrm{i}$ is a vector of size $N_{\mathrm{e}_\mathrm{i}} \times 1$ and represents the e's contributions from all neighbor subdomains. To implement (3.15), one can first identify the m interface edges in i-th subdomain, then collect all neighbor subdomains' contribution $\{\mathrm{e}\}_\mathrm{j}$ at those m edges, the sum of which becomes corresponding m nonzero values in vector $\{\mathrm{e}_{interface}\}_{\mathrm{i}}$. As a result, we can avoid mathematically invalid $\mathbf{S}_{\mathrm{i}}\{\mathrm{e}\}_{\mathrm{j}}$ and transform (3.11) to

$$\left\{\begin{array}{c}\mathbf{D}_{\epsilon_1}\{\ddot{\mathrm{e}}\}_1+\mathbf{D}_{\sigma_1}\{\dot{\mathrm{e}}\}_1+\mathbf{S}_1\{\mathrm{e}\}_1\\ \vdots\\ \mathbf{D}_{\epsilon_p}\{\ddot{\mathrm{e}}\}_p+\mathbf{D}_{\sigma_p}\{\dot{\mathrm{e}}\}_p+\mathbf{S}_p\{\mathrm{e}\}_p\end{array}\right\}=-\left\{\begin{array}{c}\{b_{ns}\}_1+\{\dot{\mathrm{j}}_s\}_1\\ \vdots\\ \{b_{ns}\}_p+\{\dot{\mathrm{j}}_s\}_p\end{array}\right\}. \tag{3.16}$$

Here, each row represents one subdomain, which already shows a high potential to be paralleled.

The disassembling of $\{\mathrm{e}\}_{\mathrm{i}}$ and $\mathbf{S}_{\mathrm{i}}$ in i-th subdomain is mathematically rigorous without any approximation. What's more, the physical interactions among subdomains through curl of $\mathbf{H}$ are clearly revealed in the disassembled system (3.16). All those are necessary to have a fast convergence of the proposed parallel iterative solver.

3.2.3 Parallel Iteration Scheme in Frequency Domain

In frequency domain, simply replace all $\frac{\partial}{\partial t}=\mathrm{j}\omega$, then (3.16) becomes

$$\left\{\begin{array}{c}(\omega^2\mathbf{D}_{\epsilon_1}-\mathrm{j}\omega\mathbf{D}_{\sigma_1}-\mathbf{S}_1)\{\mathrm{e}\}_1\\ \vdots\\ (\omega^2\mathbf{D}_{\epsilon_p}-\mathrm{j}\omega\mathbf{D}_{\sigma_p}-\mathbf{S}_p)\{\mathrm{e}\}_p\end{array}\right\}=\left\{\begin{array}{c}\{b_{ns}\}_1+\mathrm{j}\omega\{\mathrm{j}_s\}_1\\ \vdots\\ \{b_{ns}\}_p+\mathrm{j}\omega\{\mathrm{j}_s\}_p\end{array}\right\}. \tag{3.17}$$

The $\mathbf{A}_{\mathrm{i}}=\omega^2\mathbf{D}_{\epsilon_{\mathrm{i}}}-\mathrm{j}\omega\mathbf{D}_{\sigma_{\mathrm{i}}}-\mathbf{S}_{\mathrm{i}}$ is the local system matrix in the i-th subdomain. $\mathbf{A}_{\mathrm{i}}$ can be generated locally according to Algorithm 2. Compared to the system matrix $\mathbf{A}$ of the entire domain, after partitioning into many small subdomains, $\mathbf{A}_{\mathrm{i}}$ can have a much smaller matrix size thus is much cheaper to solve. Besides, formula (3.17) allows solving each subdomain in parallel, where the communication among neighbor subdomains is only $\{b_{ns}\}_{\mathrm{i}}$ at interface edges. The communication cost is just passing interface $\{\mathrm{e}\}_{\mathrm{i}}$ to its neighbor subdomains, thus the parallel overhead is not very large. According to above analysis on (3.17), we propose such a parallel iteration scheme:

$$\left\{ \begin{array}{c} \{\mathrm{e}\}_1^{nIter+1} \\ \vdots \\ \{\mathrm{e}\}_p^{nIter+1} \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{A}_1^{-1}(\{b_{ns}\}_1^{nIter} + \mathrm{j}\omega\{\mathrm{j}_s\}_1) \\ \vdots \\ \mathbf{A}_p^{-1}(\{b_{ns}\}_p^{nIter} + \mathrm{j}\omega\{\mathrm{j}_s\}_p) \end{array} \right\}, \tag{3.18}$$

where $nIter$ means the iteration step. In i-th subdomain, $\mathbf{A}_\mathrm{i} = \omega^2\mathbf{D}_{\epsilon_\mathrm{i}} - \mathrm{j}\omega\mathbf{D}_{\sigma_\mathrm{i}} - \mathbf{S}_\mathrm{i}$ is local system matrix, $\{b_{ns}\}_\mathrm{i}^{nIter} = \mathbf{S}_\mathrm{i}\{\mathrm{e}_{interface}\}_\mathrm{i}^{nIter}$ is the interaction with all neighbor subdomains at $nIter$-th iteration step. For on-chip problems, a good initial guess to start the iteration is $\{\mathrm{e}\} = 0$.

We use relative residual of (3.18) as the convergence criterion. To make the relative residual clear, we can let

$$\tilde{\mathbf{A}} = \mathrm{diag}(\mathbf{A}_1, \cdots, \mathbf{A}_p), \quad \{b\}_\mathrm{i}^{nIter} = \{b_{ns}\}_\mathrm{i}^{nIter} + \mathrm{j}\omega\{\mathrm{j}_s\}_\mathrm{i}, \tag{3.19}$$

and

$$\{\tilde{\mathrm{e}}\}^{nIter} = \left\{ \begin{array}{c} \{\mathrm{e}\}_1^{nIter} \\ \vdots \\ \{\mathrm{e}\}_p^{nIter} \end{array} \right\}, \quad \{\tilde{b}\}^{nIter} = \left\{ \begin{array}{c} \{b\}_1^{nIter} \\ \vdots \\ \{b\}_p^{nIter} \end{array} \right\}. \tag{3.20}$$

Then, the iteration in (3.18) is simply

$$\tilde{\mathbf{A}}\{\tilde{\mathrm{e}}\}^{nIter+1} = \{\tilde{b}\}^{nIter}. \tag{3.21}$$

So, a natural definition of relative residual with Euclidean norm is

$$relaRes\_b = \frac{\left\| \{\tilde{\mathbf{A}}\{\tilde{\mathrm{e}}\}^{nIter+1} - \{\tilde{b}\}^{nIter+1} \right\|}{\left\| \{\tilde{b}\}^{nIter+1} \right\|} = \frac{\left\| \{\tilde{b}\}^{nIter} - \{\tilde{b}\}^{nIter+1} \right\|}{\left\| \{\tilde{b}\}^{nIter+1} \right\|}. \tag{3.22}$$

In parallel coding, to save communication cost, instead of collecting the entire vector $\{\tilde{b}\}^{nIter}$ from all subdomains, a cheaper way is to only collect $\left\|\{\tilde{b}\}_{\mathrm{i}}^{nIter+1}\right\|$ and $\left\|\{\tilde{b}\}_{\mathrm{i}}^{nIter} - \{\tilde{b}\}_{\mathrm{i}}^{nIter+1}\right\|$ from subdomain i. Then, under Euclidean norm,

$$\left\|\{\tilde{b}\}^{nIter+1}\right\| = \sqrt{\sum_{\mathrm{i}=1}^{p} \left\|\{\tilde{b}\}_{\mathrm{i}}^{nIter+1}\right\|^2}. \tag{3.23}$$

Similar trick can be used to get $\left\|\{\tilde{b}\}^{nIter} - \{\tilde{b}\}^{nIter+1}\right\|$. In our implementation of the parallel iterative solver (3.18), we also limit the maximum number of iterations. Thus, we stop the iteration when

$$relaRes\_b < 10^{-3} \quad \text{or} \quad nIter > 200. \tag{3.24}$$

To compare with the proposed parallel solver, a direct solver in frequency domain directly solve

$$(\omega^2 \mathbf{D}_\epsilon - \mathrm{j}\omega \mathbf{D}_\sigma - \mathbf{S})\{\mathrm{e}\} = \mathrm{j}\omega\{\mathrm{j}_s\}, \tag{3.25}$$

where all matrices and vectors are in the entire domain. Most direct matrix solvers like the PARDISO function in Intel MKL library and the backslash in Matlab will first LU factorize the system matrix, then solve much cheaper L matrix and U matrix. Because LU factorization for non-symmetric system matrix in (3.25) is not scaling that well as seen in later numerical experiments, the proposed parallel solver can save a lot of time by LU factorizing many much smaller system matrices in parallel.

3.2.4 Parallel Iteration Scheme in Time Domain

The idea that partitioning the entire domain into smaller subdomains then solving smaller system matrices in parallel can also be applied to time domain finite difference solvers. In time domain simulation, many unconditionally stable FDMs, such as CN-FDTD [44] and backward-difference based FDTD [20], reply on solving a big matrix to get rid of the limit of the small CFL time step. We find that backward-difference based FDTD can adopt a similar parallel iteration scheme as the frequency domain one in (3.18).

An unconditionally stable backward-difference based FDTD [20] discretizes time domain Maxwell's equations (3.5) into

$$\mathbf{D}_\epsilon(\{\mathrm{e}\}^{n+1} - 2\{\mathrm{e}\}^{n} + \{\mathrm{e}\}^{n-1}) + \Delta t\mathbf{D}_\sigma(\{\mathrm{e}\}^{n+1} - \{\mathrm{e}\}^{n}) + \Delta t^2\mathbf{S}\{\mathrm{e}\}^{n+1} = -\Delta t^2\{\dot{\mathrm{j}_s}\}^{n+1}, \tag{3.26}$$

where Δt is the time step and n means n-th time instant. By putting the latest time instant $\{\mathrm{e}\}^{n+1}$ on the left hand side, we obtain the time marching

$$(\mathbf{D}_\epsilon + \Delta t\mathbf{D}_\sigma + \Delta t^2\mathbf{S})\{\mathrm{e}\}^{n+1} = (2\mathbf{D}_\epsilon + \Delta t\mathbf{D}_\sigma)\{\mathrm{e}\}^{n} - \mathbf{D}_\epsilon\{\mathrm{e}\}^{n-1} - \Delta t^2\{\dot{\mathrm{j}_s}\}^{n+1}. \tag{3.27}$$

A direct solver in time domain will factorize the asymmetric system matrix $\mathbf{D}_\epsilon+\Delta t\mathbf{D}_\sigma+\Delta t^2\mathbf{S}$ then solve the factorized matrices. However, as analyzed before in frequency domain, the factorization can be very expensive in terms of memory and CPU time for super large system matrix. So, a parallelization based on the dissembled system matrix can also be applied here.

Table 3.1. One-on-One Mapping between Time Domain and Frequency Domain

	Time Domain in (3.27)	Frequency Domain in (3.25)
solution	$\{\mathrm{e}\}^{n+1}$	$\{\mathrm{e}\}$
system matrix $\mathbf{A}$	$\mathbf{D}_\epsilon + \Delta t\mathbf{D}_\sigma + \Delta t^2\mathbf{S}$	$\omega^2\mathbf{D}_\epsilon - \mathrm{j}\omega\mathbf{D}_\sigma - \mathbf{S}$
stiffness matrix	$\Delta t^2\mathbf{S}$	$-\mathbf{S}$
right hand side	$(2\mathbf{D}_\epsilon + \Delta t\mathbf{D}_\sigma)\{\mathrm{e}\}^{n} -$ $\mathbf{D}_\epsilon\{\mathrm{e}\}^{n-1} - \Delta t^2\{\dot{\mathrm{j}_s}\}^{n+1}$	$\mathrm{j}\omega\{\mathrm{j}_s\}$

Due to the similarity between the time marching (3.27) in time domain and the dominant equation (3.25) in frequency domain, we can solve each time step in (3.27) with a similar parallel iteration scheme as (3.18). To get the parallel iteration scheme in time domain, we only need to do a one-on-one mapping between time domain (3.27) and frequency domain (3.25). The mapping is listed in Table 3.1. Let's define the right hand side of (3.27) as

$$\{rhs\}^{n+1} = (2\mathbf{D}_\epsilon + \Delta t\mathbf{D}_\sigma)\{\mathrm{e}\}^{n} - \mathbf{D}_\epsilon\{\mathrm{e}\}^{n-1} - \Delta t^2\{\dot{\mathrm{j}_s}\}^{n+1}, \tag{3.28}$$

which can be viewed as a constant vector when iteratively solving $\{\mathrm{e}\}^{n+1}$. After a one-on-one mapping from the frequency domain parallel iteration scheme (3.18), we have such a time domain parallel iteration scheme to solve $\{\mathrm{e}\}^{n+1}$ at $(n+1)$-th time instant:

$$\left\{\begin{array}{c} \{\mathrm{e}\}_1^{nIter+1,n+1} \\ \vdots \\ \{\mathrm{e}\}_p^{nIter+1,n+1} \end{array}\right\} = \left\{\begin{array}{c} \mathbf{A}_1^{-1}(\{b_{ns}\}_1^{nIter,n+1} + \{rhs\}_1^{n+1}) \\ \vdots \\ \mathbf{A}_p^{-1}(\{b_{ns}\}_p^{nIter,n+1} + \{rhs\}_p^{n+1}) \end{array}\right\}. \tag{3.29}$$

Here, $(n+1)$ denotes the time instant and $nIer$ denotes the iteration step. During the iteration, all time-related terms have the same time instant $(n+1)$. Only when moving to the next time instant, will the $\{rhs\}$ be updated. In i-th subdomain, the terms in time domain iteration (3.29) are summarized here:

$$\mathbf{A}_\mathrm{i} = \mathbf{D}_{\epsilon_\mathrm{i}} + \Delta t \mathbf{D}_{\sigma_\mathrm{i}} + \Delta t^2 \mathbf{S}_\mathrm{i},$$

$$\{b_{ns}\}_\mathrm{i}^{nIter,n+1} = -\Delta t^2 \mathbf{S}_\mathrm{i} \{\mathrm{e}_{interface}\}_\mathrm{i}^{nIter,n+1}.$$

$$\{\mathrm{e}_{interface}\}_\mathrm{i}^{nIter,n+1} = \sum_{\mathrm{j}=1,\mathrm{j}\neq\mathrm{i}}^{p} \{\mathrm{e}\}_\mathrm{j}^{nIter,n+1},$$

$$\{rhs\}_\mathrm{i}^{n+1} = (2\mathbf{D}_{\epsilon_\mathrm{i}} + \Delta t \mathbf{D}_{\sigma_\mathrm{i}})\{\mathrm{e}\}_\mathrm{i}^{n} - \mathbf{D}_{\epsilon_\mathrm{i}}\{\mathrm{e}\}_\mathrm{i}^{n-1} - \Delta t^2 \{\dot{\mathrm{j}}_s\}_\mathrm{i}^{n+1}.$$

Because the field distribution typically will not change dramatically between two time steps, the solution at previous time step is used as the initial guess to start the iteration in (3.29). Because the initial guess is already close to the solution, the number of iterations is not large. From our numerical testing, most iterations can converge to $relaRes\_b < 10^{-3}$ within 20 iterations.

The proposed parallel iterative solver in time domain is still unconditionally stable thus allows a large time step. Also, since only much smaller local system matrices are solved in parallel, both memory and CPU time can be saved relative to a direct solver.

3.3 Numerical Results in Frequency Domain

Next, we show numerical results in both frequency domain and time domain. In all those examples, when the entire computational domain is partitioned into p subdomains, p CPUs are used so that each CPU only solves one subdomain with single thread. Both the parallel solver and the direct solver are run in a server where 20 CPUs and 256 GB shared memory are available. The direct solver solves the entire system matrix with LU factorization, whereas our proposed parallel solver only solves system matrix $\mathbf{A}$ in each subdomain. The direct solver uses Matlab built-in function backslash, whereas parallel solver uses "spmd" in Matlab.

3.3.1 Accuracy and Convergence

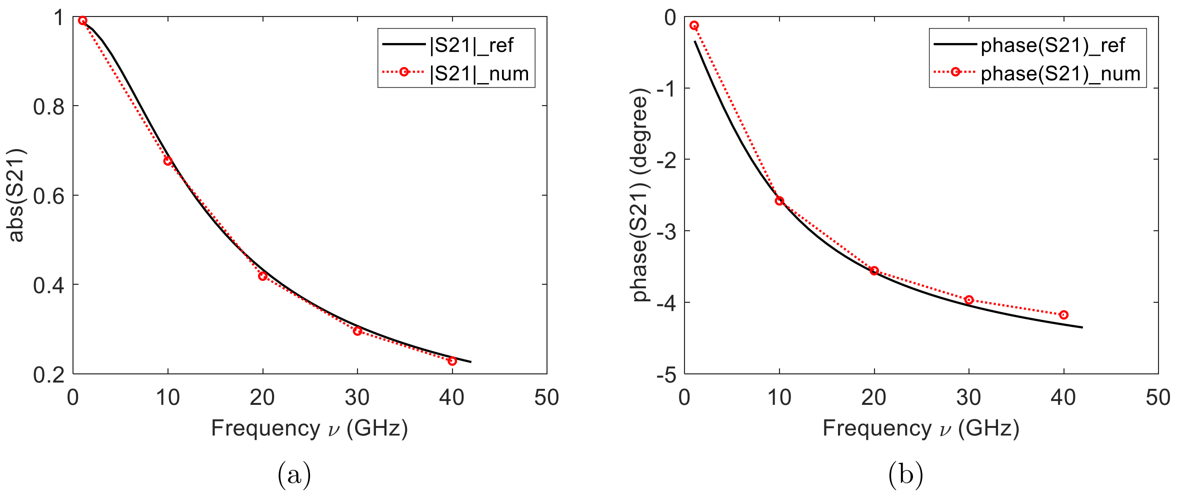
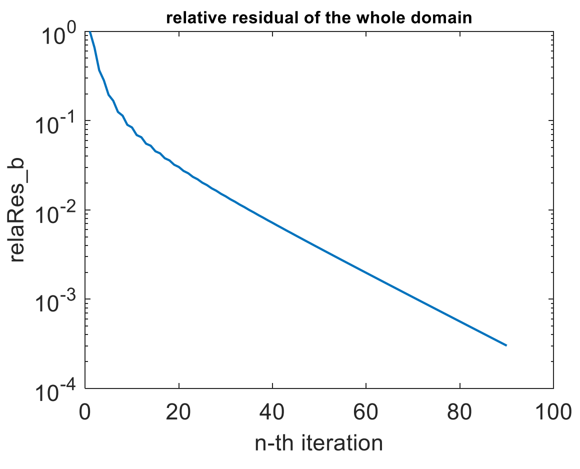
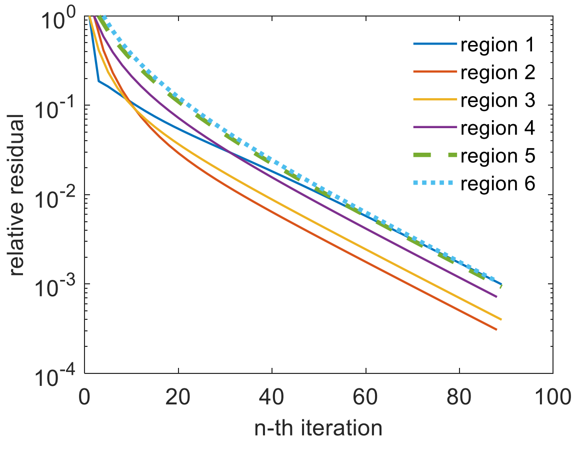


Figure 3.2. Amplitude and phase of S_{21} of the test chip interconnect.

We first validate the accuracy of the parallel solver by solving the test-chip interconnect example in Fig. 2.4. The same port 1 and port 2 are used. This time, a non-uniform mesh is used. Along stack growth direction z, 7 stacks are discretized into mesh sizes $N_z = [3, 3, 7, 3, 2, 2, 3]$ from bottom to top. The 100 μm-long interconnect is padded with 30 μm-long air at front and back. The 100 μm length along y is discretized into $N_y = 20$, and each 30 μm air padding is discretized into $N_y = 6$. The central wire is discretized into $N_x = 10$ along x direction. In the parallel solver, the entire structure is equally partitioned into 6 subdomains (regions) along y direction. As shown in Fig. 3.2, the obtained S_{21} from parallel solver matches very well with reference results. In Fig. 3.3 (a), the $relaRes\_b$ of the entire domain solution quickly converges to a very small value. Fig. 3.3 (b) shows that, for this example, every subdomain has its own $relaRes^{\mathrm{i}}\_b$ converged at a similar speed as that of the entire domain solution, even through the subdomains far from excitation should have worse accuracy.



(a)



(b)

Figure 3.3. Convergence in terms of $relaRes\_b$ of the test chip interconnect at 1 GHz.

3.3.2 Speed up Relative to Direct Solver

Table 3.2. Structure information and run time of logic gates inv, nand2, and xor

	Structure info		Direct solver	Parallel solver, (time: LU + iteration)		
	N_{edge}	N_{cell_y}	entire domain	4 CPUs	10 CPUs	20 CPUs
inv	113,664	79	116.3 s	17.4+17.9 s	2.02+16.8 s	1.26+32.4 s
nand2	672,759	98	1257 s	162+804 s	21.4+258 s	3.34+171 s
xor	679,578	99	1605 s	154+800 s	20.7+232 s	3.17+177 s

We then simulate three standard logic gate examples as in Fig. 3.4. The frequency is 1 GHz and the structure information of those logic gates is in Table 3.2. The direct solver solves the system matrix of the entire computational domain using 1 CPU and all 256 GB memory. The parallel solver equally partitions the logic gate along y direction into p subdomains,

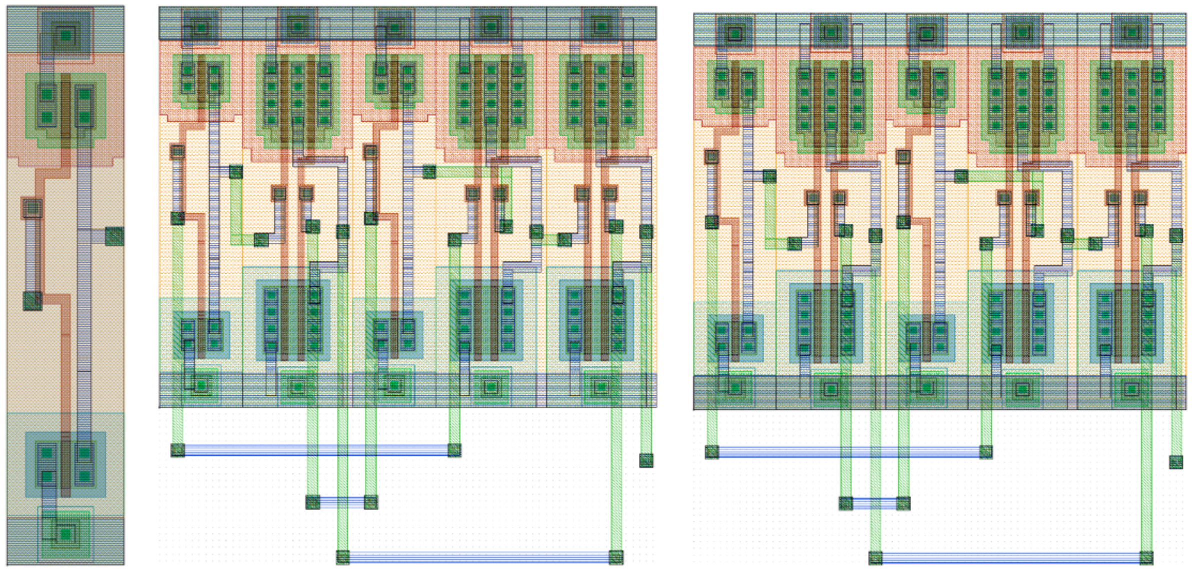

(a) (b) (c)

Figure 3.4. Layout structure of logic gate examples. (a) ”inv”, an inverter (NOT gate). (b) ”nand2”, a NAND gate. (c) ”xor”, a XOR gate.

where each domain is assigned to 1 CPU to solve. The parallel solver first factorizes local system matrix then iterates until convergence. The time of both LU and iteration is given in Table 3.2, from which we can see a sharp decrease of LU time when more subdomains (CPUs) are used in parallel solver. However, as p becomes larger, the communication cost might also increase. The speedup of the parallel solver relative to direct solver is plotted in Fig. 3.5 (a). The speedup no long increases with num of CPUs in ”inv” example because the parallel overhead (communication time) becomes larger in this relative small example. Fig. 3.5 (b) shows the convergence of each logic gate in terms of $relaRes\_b$ when $p = 10$.

3.4 Numerical Results in Time Domain

In this section, we show numerical experiments in time domain.

3.4.1 Accuracy

To show the accuracy of the proposed parallel solver, we first simulate logic gate example ”inv” and ”nand2” in time domain. The same discretization of logic gates is used as in Table

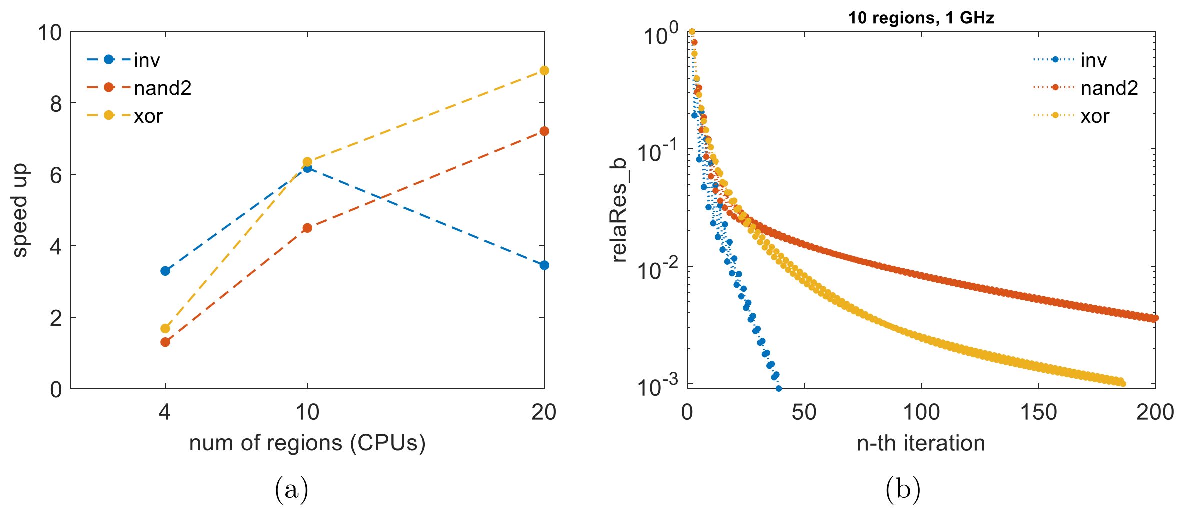


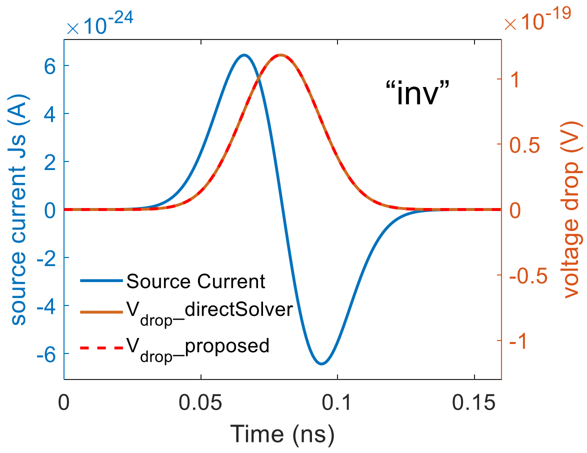
Figure 3.5. Logic gate examples in Fig. 3.4. (a) speed up relative to direct solver. (b) convergence in terms of $relaRes\_b$.

3.2. The parallel solver partitions the structure into 10 subdomains and uses 10 CPUs to solve it. At each time step, limit the maximum number of iteration to be $N_{iter} = 20$ and set convergence criterion as $relaRes\_b = 10^{-4}$. The logic gate is excited with Gaussian derivative source current density $\mathrm{j}_s = -(t-t_0)\exp[-(\frac{t-t_0}{\tau_s})^2]$ A/m^2 ($t_0 = 4\tau_s$, $\tau_s = 2 \times 10^{-11}$ s). The time window $8\tau_s$ of the pulse is equally discretized into 200 time steps according to sampling theorem. Then, the voltage drop between two selected ports or the port voltage relative to ground PEC is plotted in Fig. 3.6, where the response from parallel solver matches very well with that from direct solver.

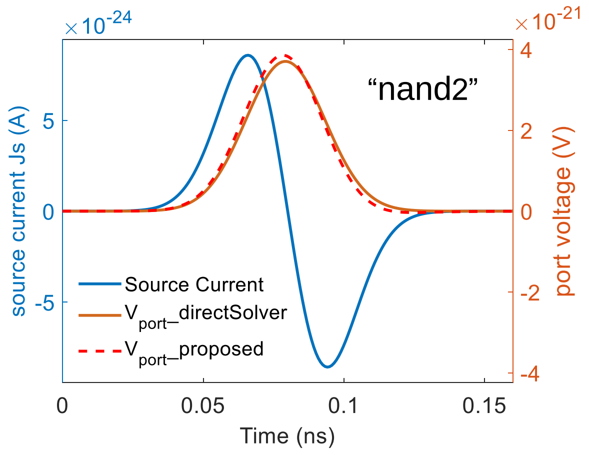
Table 3.3. Structure information and run time of example Intel 4004 processor

Intel 4004	Structure info		Direct solver		Parallel solver	
	N_{edge}	N_{cell_y}	LU	200 time steps	LU	200 time steps
Coarse mesh	1,157,768	262	465 s	1155 s	5.0 s	2023 s
Finer mesh	7,240,090	657	LU out of memory		55.6 s	12401 s

The last example is an Intel 4004 processor (a 4-bit central processing unit), the layout of which is shown in Fig. 3.7. It has 7 layers and over 86,220 objects. A coarse discretization results in 1,157,768 unknowns and a finer discretization results in 7,240,090 unknowns. The conventional FDTD fails to simulate this example in a feasible run time because of the



(a)



(b)

Figure 3.6. Time domain source current and voltage in logic gate example (a) "inv" and (b) "nand2".

requirement of a small time step 10e-17 s for stability. A Gaussian derivative source current is used and discretized into 200 time steps. Limited to the hardware capacity of our server (20 CPUs, 256 GB memory), the parallel solver partitions the structure into 20 subdomains and uses all 20 CPUs. The run time is shown in Table 3.3. The direct solver can barely solve the example in coarse mesh and fails in finer mesh (7 million unknowns). Compared to the direct solver, the parallel solver has a smaller memory requirement, thus can still handle the example in finer mesh. The simulated port voltage is plotted in Fig. 3.8. In coarse mesh where both parallel solver and direct solver can run through, their simulated responses match very well with each other.

3.4.2 Scaling Performance

Here, we show the scaling performance of the parallel solver using previous examples including Intel 4004 processor and logic gates "inv", "nand2", and "xor". A full time domain simulation uses 200 time steps. At each time step, the parallel solver uses iteration to find the accurate solution, where the convergence criterion is set as $relaRes\_b = 10^{-3}$. Also, the maximum number of iteration is limited to $N_{iter} = 20$ for the first 66 time steps and $N_{iter} = 10$ for the rest of time steps. In such a way, we can reduce the accumulation of

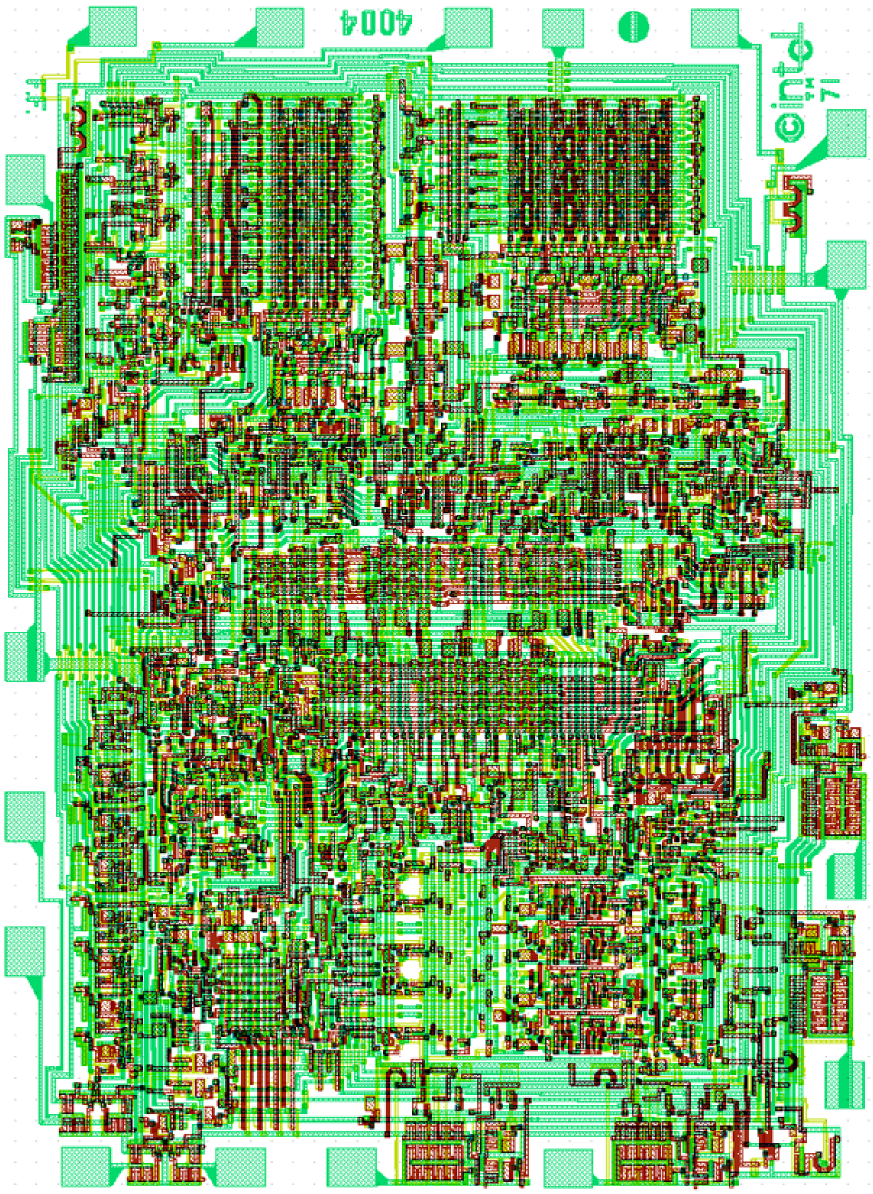
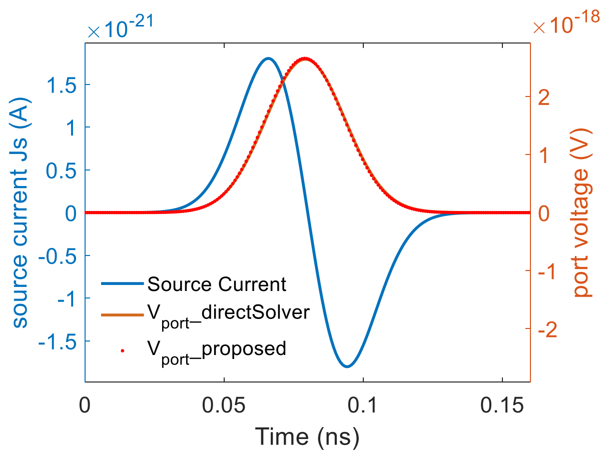


Figure 3.7. Layout structure of Intel 4004 processor.

numerical error at early time steps while not increase the run time too much. The run time at different examples are plotted in Fig. 3.9. In Fig. 3.9 (a), when more subdomains are partitioned into and more CPUs are used, the run time of parallel solver decreases. For smaller example like ”inv”, the run time first decreases then increases because the parallel overhead and communication cost are larger when more CPUs are used. From the data in Fig. 3.9 (b), we can see that, when the number of unknowns is smaller than 10 million, the



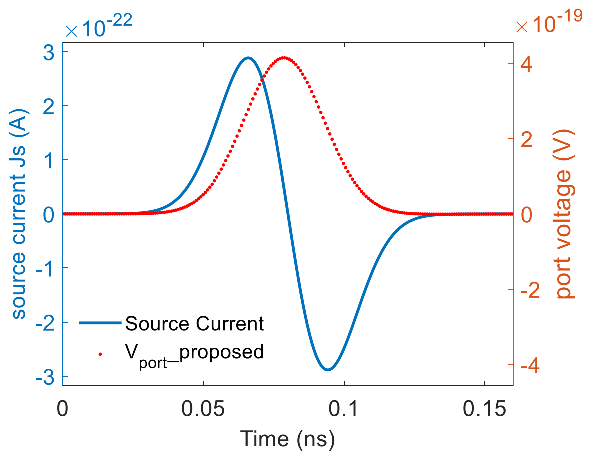
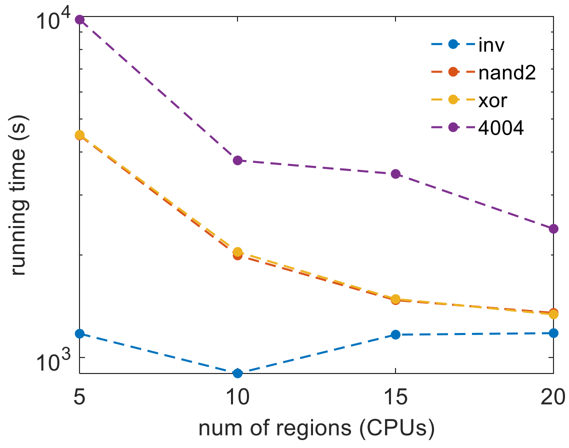


Figure 3.8. Time domain source current and port voltage in Intel 4004 processor. (a) A coarse mesh is used. (b) A finer mesh is used.



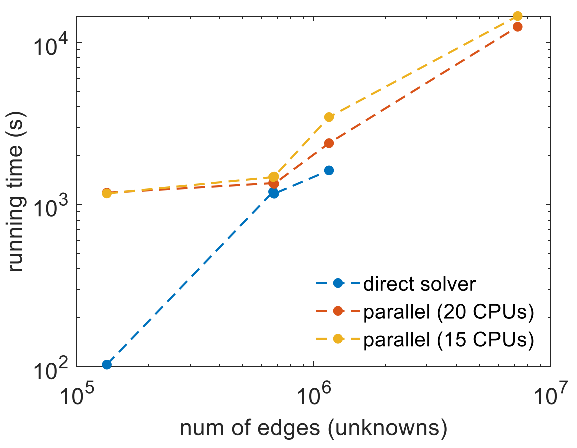


Figure 3.9. Scaling performance of the TD parallel solver.

parallel solver has a linear run time complexity $\sim \mathcal{O}(N_{edge})$ with respect to number of edges (unknowns).

3.5 Conclusion

A non-overlapping domain decomposition parallel algorithm for finite-difference method is developed in this work, which enables simple parallelization by solving each subdomain and adding back contribution from neighbor subdomains. The proposed parallel solver can overcome the large memory complexity of LU factorization while maintaining a controllable accuracy, thereby enables the simulation of large-scale on-chip IC structures.

4. SPLIT-FIELD DOMAIN DECOMPOSITION PARALLEL ALGORITHM WITH FAST CONVERGENCE FOR ELECTROMAGNETIC ANALYSIS

4.1 Introduction

A parallel domain decomposition method (DDM) [17] is popular in solving large-scale problems. Essentially, the DDM is a divide-and-conquer algorithm. To solve a partial differential equation (PDE), the DDM decomposes the whole computational domain into smaller overlapping or non-overlapping subdomains, solves each subdomain individually, then recombines local solutions to provide a global solution.

In open literature, the DDM-based parallelization has been widely used in the PDE-based CEM solvers. The conventional Yee's finite-difference time-domain (FDTD) method has a simple iterative scheme and naturally high parallelism, therefore, it can be paralleled with DDM for various applications [39], [40]. However, the time step of Yee's FDTD is limited by the Courant–Friedrichs–Lewy (CFL) stability condition, which requires way too many time steps to finish a full-package simulation. To eliminate the restriction on the time step, several implicit temporal difference schemes have been developed in recent years, such as alternating direction implicit FDTD (ADI-FDTD) method [41], locally 1-D FDTD (LOD-FDTD) method [42], Laguerre-FDTD method [43], and Crank–Nicolson FDTD (CN-FDTD) method [44]. Both the ADI-FDTD method and LOD-FDTD method involve the solution of tri-diagonal systems along the three directions of Cartesian coordinates, and their parallel implementations with DDM have been introduced in [45] and [18]. However, despite higher computational efficiency, the ADI-FDTD method and LOD-FDTD method suffer from worse accuracy as the time step increases [46], [47]. The Laguerre-FDTD method is an order-marching method that can be paralleled with conformal DDM [48] and nonconformal DDM [19]. The CN-FDTD method has a second order accuracy in both time and space, and has been paralleled with DDM in [49]. At every time step, both the Laguerre-FDTD method and CN-FDTD method need to solve a large sparse matrix equation. The Schur-based parallel DDMs [19], [48], [49] solve this large matrix with a block Gaussian elimination, where each cascaded subdomain can be computed in parallel. DDMs have also been presented in finite-element method (FEM) in both frequency and time domain [50]–[53].

Among existing DDM-based methods, the parallelization is achieved mainly in two approaches. The first approach is a direct solver using Schur complement. The major computational cost of this approach is the dense system of interface unknowns. The second approach is an iterative solver that applies transmission conditions on the interfaces to exchange solutions between subdomains. However, the convergence of this approach is problem dependent.

One common feature in existing DD methods is that the interface field is treated as a whole, and shared in common by adjacent subdomains. The transmission condition can be formulated using tangential **E**, tangential **H**, or their weighted sum on the interface. The interface fields can be viewed as the sources of each subdomain. Given an interface field, the unknowns inside the subdomains are solved in sequential or in parallel, which is essentially how the overall work is partitioned into each subdomain to solve.

In this work, we develop a new split-field DDM. Different from existing methods, we split the interface field e into multiple components, for example, $\mathrm{e} = \mathrm{e}_1 + \mathrm{e}_2 + ... + \mathrm{e}_m$, and let subdomain 1 compute e_1, subdomain 2 compute e_2, etc. The resulting system allows for a natural partition into fully decoupled subsystems. Meanwhile, since each subdomain is not just using interface e as a boundary condition, but also computing part of its solution, a fast convergence can be achieved. We theoretically analyze the convergence of the new DD method, and find it is guaranteed. The communication cost is minimal only involving an addition of interface unknowns at each time step, and hence a communication between adjacent subdomains on a small set of interface unknowns. The proposed DDM algorithm has the advantages of both direct and iterative solutions.

The rest of this chapter is organized as follows. In Section 4.2, we present the preliminaries of this work. In Section 4.3, we elaborate the proposed DD method. In Section 4.4, we discuss details of a parallel implementation. A suite of on-chip and package simulations are then carried out to validate the proposed method in Section 4.5.

4.2 Preliminaries

A PDE-based solution of time domain Maxwell's equations results in the following system of equations

$$\mathbf{D}_\epsilon \frac{\partial^2 \{\mathrm{e}\}}{\partial t^2} + \mathbf{D}_\sigma \frac{\partial \{\mathrm{e}\}}{\partial t} + \mathbf{S}\{\mathrm{e}\} = -\frac{\partial \{\mathrm{j}_s\}}{\partial t}. \tag{4.1}$$

If discretized in time using a backward difference, we obtain [20]

$$\begin{aligned} &(\mathbf{D}_\epsilon + \Delta t \mathbf{D}_\sigma + \Delta t^2 \mathbf{S})\{\mathrm{e}\}^{n+1} = \\ &(2\mathbf{D}_\epsilon + \Delta t \mathbf{D}_\sigma)\{\mathrm{e}\}^n - \mathbf{D}_\epsilon \{\mathrm{e}\}^{n-1} - \Delta t^2 \{\dot{\mathrm{j}_s}\}^{n+1}, \end{aligned} \tag{4.2}$$

where $\{\mathrm{e}\}$ is a global electric field unknown vector of length N_e, $\{\mathrm{j}_s\}$ represents a current source vector, Δt is the time step, and n means n-th time instant. $\mathbf{D}_\sigma$, and $\mathbf{D}_\epsilon$ are diagonal matrices of conductivity, and permittivity respectively. $\mathbf{S}$ is the discretized $\nabla \times \mu^{-1} \nabla \times$ operator. Unlike an explicit time marching, (4.2) is implicit, and allowing for the use of a large time step independent of space step.

At every time step, the system (4.2) essentially is solving

$$\mathbf{A}\{\mathrm{e}\}^{n+1} = \{b\}, \tag{4.3}$$

where system matrix

$$\mathbf{A} = \mathbf{D}_\epsilon + \Delta t \mathbf{D}_\sigma + \Delta t^2 \mathbf{S} \tag{4.4}$$

and right hand side

$$\{b\} = (2\mathbf{D}_\epsilon + \Delta t \mathbf{D}_\sigma)\{\mathrm{e}\}^n - \mathbf{D}_\epsilon \{\mathrm{e}\}^{n-1} - \Delta t^2 \{\dot{\mathrm{j}_s}\}^{n+1}. \tag{4.5}$$

In a PDE method like the FDTD and the matrix-free time-domain method, the $\mathbf{D}_\epsilon$ and $\mathbf{D}_\sigma$ are diagonal. In an FEM, the two are sparse but not diagonal in general. The $\mathbf{S}$ is a sparse matrix in all PDE methods. In this paper, we use the FDTD as an example to present the proposed DD method, but the general idea is applicable to other PDE methods.

4.3 Proposed Split-Field DD Method

In this section, we will start from a two-domain formulation to help readers understand the key concepts of the proposed method. We then show a general p-domain formulation, analyze the convergence, and compare with traditional DD methods that do not split fields.

4.3.1 *Two*-Domain Problems

Consider two subdomains, let the vector of interface fields be $\{\mathrm{e}\}_s$, we split $\{\mathrm{e}\}_s$ as follows

$$\{\mathrm{e}\}_s = \{\mathrm{e}\}_{s_1} + \{\mathrm{e}\}_{s_2}, \tag{4.6}$$

where $\{\mathrm{e}\}_{s_1}$ and $\{\mathrm{e}\}_{s_2}$ are the two components of $\{\mathrm{e}\}_s$, computed from subdomain 1 and 2 respectively. Using a simple example to illustrate the concept.

We then partition the unknowns to be solved into $\{\mathrm{e}\}_1$, and $\{\mathrm{e}\}_2$ respectively. The $\{\mathrm{e}\}_1$ consists of the unknown fields interior to subdomain 1 and *one component* of the interface field ($\{\mathrm{e}\}_{s_1}$). The $\{\mathrm{e}\}_2$ is composed of unknowns interior to subdomain 2 and *the other component* of the interface field, $\{\mathrm{e}\}_{s_2}$. As can be seen, this partition is very different from existing DD methods. In existing methods, the entire interface field $\{\mathrm{e}\}_s$ is categorized into one subdomain, not split into multiple components. In the proposed new method, the $\{\mathrm{e}\}_1$ and $\{\mathrm{e}\}_2$ overlap at the entries corresponding to the interface unknowns, but each having only one component of the interface field unknown. If augmenting both $\{\mathrm{e}\}_1$ and $\{\mathrm{e}\}_2$ to the full length of the entire unknown field vector $\{\mathrm{e}\}$, we have

$$\{\mathrm{e}\} = \{\mathrm{e}\}_1 + \{\mathrm{e}\}_2. \tag{4.7}$$

For those unknowns interior to each subdomain, they only appear in one $\{\mathrm{e}\}_\mathrm{i}$ vector $(\mathrm{i} = 1, 2)$, and they are zero in other vectors. For those unknowns on the interface, they are superposed from the contribution of each subdomain as shown in (4.6).

The original system equation (4.1) can then be decomposed into the following two subsystems to solve

$$\mathbf{D}_\epsilon \frac{\partial^2 \{\mathrm{e}\}_1}{\partial t^2} + \mathbf{D}_\sigma \frac{\partial \{\mathrm{e}\}_1}{\partial t} + \mathbf{S}_1 \{\mathrm{e}\} = -\frac{\partial \{\mathrm{j}_s\}_1}{\partial t} \tag{4.8}$$

$$\mathbf{D}_\epsilon \frac{\partial^2 \{\mathrm{e}\}_2}{\partial t^2} + \mathbf{D}_\sigma \frac{\partial \{\mathrm{e}\}_2}{\partial t} + \mathbf{S}_2 \{\mathrm{e}\} = -\frac{\partial \{\mathrm{j}_s\}_2}{\partial t}, \tag{4.9}$$

in which $\mathbf{S}_1$ and $\mathbf{S}_2$ are formulated in each subdomain, and

$$\mathbf{S}_1 + \mathbf{S}_2 = \mathbf{S} \tag{4.10}$$

makes the entire $\mathbf{S}$ matrix in the 2-domain problem. All matrices and vectors in the above are written using a global dimension, N_e, which is the total number of electric field unknowns in the computational domain. Adding up (4.8) and (4.9), it can be readily verified that (4.7) is the solution of the original equation (4.1).

Equation (4.8) and (4.9) can further be written as

$$\mathbf{D}_\epsilon \frac{\partial^2 \{\mathrm{e}\}_1}{\partial t^2} + \mathbf{D}_\sigma \frac{\partial \{\mathrm{e}\}_1}{\partial t} + \mathbf{S}_1 (\{\mathrm{e}\}_1 + \{\mathrm{e}\}_2) = -\frac{\partial \{\mathrm{j}_s\}_1}{\partial t} \tag{4.11}$$

$$\mathbf{D}_\epsilon \frac{\partial^2 \{\mathrm{e}\}_2}{\partial t^2} + \mathbf{D}_\sigma \frac{\partial \{\mathrm{e}\}_2}{\partial t} + \mathbf{S}_2 (\{\mathrm{e}\}_1 + \{\mathrm{e}\}_2) = -\frac{\partial \{\mathrm{j}_s\}_2}{\partial t}, \tag{4.12}$$

whose implicit time marching can be performed as the following

$$\begin{bmatrix} \mathbf{D}_\epsilon + \Delta t \mathbf{D}_\sigma + \Delta t^2 \mathbf{S}_1 & \Delta t^2 \mathbf{S}_1 \\ \Delta t^2 \mathbf{S}_2 & \mathbf{D}_\epsilon + \Delta t \mathbf{D}_\sigma + \Delta t^2 \mathbf{S}_2 \end{bmatrix} \begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \{\mathrm{e}\}_2^{n+1} \end{bmatrix} = \begin{bmatrix} \{b\}_1 \\ \{b\}_2 \end{bmatrix} \tag{4.13}$$

with local right hand side

$$\{b\}_\mathrm{i} = (2\mathbf{D}_\epsilon + \Delta t \mathbf{D}_\sigma) \{\mathrm{e}\}_\mathrm{i}^n - \mathbf{D}_\epsilon \{\mathrm{e}\}_\mathrm{i}^{n-1} - \Delta t^2 \{\dot{\mathrm{j}_s}\}_\mathrm{i}^{n+1}. \tag{4.14}$$

The above (4.13) can be decoupled into each subdomain, and solved iteratively as follows until convergence.

$$\begin{aligned}&\begin{bmatrix} \mathbf{D}_1+\Delta t^2\mathbf{S}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2+\Delta t^2\mathbf{S}_2 \end{bmatrix} \begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \{\mathrm{e}\}_2^{n+1} \end{bmatrix}^{(k+1)} \\ &= -\Delta t^2 \begin{bmatrix} \mathbf{0} & \mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \{\mathrm{e}\}_2^{n+1} \end{bmatrix}^{(k)} + \begin{bmatrix} \{b\}_1 \\ \{b\}_2 \end{bmatrix},\end{aligned} \tag{4.15}$$

in which k denotes the iteration number, and

$$\mathbf{D}_{\mathrm{i}} = \mathbf{D}_{\epsilon} + \Delta t \mathbf{D}_{\sigma}. \tag{4.16}$$

Since the off-diagonal part of the system matrix in (4.13) is one part of the diagonal part, the convergence of the above iteration is guaranteed, and the number of iterations is also small. The proof of the convergence is given in the following Section 4.3.4.

In the formulations developed above, to ease a mathematical understanding, all matrices and vectors are written using a global dimension, N_{e}. In real implementation, (4.15) can be solved in each subdomain using unknowns residing only in one subdomain. This is because $\{\mathrm{e}\}_{\mathrm{i}}$ only contains the field unknowns in one subdomain. The matrices acting on $\{\mathrm{e}\}_{\mathrm{i}}$ are hence only effective in the part corresponding to $\{\mathrm{e}\}_{\mathrm{i}}$. The number of nonzero elements in $\{\mathrm{e}\}_{\mathrm{i}}$ is only $N_{\mathrm{e,i}}$, i.e., the number of electric field unknowns belonging to subdomain i.

4.3.2 P-Domain Problems

Extending (4.8) and (4.9) to p subdomains, we obtain

$$\begin{bmatrix} \mathbf{D}_{\epsilon}\frac{\partial^2\{\mathrm{e}\}_1}{\partial t^2} \\ \vdots \\ \mathbf{D}_{\epsilon}\frac{\partial^2\{\mathrm{e}\}_p}{\partial t^2} \end{bmatrix} + \begin{bmatrix} \mathbf{D}_{\sigma}\frac{\partial\{\mathrm{e}\}_1}{\partial t} \\ \vdots \\ \mathbf{D}_{\sigma}\frac{\partial\{\mathrm{e}\}_p}{\partial t} \end{bmatrix} + \begin{bmatrix} \mathbf{S}_1 & \cdots & \mathbf{S}_1 \\ \vdots & \ddots & \vdots \\ \mathbf{S}_p & \cdots & \mathbf{S}_p \end{bmatrix} \begin{bmatrix} \{\mathrm{e}\}_1 \\ \vdots \\ \{\mathrm{e}\}_p \end{bmatrix} = \begin{bmatrix} -\frac{\partial\{\mathrm{j}_s\}_1}{\partial t} \\ \vdots \\ -\frac{\partial\{\mathrm{j}_s\}_p}{\partial t} \end{bmatrix}, \tag{4.17}$$

and the entire domain solution $\{\mathrm{e}\}$ can be obtained from the $\{\mathrm{e}\}_\mathrm{i}$ of each subdomain as

$$\{\mathrm{e}\} = \sum_{\mathrm{i}=1}^{p} \{\mathrm{e}\}_\mathrm{i}. \tag{4.18}$$

Here, $\{\mathrm{e}\}_\mathrm{i}$ is the contribution to solution vector $\{\mathrm{e}\}$ from the i-th subdomain and it is only nonzero at $N_{\mathrm{e_i}}$ edges (unknowns) residing in subdomain i. For an interface field unknown shared by m subdomains, the $\{\mathrm{e}\}_\mathrm{i}$ vector in each of the m subdomains has one component of the interface field unknown.

Performing a backward difference on (4.17), we obtain

$$\begin{bmatrix} \mathbf{D}_1 & & \\ & \ddots & \\ & & \mathbf{D}_p \end{bmatrix} \begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \vdots \\ \{\mathrm{e}\}_p^{n+1} \end{bmatrix} + \Delta t^2 \begin{bmatrix} \mathbf{S}_1 & \cdots & \mathbf{S}_1 \\ \vdots & \ddots & \vdots \\ \mathbf{S}_p & \cdots & \mathbf{S}_p \end{bmatrix} \begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \vdots \\ \{\mathrm{e}\}_p^{n+1} \end{bmatrix} = \begin{bmatrix} \{b\}_1 \\ \vdots \\ \{b\}_p \end{bmatrix}, \tag{4.19}$$

where

$$\{b\}_\mathrm{i} = (2\mathbf{D}_{\epsilon_\mathrm{i}} + \Delta t \mathbf{D}_{\sigma_\mathrm{i}})\{\mathrm{e}\}_\mathrm{i}^n - \mathbf{D}_{\epsilon_\mathrm{i}}\{\mathrm{e}\}_\mathrm{i}^{n-1} - \Delta t^2 \{\dot{\mathrm{j}_s}\}_\mathrm{i}^{n+1}. \tag{4.20}$$

Adding the rows of equations in (4.19), it can be seen clearly that it reverts to the original equation shown in (4.2), and

$$\{b\} = \sum_{\mathrm{i}=1}^{p} \{b\}_\mathrm{i}. \tag{4.21}$$

The second term in (4.19) can be further separated into two parts: The diagonal part that represents the curl-curl operation performed on the unknowns of each subdomain including interior unknowns and one component of the interface field unknown; and the off-diagonal part that represents the curl-curl operation performed on the rest of the components of the interface field contributed by the neighbor subdomains. Moving the off-diagonal part to the right hand side, (4.19) can be solved in a decoupled manner by performing the following iteration until it converges

$$\begin{bmatrix} \mathbf{D}_1 + \Delta t^2 \mathbf{S}_1 & & \\ & \ddots & \\ & & \mathbf{D}_p + \Delta t^2 \mathbf{S}_p \end{bmatrix} \begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \vdots \\ \{\mathrm{e}\}_p^{n+1} \end{bmatrix}^{(k+1)} = \\ - \Delta t^2 \begin{bmatrix} \mathbf{0} & \mathbf{S}_1 & \cdots & \mathbf{S}_1 \\ \mathbf{S}_2 & \mathbf{0} & \cdots & \mathbf{S}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_p & \mathbf{S}_p & \cdots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \vdots \\ \{\mathrm{e}\}_p^{n+1} \end{bmatrix}^{(k)} + \begin{bmatrix} \{b\}_1 \\ \vdots \\ \{b\}_p \end{bmatrix}, \tag{4.22}$$

in which k denotes the iteration index. Obviously, when it converges,

$$\begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \vdots \\ \{\mathrm{e}\}_p^{n+1} \end{bmatrix}^{(k+1)} = \begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \vdots \\ \{\mathrm{e}\}_p^{n+1} \end{bmatrix}^{(k)}, \tag{4.23}$$

we obtain the solution of (4.19). This iteration (index k) has a convergence rate dictated by the spectral radius of $(\mathbf{D}_\mathrm{i} + \Delta t^2 \mathbf{S}_\mathrm{i})^{-1}(\Delta t^2 \mathbf{S}_\mathrm{i})$ which is bounded by 1, and hence the convergence is guaranteed. A more detailed proof is provided in Section 4.3.4.

The implementation of (4.22) can be done domain by domain, where in each domain, the dimension of matrices and vectors involved in the computation is only the number of unknowns in a single domain. The curl-curl operator based coupling in the right hand side can be computed as a single vector in each domain as

$$\{b_s\}_\mathrm{i}^{n+1} = \Delta t^2 \mathbf{S}_\mathrm{i} \sum_{\mathrm{j}=1,\mathrm{j}\neq\mathrm{i}}^{p} \{\mathrm{e}\}_\mathrm{j}^{n+1}. \tag{4.24}$$

At each time instant $(n+1)$, the $\{b_s\}_\mathrm{i}^{n+1}$ term can be first generated from $\{\mathrm{e}\}^n$ since $\{b_s\}_\mathrm{i}^n = \Delta t^2 \mathbf{S}_\mathrm{i} \sum_{\mathrm{j}=1,\mathrm{j}\neq\mathrm{i}}^{p} \{\mathrm{e}\}_\mathrm{j}^n$. This is known, and hence can be moved to the right hand side, and used as the solution of the $k = 0$ step of (4.22). The iteration then continues until the convergence is reached.

4.3.3 Matrix Partition

One major part of the proposed split-field DD method is how to partition the system matrix into

$$\mathbf{S} = \sum_{\mathrm{i}=1}^{p} \mathbf{S}_{\mathrm{i}}. \tag{4.25}$$

In FDTD, a conventional difference based formula does not reveal the composition of $\mathbf{S}$ clearly. We can use the patch-based single-grid FDTD formulation [29] to partition $\mathbf{S}$ easily. Based on this formulation, each patch j contributes to $\mathbf{S}$ in the format of a rank-1 matrix $\left(\mathbf{S}_h^{(\mathrm{j})}\right)_{N_\mathrm{e}\times 1} \mu_\mathrm{j}^{-1} \left(\mathbf{S}_\mathrm{e}^{(\mathrm{j})}\right)_{1\times N_\mathrm{e}}$. Using this, we can partition the $\mathbf{S}$ matrix based on patches in each subdomain. Namely, assemble

$$\mathbf{S}_\mathrm{i} = \sum_{\mathrm{j}=1}^{N_{h_\mathrm{i}}} \mu_\mathrm{j}^{-1} \left(\mathbf{S}_h^{(\mathrm{j})}\right)_{N_{\mathrm{e_i}}\times 1} \left(\mathbf{S}_\mathrm{e}^{(\mathrm{j})}\right)_{1\times N_{\mathrm{e_i}}} \tag{4.26}$$

over all N_{h_i} patches in i-th subdomain. The assembly of $\mathbf{S}_\mathrm{i}$ from patches applies to subdomains with regular shape, irregular shape, and even discontinuous structure.

In the FEM, the $\mathbf{S}$ is a summation of elemental contributions. Hence, we can partition $\mathbf{S}$ into the sum of $\mathbf{S}$ matrices formulated in each element in each subdomain. Since the $\mathbf{D}_{eps}$ and $\mathbf{D}_{sig}$ are not diagonal in a conventional FEM, they can be partitioned in the same way as how $\mathbf{S}$ is partitioned. Other PDE methods follow the same rule.

4.3.4 Convergence Analysis of the Proposed DD Method

Using a 2-domain as an example, the iteration of the proposed algorithm is performed on (4.15) until convergence. When convergence reaches, the following is satisfied

$$\begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \{\mathrm{e}\}_2^{n+1} \end{bmatrix}^{(k+1)} = \begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \{\mathrm{e}\}_2^{n+1} \end{bmatrix}^{(k)}, \tag{4.27}$$

and hence the first term in the right hand side can be moved to the left hand side to be combined with the **S**-based term, yielding the same solution as the original equation. Equation (4.15) can further be written as

$$\begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \{\mathrm{e}\}_2^{n+1} \end{bmatrix}^{(k+1)} = - \begin{bmatrix} \mathbf{0} & \mathbf{M}_1 \\ \mathbf{M}_2 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \{\mathrm{e}\}_1^{n+1} \\ \{\mathrm{e}\}_2^{n+1} \end{bmatrix}^{(k)} + \begin{bmatrix} \{b\}_1 \\ \{b\}_2 \end{bmatrix} \tag{4.28}$$

where

$$\mathbf{M}_\mathrm{i} = (\mathbf{D}_\mathrm{i} + \Delta t^2 \mathbf{S}_\mathrm{i})^{-1} (\Delta t^2 \mathbf{S}_\mathrm{i}) \quad (\mathrm{i} = 1, 2) \tag{4.29}$$

Hence, the convergence of the iteration is determined by the spectral radius of the following matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{0} & \mathbf{M}_1 \\ \mathbf{M}_2 & \mathbf{0} \end{bmatrix}, \tag{4.30}$$

which is

$$\rho(\mathbf{M}) = \sqrt{\rho(\mathbf{M}_2 \mathbf{M}_1)}. \tag{4.31}$$

Since $\mathbf{M}_\mathrm{i}$ has a form shown in (4.29), the following always holds true

$$\rho(\mathbf{M}_\mathrm{i}) < 1, \quad \forall \ \mathrm{i} \tag{4.32}$$

As a result, we have

$$\rho(\mathbf{M}) < 1, \tag{4.33}$$

and hence the convergence of (4.15) is guaranteed.

The above proof is developed for a 2-domain system. But the same proof is readily applicable to an arbitrary p-domain system by applying the proof recursively. For example, between the two domains in the above, one domain again can be partitioned into two subdomains. The resultant amplification matrix is the same as (4.30), but with $\mathbf{M}_2$ now having the same two-domain matrix structure as (4.30). Again, $\mathbf{M}_2$ would have a less than 1 spectral radius since (4.32) holds true in every domain, and hence (4.33) is satisfied for a 3-domain problem. Doing so recursively, one can obtain (4.33) for an arbitrary p-domain and general 3-D problem.

4.3.5 Comparison with Non-split-field DD Algorithms

In a prevailing non-split-field DD parallel algorithm [19], [48], [49], a computational domain is partitioned into unknowns interior to each subdomain $\{\mathrm{e}\}_{\mathrm{i}}^{in}$, and unknowns residing on the interface between subdomains $\{\mathrm{e}\}_{\Gamma}$, obtaining a partitioned system

$$\begin{bmatrix} \mathbf{A}_{11} & & & \mathbf{A}_{1\Gamma} \\ & \ddots & & \vdots \\ & & \mathbf{A}_{pp} & \mathbf{A}_{p\Gamma} \\ \mathbf{A}_{\Gamma 1} & \cdots & \mathbf{A}_{\Gamma p} & \mathbf{A}_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \{\mathrm{e}\}_1^{in} \\ \vdots \\ \{\mathrm{e}\}_p^{in} \\ \{\mathrm{e}\}_\Gamma \end{bmatrix} = \begin{bmatrix} \{b\}_1^{in} \\ \vdots \\ \{b\}_p^{in} \\ \{b\}_\Gamma \end{bmatrix}, \tag{4.34}$$

where $\mathbf{A}_{\mathrm{ii}}$, $\mathbf{A}_{\Gamma\Gamma}$, $\mathbf{A}_{\mathrm{i}\Gamma}$, and $\mathbf{A}_{\Gamma\mathrm{i}}$ represent the interior to interior coupling, the interface to interface coupling, the interior to interface coupling, and the interface to interior coupling, respectively. Unlike the proposed method, the off-diagonal part of the above system matrix has no relationship with the diagonal part, since they are generated for different sets of unknowns. If the off-diagonal part of the above system matrix is moved to the right hand side, the convergence is not guaranteed, and it is problem dependent.

One can also solve the (4.34) directly by first building a Schur complement

$$\mathbf{A}_{Sch} = \mathbf{A}_{\Gamma\Gamma} - \sum_{\mathrm{i}=1}^{p} \mathbf{A}_{\Gamma\mathrm{i}} \mathbf{A}_{\mathrm{ii}}^{-1} \mathbf{A}_{\mathrm{i}\Gamma}, \tag{4.35}$$

which is then used to solve unknowns at all interface Γ

$$\{\mathrm{e}\}_\Gamma = \mathbf{A}_{Sch}^{-1} \left(\{b\}_\Gamma - \sum_{\mathrm{i}=1}^{p} \mathbf{A}_{\Gamma\mathrm{i}} \mathbf{A}_{\mathrm{ii}}^{-1} \{b\}_{\mathrm{i}} \right). \tag{4.36}$$

Knowing $\{\mathrm{e}\}_\Gamma$, $\{\mathrm{e}\}_\mathrm{i}$ at subdomain i can be solved in a decoupled manner as

$$\{\mathrm{e}\}_\mathrm{i} = \mathbf{A}_\mathrm{ii}^{-1} \left(\{b\}_\mathrm{i} - \mathbf{A}_{\mathrm{i}\Gamma}\{\mathrm{e}\}_\Gamma\right). \tag{4.37}$$

The computational bottleneck of this approach is the direct solution of the interface problems, i.e., $\mathbf{A}_{Sch}$, when the interface problem is large.

In contrast, the proposed split-field DD algorithm possesses the advantages of both direct and iterative solutions. The direct solver only needs to handle a small sparse matrix in each subdomain, while the iterative solver that solves the global coupling has a guaranteed fast convergence.

4.4 Split-Field DD Parallel Algorithm

The DD algorithm presented in previous section can be readily parallelized, owing to its domain-decomposed formulation. In this section, we elaborate how to assess the error and control the accuracy when the proposed DD algorithm is run in parallel.

To assess the error, a straightforward approach is to compute relative residual at each iteration as follows.

$$relaResb = \frac{\|\mathbf{A}\{\mathrm{e}\}^{k,n+1} - \{b\}\|}{\|\{b\}\|}. \tag{4.38}$$

However, gathering a large vector from every subdomain at every iteration is too expensive in parallel computing.

Here, we develop a much more efficient method with little communication cost to compute the relative residual of the entire system. Instead of using the right hand side of (4.3) directly, we use the right hand side vector in (4.19), and thus

$$\{\tilde{b}\} = \begin{bmatrix} \{b\}_1 \\ \vdots \\ \{b\}_p \end{bmatrix}. \tag{4.39}$$

The relative residual of (4.19) with respect to $\{\tilde{b}\}$ is a good characterization of accuracy of the entire system solution. What's more, $relaRes\tilde{b}$ can be computed in parallel by using the field unknowns local to each subdomain. We can rewrite (4.19) as

$$\begin{bmatrix} \mathbf{D}_1\{\mathrm{e}\}_1^{n+1} + \Delta t^2 \mathbf{S}_1 \{\mathrm{e}\}^{n+1} \\ \vdots \\ \mathbf{D}_p\{\mathrm{e}\}_p^{n+1} + \Delta t^2 \mathbf{S}_p \{\mathrm{e}\}^{n+1} \end{bmatrix} = \begin{bmatrix} \{b\}_1 \\ \vdots \\ \{b\}_p \end{bmatrix}, \tag{4.40}$$

which can be written in short as

$$f\left(\{\mathrm{e}\}^{n+1}\right) = \{\tilde{b}\}. \tag{4.41}$$

Then, for the solution $\{\mathrm{e}\}^{k,n+1}$ obtained at the k-th iteration, we can evaluate the relative residual as

$$relaRes\tilde{b} = \frac{\|f\left(\{\mathrm{e}\}^{k,n+1}\right) - \{\tilde{b}\}\|}{\|\{\tilde{b}\}\|}. \tag{4.42}$$

The above is nothing but

$$relaRes\tilde{b} = \frac{\|\{\tilde{b}_s\}^{k,n+1} - \{\tilde{b}_s\}^{k-1,n+1}\|}{\|\{\tilde{b}\}\|}, \tag{4.43}$$

where $\{\tilde{b}_s\}^{k,n+1}$ is the union of $\{b_s\}_\mathrm{i}^{k,n+1}$ in each subdomain, which is shown in (4.24). This can be derived as follows:

$$\begin{aligned} &\left\{f\left(\{\mathrm{e}\}^{k,n+1}\right) - \{\tilde{b}\}\right\}_{\text{at subdomain i}} \\ &= \mathbf{D}_\mathrm{i}\{\mathrm{e}\}_\mathrm{i}^{k,n+1} + \Delta t^2 \mathbf{S}_\mathrm{i}\{\mathrm{e}\}^{k,n+1} - \{b\}_\mathrm{i} \\ &= (\mathbf{D}_\mathrm{i} + \Delta t^2 \mathbf{S}_\mathrm{i})\{\mathrm{e}\}_\mathrm{i}^{k,n+1} + \{b_s\}_\mathrm{i}^{k,n+1} - \{b\}_\mathrm{i} \\ &= \{b_s\}_\mathrm{i}^{k,n+1} - \{b_s\}_\mathrm{i}^{k-1,n+1}. \end{aligned} \tag{4.44}$$

Under Euclidean norm,

$$\|\{\tilde{b}\}\| = \sqrt{\sum_{\mathrm{i}=1}^{p} \|\{b\}_\mathrm{i}\|^2}. \tag{4.45}$$

The other norm $\|\{\tilde{b}_s\}^{k,n+1} - \{\tilde{b}_s\}^{k-1,n+1}\|$ in numerator can be calculated similarly. Therefore,

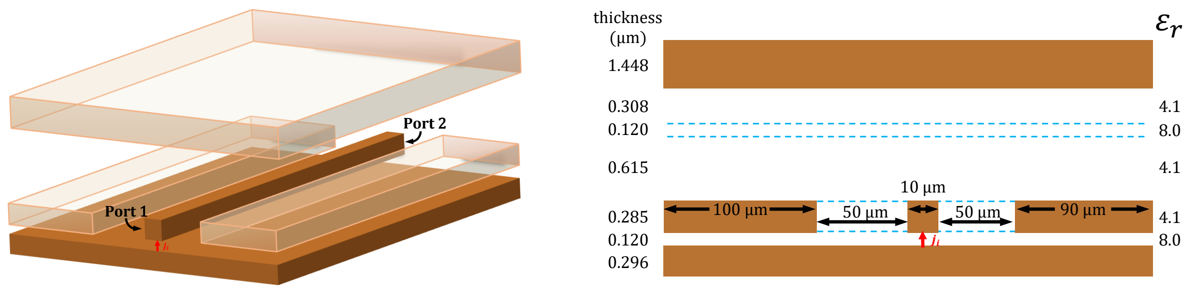
$$relaRes\tilde{b} = \frac{\sqrt{\sum_{\mathrm{i}=1}^{p} \|\{b_s\}_{\mathrm{i}}^{k,n+1} - \{b_s\}_{\mathrm{i}}^{k-1,n+1}\|^2}}{\sqrt{\sum_{\mathrm{i}=1}^{p} \|\{b\}_{\mathrm{i}}\|^2}}. \tag{4.46}$$

Instead of gathering the entire vector from every subdomain, using (4.46), we now can solve $relaRes\tilde{b}$ by only gathering 2 numbers $\|\{b\}_{\mathrm{i}}\|$ and $\|\{b_s\}_{\mathrm{i}}^{k,n+1} - \{b_s\}_{\mathrm{i}}^{k-1,n+1}\|$ from each subdomain. Both $\{b\}_{\mathrm{i}}$ and $\{b_s\}_{\mathrm{i}}^{k,n+1}$ are already calculated in the parallel iterative solver, so, we can directly reuse the vectors to generate their local norms. Therefore, the calculation of relative residual $relaRes\tilde{b}$ using (4.46) has very little computing cost and communication cost.

4.5 Numerical Results

In this section, we use numerical examples to examine the accuracy, convergence, runtime complexity, and memory complexity of the proposed DD solver. Both IC chips and packages are simulated.

4.5.1 Test-chip Interconnect



(a) (b)

Figure 4.1. Geometry of a test-chip interconnect. (a) 3-D view of three metal layers, where the current source is supplied from bottom metal layer to the center wire at port 1. (b) Front view of the test-chip interconnect.

We first validate the accuracy of the proposed method by simulating a realistic test-chip interconnect structure [37], which is fabricated using a silicon processing technology. This

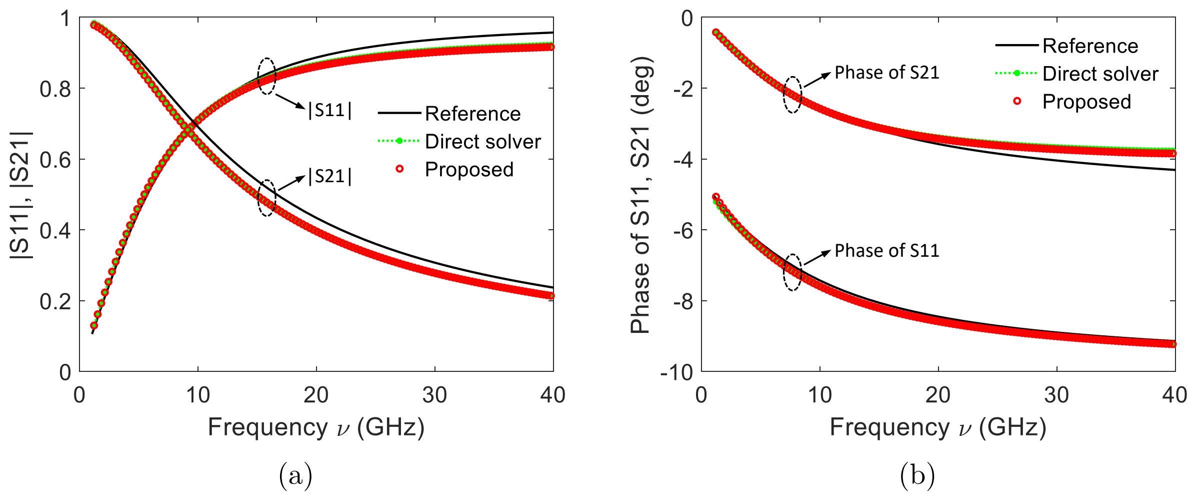


Figure 4.2. Simulated S-parameters of the test-chip interconnect. (a) Magnitude of S_{11} and S_{21}. (b) Phase of S_{11} and S_{21}.

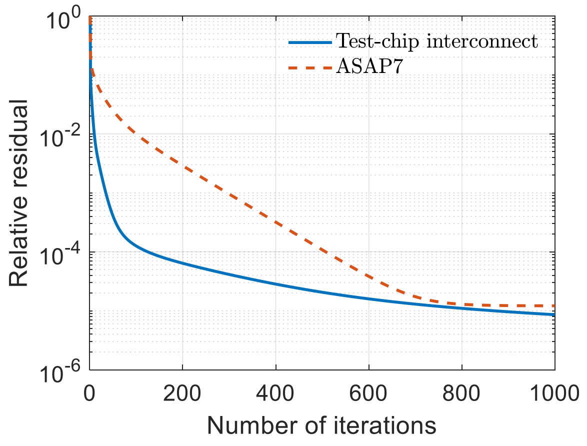


Figure 4.3. Relative residual of the test-chip interconnect.

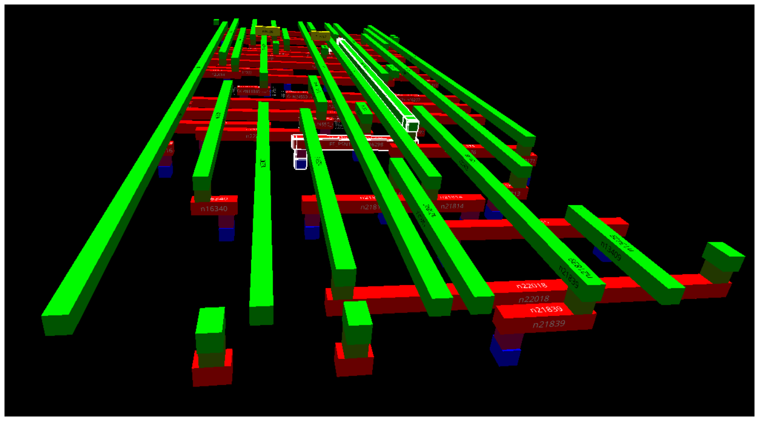
100 μm-long test-chip interconnect comprises 3 metal layers and 5 inhomogeneous dielectric stacks, whose 3-D structure and cross-sectional view are illustrated in Fig. 4.1. Fig. 4.1 also shows all geometrical dimensions and the relative permittivity ϵ_r of each layer. The 100 μm-long interconnect is sandwiched between two 30 μm-long air layers in the front and at the back. A non-uniform mesh is used. Along stack growth direction z, 7 stacks are discretized into mesh sizes $N_z = [3, 3, 7, 3, 2, 2, 3]$ from bottom to top. The 100 μm length along y is discretized into $N_y = 20$, and each 30 μm air padding is discretized into $N_y = 6$. The center interconnect is discretized into $N_x = 10$ along x direction. In the proposed DD solver, the

entire structure is equally partitioned into 4 subdomains along y direction. The convergence criterion of the DD solver is set as $tol = 5 \times 10^{-3}$ and maximum iteration number $maxit = 30$. A current source of a time derivative Gaussian pulse $\mathrm{j}_s = -(t - t_0)\exp[-(\frac{t-t_0}{\tau_s})^2]$ A/m^2 ($t_0 = 4\tau_s$, $\tau_s = 2 \times 10^{-11}$ s) is placed right in the middle at the near-end of the center interconnect. A time window of $8\tau_s$ with $200dt$ is simulated. After performing a Fast Fourier Transform (FFT) on the current source and the simulated time-domain port voltages, we directly obtain the Z-parameters of the structure, which are then converted to S-parameters with a 50 Ω reference impedance. The reference S-parameters of this test-chip interconnect in Fig. 4.2 are from [37], which have been validated by experimental measurements. The simulated S-parameters from the proposed DD solver, as shown in Fig. 4.2, agree very well with both those from a direct solver and reference results.

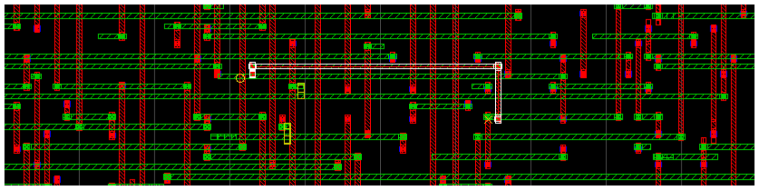
To further determine how fast the DD solver converges on this test-chip interconnect in general, we set the entire right hand side vector $\{\tilde{b}\}$ to be a random vector between 0 and 1. Then, we gather the solution $\{\mathrm{e}\}^{k+1}$ at $k+1$-th iteration from all 4 subdomains and calculate the $relaResx = \frac{\|\{\mathrm{e}\}^{k+1} - \{\mathrm{e}\}^{k}\|}{\|\{\mathrm{e}\}^{k+1}\|}$. As seen in Fig. 4.3, after 30 iterations, the $relaResx$ can quickly converge to 1.3×10^{-3}.

4.5.2 ASAP7 On-chip Interconnect

Here, we test the performance of our DD solver on an IC layout at a 7 nm technology node. Due to the extremely fine feature size thereby ill-conditioned system matrix, iterative solvers typically suffer from slow convergence on on-chip structures fabricated with 7 nm technology node. From the OpenROAD project [56], we obtain an chip design, which is designed with ASAP7 7 nm FinFET process design kit (PDK) [57]. We simulate one net from the ASAP7 chip, where the net is highlighted in white edges in Fig. 4.4, occupying a 2.268 μm $\times$ 10.376 μm chip area. This area is discretized into a 79×131 mesh. The structure here includes the first 3 metal layers M1-M3 and first 3 via layers V0-V2 of the ASAP7 process. The thickness is 36 nm for each metal layer and 39.6 nm for each via layer. The layer stack direction is discretized into a $N_z = 6$ cells. The entire structure is shown in Fig. 4.4 with PEC at the bottom and PMC at the other 5 sides. Same as previous test-chip



(a)



(b)

Figure 4.4. Geometry of ASAP7 on-chip interconnect. (a) 3-D view. (b) Top view.

interconnect, in the proposed DD solver, the entire structure is equally partitioned into 4 subdomains along y direction. The convergence criterion of the DD solver is set to be $tol = 5 \times 10^{-3}$ and with a larger $maxit = 600$. Still a Gaussian derivative pulse is used for the source current, thus $\mathrm{j}_s = -(t - t_0)\exp[-(\frac{t-t_0}{\tau_s})^2]$ A/m^2. A slightly smaller $\tau_s = 1 \times 10^{-11}$ s is used corresponding to the maximum signal frequency $\sim 1/\tau_s = 100$ GHz, and $t_0 = 4\tau_s$. Still, a time window of $8\tau_s$ with $200dt$ is simulated. The simulated Z-parameters from the proposed DD solver, as shown in Fig. 4.5, agree very well with a brute-forth solution.

The relative residual of this ASAP7 interconnect is generated and plotted in Fig.4.3. The minimum fine feature size is 120 nm for previous test-chip interconnect and 36 nm for this ASAP7 interconnect. Besides, the fine feature size of each layer in this ASAP7 example is much smaller than that in the test-chip interconnect. The convergence of the proposed DD solver is slower for this ASAP7 interconnect, as can be seen from Fig.4.3. This agrees with our theoretical prediction since the curl-curl term has an enlarged norm.

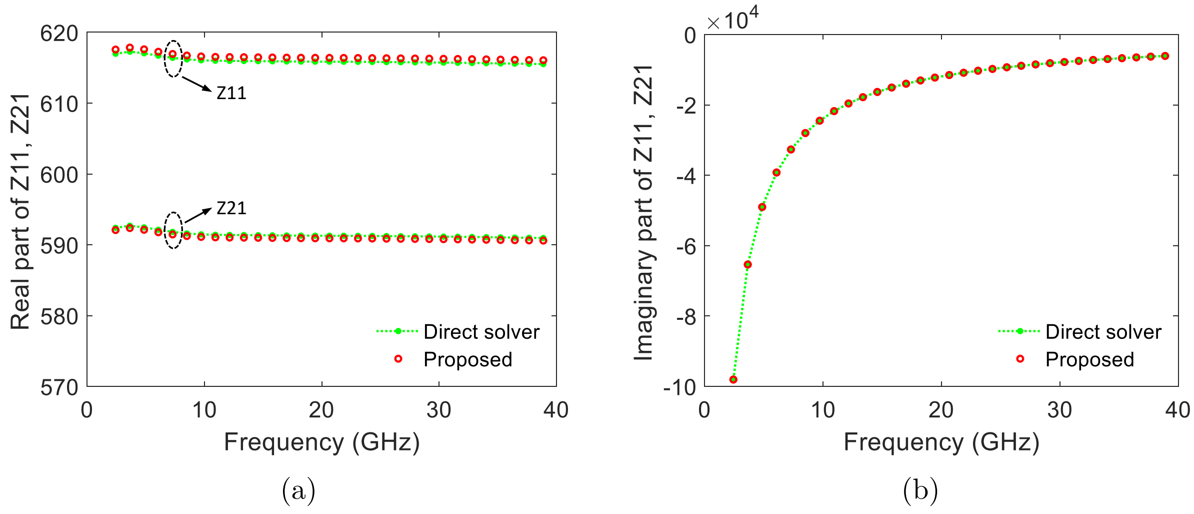
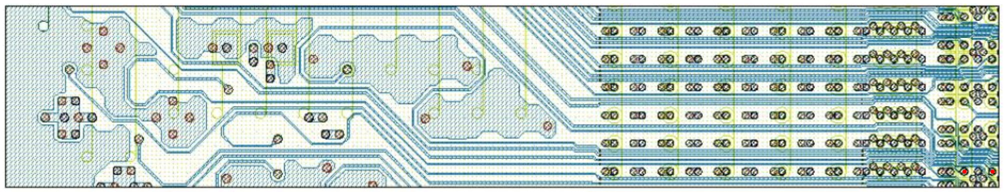


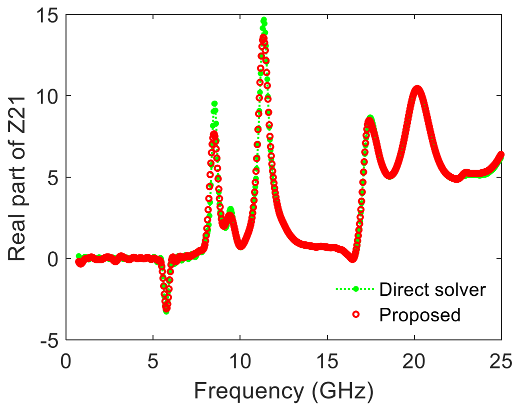
Figure 4.5. Simulated Z-parameters of the ASAP7 on-chip interconnect. (a) Real part of Z_{11} and Z_{21}. (b) Imaginary part of Z_{11} and Z_{21}.

4.5.3 A Representative Package Structure

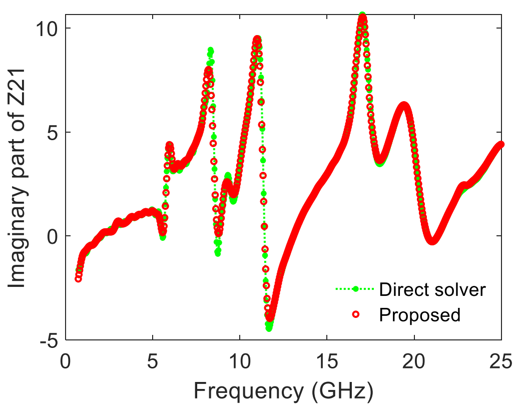
Compared to previous on-chip examples, packages typically have a larger electrical size, therefore it can have a better convergence behavior. Here, we further validate the accuracy and convergence of the proposed DD solver on package problems. A representative package structure as shown in Fig. 4.6 (a) is simulated, where two ports are marked by the red dots in the lower right corner. The entire structure is discretized into $709,339$ unknowns. The proposed DD solver partitions the structure into 20 subdomains. The same current source j_s is used as in the previous test-chip interconnect example. Due to many resonance modes in this structure, we simulate a very long time window of $128\tau_s$ with $3200dt$, so that the electromagnetic response can die down to almost 0. In the proposed time-domain iterative solver, the error at all previous time steps will accumulate over time marching. To guarantee the accuracy at later time steps for such a long time window, we use more a stringent *tol* of 1×10^{-3} and $maxit = 100$ for the DD solver. The result shows that in most time steps, the solver can converge to $relaRes\tilde{b} < 5 \times 10^{-3}$ within 30 iterations and $relaRes\tilde{b} < 1 \times 10^{-3}$ within 70 iterations. The extracted Z-parameters in Fig. 4.6 show a good match over a broad frequency band between proposed DD solver and a direct solver.



(a)



(b)



(c)

Figure 4.6. A representative package structure and its simulated impedance parameter Z_{21}. (a) Top view. (b) Real part of Z_{21}. (c) Imaginary part of Z_{21}.

Next, we numerically examine the convergence of the proposed DD method from the perspective of eigenvalue spectrum. A slightly coarser mesh is used so that an eigenvalue solution is feasible. We check the spectral radius of the entire amplification matrix which is of size 0.19 million. All 0.19 M eigenvalues are solved and the largest 17 K of them are plotted in Fig. 4.7. The spectral radius is shown to be 0.9949, and more importantly only 3.0% of all eigenvalues have its absolute value greater than 0.5. The eigenvalues close to 1 have little contributions in the field solution since they correspond to the largest eigenvalues of the $\mathbf{S}$ matrix. The field solution are dominated by the nullspace and a few small eigenmodes of $\mathbf{S}$. Therefore, even though those closest to 1 eigenvalues are the slowest to converge, since they are not important to the solution, we don't need to wait for them to converge.

4.5.4 2-D Partition of the DD Solver

Here, we demonstrate a 2D partition of the DD solver. Use the same package structure in Fig. 4.6 (a) and keep the same mesh. Still, use a Gaussian derivative source current

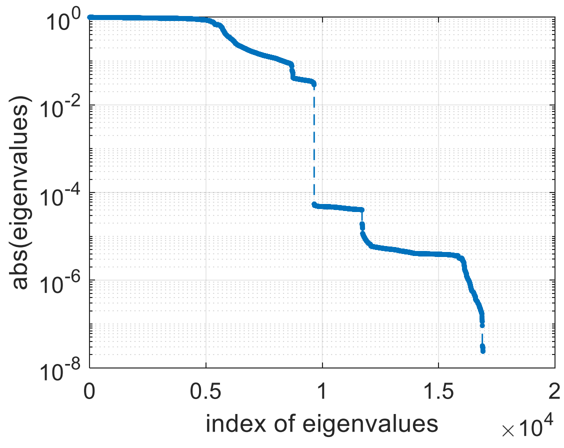


Figure 4.7. Eigenvalue spectrum of the representative package structure.

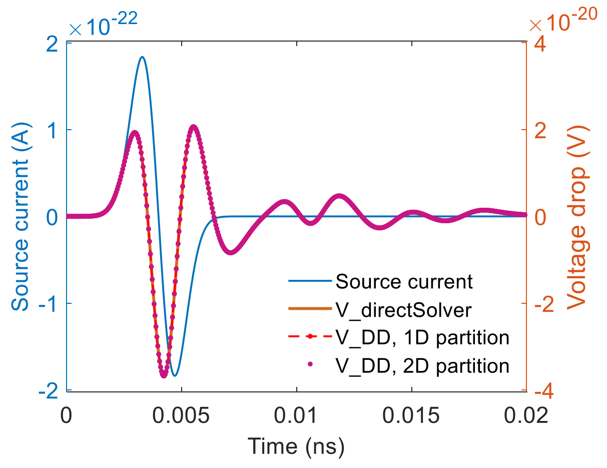


Figure 4.8. Voltage responses of the representative package structure when solved by DD solver with 2-D partition.

$j_s = -(t - t_0)\exp[-(\frac{t-t_0}{\tau_s})^2]$ A/m^2. This time, use a smaller $\tau_s = 1 \times 10^{-12}$ s. $t_0 = 4\tau_s$. A time window of $20\tau_s$ with $500dt$ is simulated. The DD solver uses $tol = 1 \times 10^{-5}$ and $maxit$ = 30. The 1-D partition of the DD solver partitions the entire structure into 15 subdomains along y direction, whereas the 2-D partition of the DD solver partitions the entire structure into 5 subdomains along y direction and 3 subdomains along x direction. The same number of subdomains for both 1-D partition and 2-D partition. All time steps of 2-D partition can converge to $relaRes\tilde{b} < 1 \times 10^{-5}$ within 5 iterations, whereas all time steps of 1-D partition can converge to $relaRes\tilde{b} < 1 \times 10^{-5}$ within 4 iterations. Given the same setup and number of computing cores, compared to 1-D partition, 2-D partition converges slightly worse, but requires less data communication, and can better accommodate the subdomains with the structure of the package. Their voltage responses are plotted in Fig. 4.8, where DD solver with 2-D partition matches very well with both 1-D partition and direct solver.

4.5.5 Large-scale IBM Plasma Package Structure

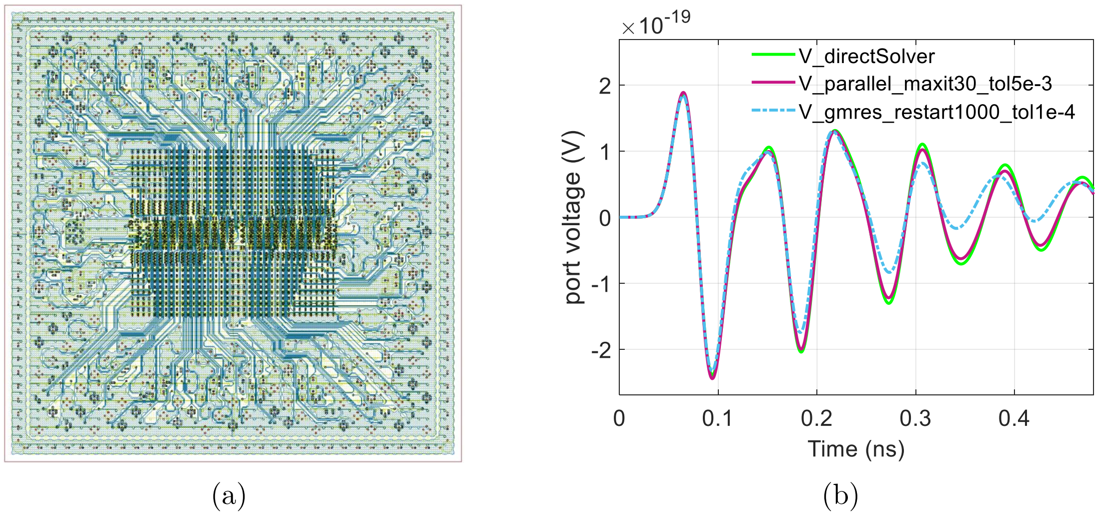


Figure 4.9. IBM plasma package example. (a) Top view of the package structure. (b) Voltage simulated in time domain.

We apply the proposed method to simulate the plasma package from IBM. The top view of the structure is shown in Fig. 4.9 (a), which is discretized into $8,580,128$ unknowns. A Gaussian derivative source current is injected in the middle of the package on the top layer, whose waveform is $\mathrm{j}_s = -(t-t_0)\exp[-(\frac{t-t_0}{\tau_s})^2]$ A/m^2, where $t_0 = 4\tau_s$ and $\tau_s = 2\times10^{-11}$s. The pulse has a maximal signal frequency of approximately 50 GHz. A time window of $24\tau_s$ with $600dt$ is simulated. The voltage simulated from this method is plotted in Fig. 4.9 and compared with a conventional direct solver, and a GMRES-based iterative solver. The proposed solver converges to $relaRes\tilde{b} < 5\times10^{-3}$ within 20 iterations at each time instant. The GMRES uses tol $= 1\times10^{-4}$, maxit $= 1$, and restart $= 1000$, and its solution converges within 110 inner iterations. Compared to GMRES, the iterative part in the proposed DD algorithm can converge in a smaller number of iterations, while yielding a much better accuracy. Compared to the conventional direct solution, the proposed method avoids the bottleneck of directly factorizing a large matrix. Only a direct solution in each small subdomain is required. The parallel solver developed in this work costs 26 s in factorization, and completes the time marching within 6.9 hours, whereas the conventional direct solver costs 11.4 hours in factorization, and 10.7 hours in time marching. The machine to run the simulation has 256 GB memory and 20 CPUs. For parallel solver, the entire structure is partitioned into 20 subdomains so that each CPU only solves one subdomain with single thread.

We also check the convergence the our proposed parallel solver from the perspective of relative residual. Fig. 4.10 shows the relative residual when the entire right hand side $\{\tilde{b}\}$ is a random vector. The $relaRes\tilde{b}$ is obtained by gathering local norms according to (4.46). For the parallel solver, we also gather the entire solution vector from every subdomain and calculate $relaResb$ according to (4.38). As expected, $relaRes\tilde{b}$ is very similar to $relaResb$ in the proposed parallel solver. Next, look at the convergence speed. For the first 30 iterations, the proposed parallel solver has a steeper slope and converges faster than GMRES. To reach even higher accuracy, the convergence speed of the proposed parallel solver will saturate to a much slower speed as compared to GMRES. However, from our test on IC package examples, most examples only require 30 iterations and $tol = 10^{-3}$ to achieve accurate results. The lower requirement on tolerance and number of iterations could be from the hybrid nature of the proposed parallel solver, namely being direct (LU) in each subdomain and iterative across subdomains.

4.5.6 Run-time Complexity, Memory Complexity, and Parallel Efficiency

Here, we study the run-time complexity, memory complexity, and parallel efficiency of the proposed DD solver based on the IBM plasma package structure in Fig. 4.9 (a). The computing machine consists of a Dell compute node with two 64-core AMD Epyc 7662 Rome processors (128 cores per node) and 1 TB of memory. All algorithms are implemented in Matlab and run under Matlab version R2020a. The parallel library uses Matlab's built-in *spmd* command, which is based on MPI.

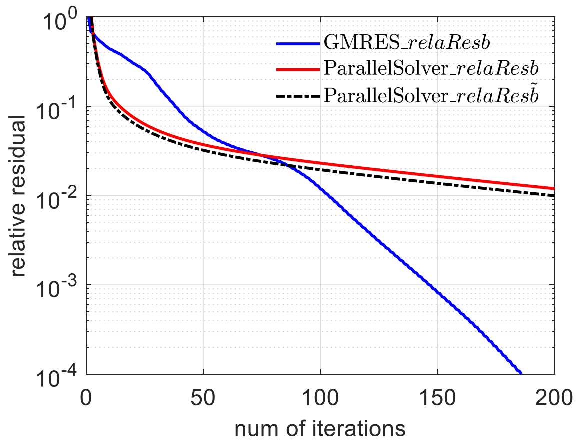


Figure 4.10. Relative residual of plasma package example.

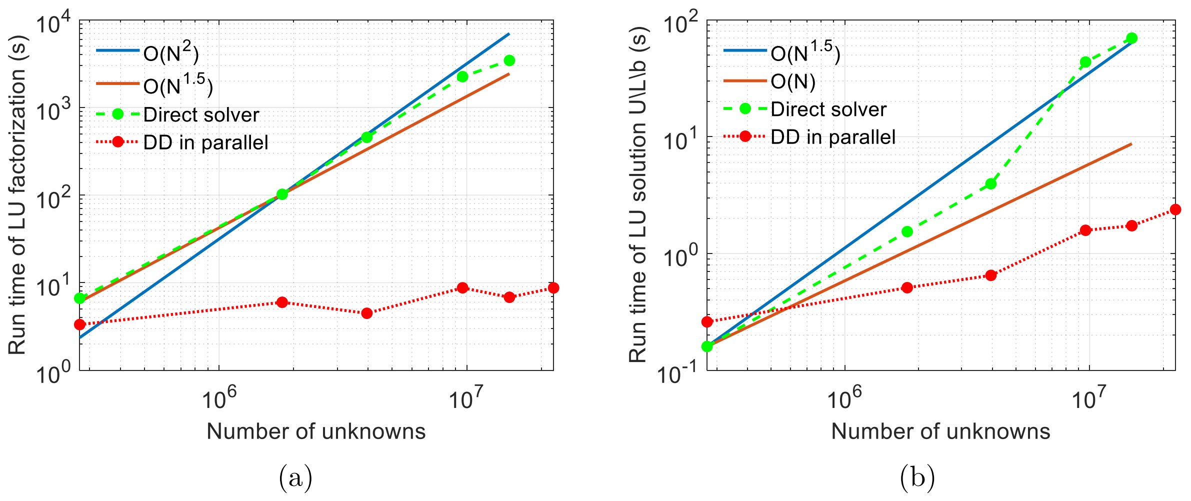


Figure 4.11. Run-time complexity of the proposed DD solver running in parallel. (a) LU factorization. (b) LU solution.

To study run-time complexity and memory complexity, we keep the same mesh accuracy, then simulate a larger and larger area each time to increase N. For a direct solver, only one core is used to solve the entire system. For the parallel DD solver, we increase the number of cores according to N so that the simulated local matrix size at each core is the same. In the example here, we let each core solve a subsystem of about 0.17 million unknowns. Despite the same N/core in each subdomain, the fact that each subdomain has different structures in this realistic package example will add variations to the run-time and memory of the DD solver.

Fig. 4.11 (a) shows the LU factorization time. The DD solver factorizes a similar-size matrix at each core simultaneously, therefore, the run time doesn't increase for large N. However, the LU factorization time of direct solver will scale at $\mathcal{O}(N^2)$ for large N. Fig. 4.11 (a) shows how the DD solver in parallel could benefit from factorizing a much smaller matrix at each core. Fig. 4.11 (b) shows the LU solution time. For direct solver, this LU solution time only comes from the cost of solving each $U\backslash L\backslash b$, which scales around $\mathcal{O}(N^{1.5})$. For DD solver in parallel, the LU solution time comes from two major costs, one is from solving $U\backslash L\backslash b$ and another is from parallel communication cost. The L and U matrices at each core are solved simultaneously and have much smaller size compared to those in direct solver. So, solving $U\backslash L\backslash b$ takes a similar small amount of run time. The increased LU solution time in Fig. 4.11 (b) is from the parallel communication cost, which increases as more computing cores are used for larger N. Fig. 4.11 (b) shows that the parallel communication cost of DD solver scales less than linear. Note that, at each time step, the direct solver only needs to solve $U\backslash L\backslash b$ once. So, the run time of one time step of direct solver is the LU solution time in Fig. 4.11 (b). The DD solver in parallel need to iterative a few iterations till the solution converges, which is typically 30 iterations for package problems. So, the run time of one time step of DD solver in parallel is LU solution time in Fig. 4.11 (b) multiplying number of iterations.

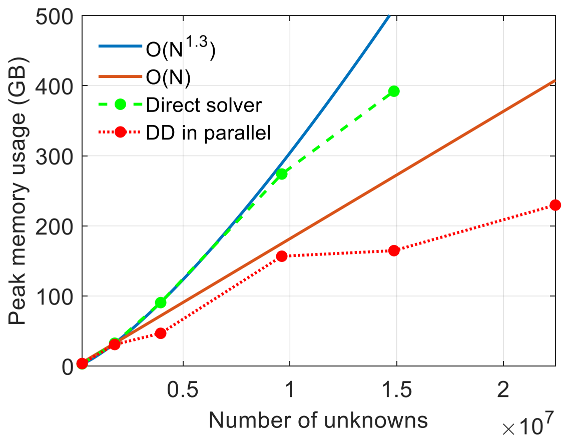
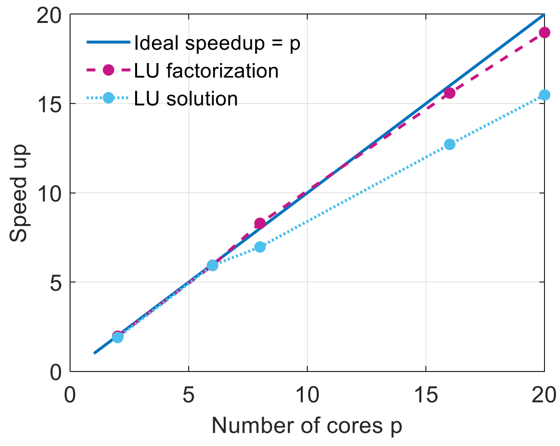
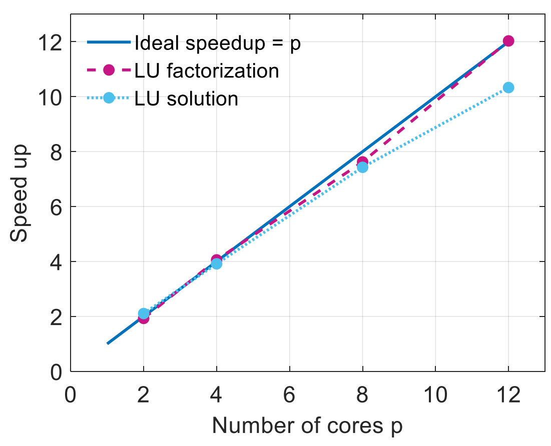


Figure 4.12. Memory complexity of the proposed DD solver running in parallel.

Fig. 4.12 shows the peak memory usage. The major memory cost is at LU factorization. For DD solver in parallel, each core uses similar amount of memory for LU factorization, therefore, the total peak memory scales linearly $\sim \mathcal{O}(N)$. For direct solver, the theoretical memory complexity is $\mathcal{O}(N^{1.5})$. In our test for this example, we find the memory complexity of direct solver is about $\mathcal{O}(N^{1.3})$. For the example with 22.4 million unknowns, the direct solver fails to solve it due to running out of total 1 TB memory. The DD solver in parallel can handle even larger examples because of its lower memory complexity. Besides, considering the usage of distributed memory machines, the DD solver in parallel can solve any large N as long as each small subdomain can fit into one computing node.



(a)



(b)

Figure 4.13. Speed up of DD solver in parallel over DD solver in sequence. (a) N/core is 1.8 million. (b) N/core is 2.9 million.

To demonstrate the parallel efficiency, we let the DD solver run in sequence, namely solving each subdomain one by one with only one computing core. Therefore, we can define a speed up

$$speedup = \frac{\text{Run time of DD solver in sequence}}{\text{Run time of DD solver in parallel}}, \tag{4.47}$$

which reveals the ratio of computing time over parallel overhead. To reduce the structure variance among different subdomains, here, we duplicate the plasma package structure to all used cores, so that every core solves the same system matrix. We also use a random excitation vector to cover different solution modes. In this example, each core handles a local system matrix of 1.8 million unknowns, which costs about 105 s to factorize and about 1.7 s to solve one $U\backslash L\backslash b$.

In the DD algorithm, the LU factorization is embarrassingly parallelizable, therefore, the *speedup* of LU factorization is almost ideal $speedup = p$ as shown in Fig. 4.13. The LU solution time includes 1) solving $U\backslash L\backslash b$, 2) passing interface unknown vectors to neighbor subdomains, and 3) calculating relative residual. The second step, passing interface unknown vectors, is the major source of parallel overhead. When few cores are used, the parallel overhead is relatively small, the *speedup* of LU solution is also close to the ideal $speedup = p$ as in Fig. 4.13. When number of cores p increases, the parallel overhead increases a little bit. For example, when $p = 20$, the parallel overhead is about 0.7 s. The non-optimal *speedup* of LU solution is due to the small solution time as compared to the communication cost. Note that the communication cost is almost a constant, which scales way less than linear when more cores as used. For even larger examples, such as the one in Fig. 4.13 (b), the communication cost will be relatively small, then the *speedup* of LU solution could better approach the ideal $speedup = p$. Limited by the 1 TB memory of the computing machine, we can run at most about 35 M unknowns in parallel.

5. STABILITY CONTROL OF UNSYMMETRICAL NUMERICAL METHODS IN TIME DOMAIN

5.1 Introduction

More and more unsymmetrical numerical methods in time domain are emerging to achieve desired numerical performance, such as modeling the polarized metasurface by surface susceptibility tensors [58], [59], a nonuniform finite-difference time-domain (FDTD) method to allow for the use of non-uniform meshes [60], subgridding algorithms to allow a finer mesh only around certain regions [61]–[63], a matrix-free method to avoid a matrix solution [20], [64], [65], and solutions of other general non-reciprocal problems with unsymmetrical system matrix. Different from classical computational electromagnetics (CEM) algorithms, such as the Yee's FDTD and the finite element method (FEM), which have positive semi-definite system matrices and thereby controllable stability, the stability of unsymmetrical numerical systems are generally undetermined.

In a numerical system in time domain, there are two levels of considerations regarding stability. The first level is on the time marching scheme regarding the choice of time step, such as the famous Courant–Friedrichs–Lewy (CFL) limit for Δt in Yee's FDTD. Most of the stability analysis focuses on which time step can make the time marching stable, assuming the system matrix already has desired physical property such as passivity. Commonly used methods to analyze time-domain stability include the Von-Neumann method [66], [67] and the Z-transform method [68]. However, for numerical systems with an unsymmetrical system matrix, one cannot assume their system matrix has the right sign of eigenvalues, this has to be ensured by construction. If the system matrix has complex-valued eigenvalues or an incorrect sign in its real eigenvalues, then no matter which method is used for time marching, the time-domain simulation would be totally unstable. Therefore, for unsymmetrical numerical systems, a more fundamental problem to study in stability analysis is whether the system matrix statifies physical property. The research problem we study here is how to ensure the numerical system has physically correct eigenvalues by construction.

On the one hand, the lack of theories on unsymmetrical matrices makes it hard to examine the eigenmodes of unsymmetrical numerical systems. Right now, most stability analyses for unsymmetrical numerical systems rely on checking the eigenvalues and need to be done case by case [69]–[71]. Generally, this stability analysis would require solving all eigenvalues of a large asymmetric matrix, which is too expensive to be done for large problems. On the other hand, the almost guaranteed existence of complex eigenvalues in an asymmetric matrix means that an unsymmetrical numerical system is very likely to have some unstable eigenmodes, even though those unstable eigenmodes might not be dominant or not even be observable in most circumstances. What's worse, as proved in [61], [72], traditional explicit time marching with an unsymmetrical system matrix becomes absolutely unstable, because an unsymmetrical matrix can support complex-valued and even negative eigenvalues. All these call for a feasible stability theory to prove and control the stability of unsymmetrical numerical systems.

Recently, a matrix-free time-domain (MFTD) method has been developed in both 2-D [20] and 3-D [64], which solves Maxwell's equations in general unstructured meshes while not requiring a matrix solution. The MFTD has a diagonal mass matrix in nature independent of the element shape used for discretization. To overcome the stability problem while retaining the advantage of a diagonal mass matrix, a backward difference scheme is employed for time marching. Then, the inverse of the system matrix is made explicit by using a series expansion, thus avoiding a matrix solution. In [65], a new time marching scheme is developed to make the MFTD method truly explicit, hence removing the need for using the backward difference and the series expansion to avoid a matrix solution. In all previous MFTD formulations, the system matrix is still asymmetric, therefore, the stability of MFTD still requires a further proof.

In this work, we propose a stability analysis method that reduces the stability analysis in the entire computational domain to that in a single-element, therefore avoiding solving all eigenvalues of a large unsymmetrical matrix. The proposed stability analysis method also provides a way to control the stability of a large numerical system by simply controlling the stability of every dissembled single element. Compared to the entire system, each dissembled single element has a much smaller local system matrix and less parameters, therefore, controlling the stability of every single element is much easier. In this work, we show the application of the method to prove and control the stability of the MFTD method. To distinguish from original MFTD, we call the new MFTD with controlled stability as CS-MFTD. In the 2-D MFTD, we can control the stability by controlling the **H**-loop size. In the 3-D MFTD, we found using the original scheme, complex eigenvalues arise from a single tetrahedron. Hence, we propose a new 3-D patch-based MFTD method in unstructured meshes. The proposed new 3-D MFTD also has well controlled stability.

5.2 Stability Analysis Theory

5.2.1 Conventional Stability Criterion by Solving All Eigenvalues of the Entire System Matrix

The numerical system in time domain of many partial differential solvers can be written as

$$\frac{\partial x}{\partial t} = \mathbf{A}x, \quad t \geq 0, \tag{5.1}$$

with $\mathbf{A}$ being the system matrix and x being the solution vector. The governing equation (5.1) has a general solution

$$x(t) = \mathrm{e}^{\mathbf{A}t}x_0, \quad t \geq 0. \tag{5.2}$$

Here, x_0 is the initial condition. Before moving forward to a temporal discretization that further discretizes $\frac{\partial}{\partial t}$, the spatial discretization and corresponding system matrix $\mathbf{A}$ already determines a level of stability for the numerical system. As can be seen from (5.2), a spatial discretization and its resulting numerical system is stable only if matrix $\mathrm{e}^{\mathbf{A}t}$ is always bounded for a passive problem. In other words, the numerical system (5.1) is stable only if the spectral radius (largest absolute value of all eigenvalues) of matrix $\mathrm{e}^{\mathbf{A}t}$ is always bounded.

To find the spectral radius $\rho(\mathrm{e}^{\mathbf{A}t})$, we first do an eigenvalue decomposition for $\mathbf{A}$ matrix

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}, \tag{5.3}$$

where $\mathbf{\Lambda}$ denotes the diagonal matrix of all eigenvalues of system matrix $\mathbf{A}$. Each column of matrix $\mathbf{V}$ is an eigenvector. Then, using the property of matrix exponential, $\mathrm{e}^{\mathbf{A}t}$ can be decomposed into

$$\mathrm{e}^{\mathbf{A}t} = \mathrm{e}^{\mathbf{V}\mathbf{\Lambda}t\mathbf{V}^{-1}} = \mathbf{V}\mathrm{e}^{\mathbf{\Lambda}t}\mathbf{V}^{-1}. \tag{5.4}$$

Therefore, the eigenvalues of matrix $\mathrm{e}^{\mathbf{A}t}$ are simply $\mathrm{e}^{\mathbf{\Lambda}t}$. For a stable numerical system (5.1), $\rho(\mathrm{e}^{\mathbf{A}t}) = \max(|\mathrm{e}^{\mathbf{\Lambda}t}|)$ always being bounded for arbitrary $t \geq 0$ means that all eigenvalues of $\mathbf{A}$ should have non-positive real part

$$\mathbf{\Lambda}.real \leq 0, \tag{5.5}$$

which also means the spectral radius of $\mathrm{e}^{\mathbf{A}t}$ should be not only bounded but also bounded by 1, namely

$$\rho(\mathrm{e}^{\mathbf{A}t}) \leq 1. \tag{5.6}$$

From the analysis above, we can clearly see that the stability is directly determined by the eigenvalues of system matrix $\mathbf{A}$ as in (5.5) or (5.6).

To prove that numerical system (5.1) is stable, we need to prove either (5.5) or (5.6) is satisfied. However, for a large unsymmetrical numerical system, directly checking the real part of all $\mathbf{\Lambda}$ is too expensive as it requires solving all eigenvalues of a large matrix.

5.2.2 Proposed Stability Criterion by Reducing to Single Element

In this work, we propose to reduce the scope of the stability analysis from the entire-domain system matrix to dissembled single-element local system matrix.

In many PDE methods, the entire-domain system matrix $\mathbf{A}$ is assembled from all N elements

$$\mathbf{A} = \sum_{\mathrm{i}=1}^{N} \mathbf{A}_{\mathrm{i}}, \tag{5.7}$$

where $\mathbf{A}_{\mathrm{i}}$ is the contribution from the i-th element. The assembly in (5.7) inspires us to think that maybe we could regulate every single element to guarantee the overall stability of the entire numerical system. Fortunately, we do find such a connection through the spectral radius.

Before proving our proposed stability criterion, we first prove a few lemmas as below.

Lemma 5.2.1. *All eigenvalues* $\mathbf{\Lambda}$ *of matrix* $\mathbf{A}$ *satisfy* $\mathbf{\Lambda}.real \leq 0 \Leftrightarrow \rho(\mathrm{e}^{\mathbf{A}}) \leq 1$.

Proof. $\mathrm{e}^{\mathbf{A}}$ has eigenvalues $\mathrm{e}^{\mathbf{\Lambda}}$, $\rho(\mathrm{e}^{\mathbf{A}}) = \max(|\mathrm{e}^{\mathbf{\Lambda}}|)$, so this Lemma is obvious. □

Lemma 5.2.2. *If* $\rho(\mathrm{e}^{\mathbf{A}}) \leq 1$ *and* c *is a positive number, then* $\rho(\mathrm{e}^{\mathbf{A}/c}) \leq 1$*.*

Proof. $\mathrm{e}^{\mathbf{A}}$ and $\mathrm{e}^{\mathbf{A}/c}$ have the same sign for the real part of their eigenvalues. According to Lemma 5.2.1, this Lemma is also valid. □

Lemma 5.2.3. *For arbitrary square matrix* $\mathbf{P}$ *and* $\mathbf{Q}$ *of the same size, if* $\rho(\mathrm{e}^{\mathbf{P}}) \leq 1$ *and* $\rho(\mathrm{e}^{\mathbf{Q}}) \leq 1$*, then* $\rho(\mathrm{e}^{\mathbf{P}+\mathbf{Q}}) \leq 1$*.*

Proof. We can apply Lie product formula to expand the matrix exponential as

$$\mathrm{e}^{\mathbf{P}+\mathbf{Q}} = \lim_{c\to\infty} \left(\mathrm{e}^{\mathbf{P}/c}\mathrm{e}^{\mathbf{Q}/c}\right)^c = \lim_{c\to\infty} \left(\mathrm{e}^{\mathbf{Q}/c}\mathrm{e}^{\mathbf{P}/c}\right)^c, \tag{5.8}$$

where c is a sufficiently large constant number.

From Lie product formula (5.8), we can see that $\mathrm{e}^{\mathbf{P}/c}$ and $\mathrm{e}^{\mathbf{Q}/c}$ commute to each other, therefore, the submultiplicative property of spectral radius holds true

$$\rho(\mathrm{e}^{\mathbf{P}/c}\mathrm{e}^{\mathbf{Q}/c}) \leq \rho(\mathrm{e}^{\mathbf{P}/c})\rho(\mathrm{e}^{\mathbf{Q}/c}). \tag{5.9}$$

Given $\rho(\mathrm{e}^{\mathbf{P}}) \leq 1$ and $\rho(\mathrm{e}^{\mathbf{Q}}) \leq 1$, from Lemma 5.2.2, we have

$$\rho(\mathrm{e}^{\mathbf{P}/c}) \leq 1, \quad \rho(\mathrm{e}^{\mathbf{Q}/c}) \leq 1. \tag{5.10}$$

Since $\rho(\mathbf{M}^c) = (\rho(\mathbf{M}))^c$, we can prove

$$\begin{aligned}\rho(\mathrm{e}^{\mathbf{P}+\mathbf{Q}}) &= \lim_{c\to\infty} \left(\rho(\mathrm{e}^{\mathbf{P}/c}\mathrm{e}^{\mathbf{Q}/c})\right)^c \\ &\leq \lim_{c\to\infty} \left(\rho(\mathrm{e}^{\mathbf{P}/c})\rho(\mathrm{e}^{\mathbf{Q}/c})\right)^c \leq 1.\end{aligned} \tag{5.11}$$

□

With above lemmas, we can prove our proposed stability criterion

Theorem 5.2.4. *If every element satisfies* $\rho(\mathrm{e}^{\mathbf{A}_i t}) \leq 1$*, then the entire system satisfies* $\rho(\mathrm{e}^{\mathbf{A}t}) \leq 1$ *therefore is stable.*

Proof. We can iteratively prove Theorem 5.2.4 as below. Define

$$\mathbf{A}^{(n)} = \sum_{\mathrm{i}=1}^{n} \mathbf{A}_{\mathrm{i}}, \quad 1 \leq n \leq N, \tag{5.12}$$

which is the assembled first n elements in (5.7). Initially when $n = 1$, $\mathbf{A}^{(1)} = \mathbf{A}_1$ thus $\rho(\mathrm{e}^{\mathbf{A}^{(1)}t}) \leq 1$ is satisfied. Then, given $\rho(\mathrm{e}^{\mathbf{A}^{(n-1)}t}) \leq 1$ being true, because $\mathbf{A}^{(n)} = \mathbf{A}^{(n-1)} + \mathbf{A}_n$ and $\rho(\mathrm{e}^{\mathbf{A}_n t}) \leq 1$, we can directly apply Lemma 5.2.3 to approve $\rho(\mathrm{e}^{\mathbf{A}^{(n)}t}) \leq 1$. Iteratively increase n to N and we can prove $\rho(\mathrm{e}^{\mathbf{A}t}) \leq 1$. □

Using Lemma 5.2.1, we can obtain a more practical criterion to determine stability by directly looking at the eigenvalues of $\mathbf{A}_{\mathrm{i}}$.

Theorem 5.2.5. *If every element's matrix $\mathbf{A}_{\mathrm{i}}$ has all its eigenvalues $\mathbf{\Lambda}_{\mathrm{i}}.real \leq 0$, then the entire system satisfies $\rho(\mathrm{e}^{\mathbf{A}t}) \leq 1$ therefore is stable.*

Note that even though the criterion in Theorem 5.2.5 is not satisfied, the numerical system still can be stable because the criterion is only a sufficient condition for stability. But, as proved above, as along as the criterion is satisfied, we can guarantee the numerical system is stable.

5.3 Control the Stability of MFTD Method

In this section, we apply our proposed stability theory to control the stability of MFTD in both 2-D and 3-D.

5.3.1 Review of Matrix-Free Time-Domain Method

Consider a physical structure discretized into either a regular grid or an unstructured mesh consisting of arbitrarily shaped elements. Based on [20], Maxwell's equations are discretized as the following

$$\mathbf{S}_{\mathrm{e}}\{\mathrm{e}\} = -\mathbf{D}_{\mu}\frac{\partial\{h\}}{\partial t} \tag{5.13}$$

$$\mathbf{S}_{h}\{h\} = \mathbf{D}_{\epsilon}\frac{\partial\{\mathrm{e}\}}{\partial t} + \mathbf{D}_{\sigma}\{\mathrm{e}\} + \{\mathrm{j}_s\}, \tag{5.14}$$

where $\mathbf{S}_e\{e\}$ represents a discretized curl of $\mathbf{E}$, $\mathbf{S}_h\{h\}$ denotes a discretized curl of $\mathbf{H}$, the $\mathbf{D}_\mu$, $\mathbf{D}_\epsilon$ and $\mathbf{D}_\sigma$ are the diagonal matrices of permeability, permittivity, and conductivity respectively. The $\{e\}$ is a global vector of unknown electric fields whose i-th entry is $\mathbf{E}$ at point $\mathbf{r}_{ei}$ along direction $\hat{e}_i$, thus $e_i = \mathbf{E}(\mathbf{r}_{ei}) \cdot \hat{e}_i$. The $\mathbf{E}$'s points and directions are associated with the curl-conforming vector basis functions used to expand $\mathbf{E}$ in each element. From (5.13), using higher-order bases, we can evaluate $\mathbf{H}$ at any point $\mathbf{r}_{hi}$ along any direction $\hat{h}_i$, which is h_i, with a desired order of accuracy.

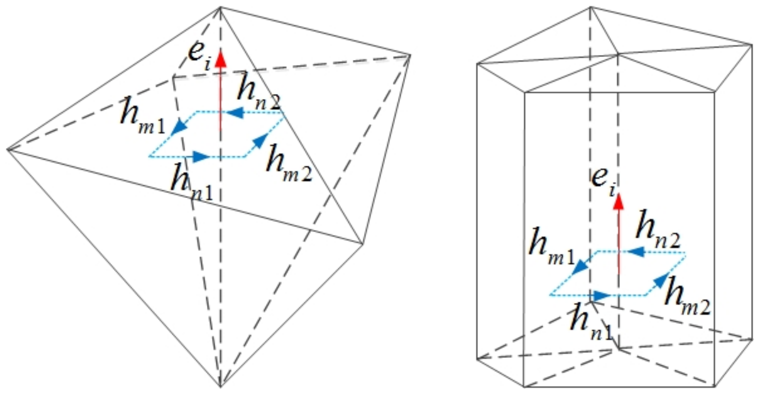


Figure 5.1. H points and directions.

Where to choose h_i is the key to ensure the accuracy in the matrix-free method. The h_i are chosen based on e so that e can be generated accurately from h. Specifically, for each e_i, the $\mathbf{H}$ unknowns are chosen along a rectangular loop perpendicular to e_i, and centering e_i, as shown in Fig. 5.1. In this way, the $\mathbf{H}$ fields selected can generate e_i accurately, which is manifested by the discretization of Ampere's law shown in (5.14). Different from the FDTD method, here, the $\mathbf{H}$ points and directions do not form a dual mesh. Only a single mesh is needed. No interpolations and projections are required either. Since the resulting matrices in front of the first-order time derivative in (5.13) and (5.14) are diagonal, an explicit marching is free of a matrix solution.

The h is the global vector of unknown magnetic fields whose i-th entry is $\mathbf{H}$ at point $\mathbf{r}_{hi}$ along direction $\hat{h}_i$, thus $\mathbf{H}(\mathbf{r}_{hi}) \cdot \hat{h}_i$. For each e_i, there are four to discretize Faraday's law, we expand the electric field $\mathbf{E}$ in each element by vector bases, yielding $\mathbf{E} = \sum_{j=1}^{m} u_j \mathbf{N}_j$, where

Eqn. (5.13) and (5.14) can be solved in a leapfrog way which is free of matrix solutions. We can also eliminate h and solve e as the following

$$\frac{\partial^2 \{\mathrm{e}\}}{\partial t^2} + \mathbf{D}_\epsilon^{-1}\mathbf{D}_\sigma \frac{\partial \{\mathrm{e}\}}{\partial t} + \mathbf{S}\{\mathrm{e}\} = -\mathbf{D}_\epsilon^{-1}\frac{\partial \{\mathrm{j}_s\}}{\partial t} \tag{5.15}$$

where

$$\mathbf{S} = \mathbf{D}_\epsilon^{-1}\mathbf{S}_h\mathbf{D}_\mu^{-1}\mathbf{S}_\mathrm{e}. \tag{5.16}$$

The governing equation of MFTD, written as general format (5.1), is

$$\frac{\partial}{\partial t}\left\{\begin{array}{c}\{\mathrm{e}\}\\ \{h\}\end{array}\right\} = \left[\begin{array}{cc}0 & \mathbf{D}_\epsilon^{-1}\mathbf{S}_h\\ -\mathbf{D}_\mu^{-1}\mathbf{S}_\mathrm{e} & 0\end{array}\right]\left\{\begin{array}{c}\{\mathrm{e}\}\\ \{h\}\end{array}\right\} - \left\{\begin{array}{c}\mathbf{D}_\epsilon^{-1}\{\mathrm{j}_s\}\\ 0\end{array}\right\}. \tag{5.17}$$

5.3.2 Control the Stability of 2-D MFTD Method

Algorithm 3 Choose **H** locations in 2-D original MFTD

```
1: Define a constant hyper-parameter $1/Hlratio \in (0, 1]$
2: for every E point at $\mathbf{r}_{\mathrm{ei}}$ along direction $\hat{\mathrm{e}}_\mathrm{i}$ do
3:     Draw a line passing $\mathbf{r}_{\mathrm{ei}}$ and vertical to $\hat{\mathrm{e}}_\mathrm{i}$
4:     Let the line intersect with all edges in the entire mesh
5:     Find the intersection point $\mathbf{r}_{p\mathrm{i}}$ that is closest to $\mathbf{r}_{\mathrm{ei}}$
6:     Choose 2 H points related to this E point:
7:         One H point is at $\mathbf{r}_{\mathrm{ei}} + (\mathbf{r}_{p\mathrm{i}} - \mathbf{r}_{\mathrm{ei}})/Hlratio$
8:         The other H point is at $\mathbf{r}_{\mathrm{ei}} - (\mathbf{r}_{p\mathrm{i}} - \mathbf{r}_{\mathrm{ei}})/Hlratio$
9: end for
```

The 2-D MFTD studied in this work uses triangular mesh and first order vector bases. Details about original 2-D MFTD can be found in [20]. Each dissembled single element is a triangle. When using first order vector bases, each dissembled triangle has 8 **E** points and 10 **H** points.

In original 2-D MFTD [20], **H** directions are always normal to the patch face, while **H** locations can be chosen anywhere inside the triangular patch. In [20], the **H** locations are chosen as in Algorithm 3. Readers may refer to Fig. 4 in [20] for better understanding. However, according to our numerical testing, such a choice of **H** locations cannot always

Algorithm 4 Choose **H** locations in 2-D CS-MFTD

```
1: Define a constant hyper-parameter $1/Hlratio \in (0, 0.1]$
2: for every E point at $\mathbf{r}_{\mathrm{ei}}$ along direction $\hat{\mathrm{e}}_{\mathrm{i}}$ do
3:     Draw a line passing $\mathbf{r}_{\mathrm{ei}}$ and vertical to $\hat{\mathrm{e}}_{\mathrm{i}}$
4:     Let the line intersect with all edges in the entire mesh
5:     Find the intersection point $\mathbf{r}_{p\mathrm{i}}$ that is closest to $\mathbf{r}_{\mathrm{ei}}$
6:     Track the min intersection distance $L_{ep\mathrm{i}} = \|\mathbf{r}_{p\mathrm{i}} - \mathbf{r}_{\mathrm{ei}}\|$ and line direction $\hat{\mathbf{r}}_{ep\mathrm{i}} = (\mathbf{r}_{p\mathrm{i}} - \mathbf{r}_{\mathrm{ei}})/L_{ep\mathrm{i}}$
7: end for
8: Set a fixed H loop size $L_H = \min_{\mathrm{i}}(L_{ep\mathrm{i}})/Hlratio$
9: for every E point at $\mathbf{r}_{\mathrm{ei}}$ along direction $\hat{\mathrm{e}}_{\mathrm{i}}$ do
10:     Choose 2 H points related to this E point:
11:         One H point is at $\mathbf{r}_{\mathrm{ei}} + \hat{\mathbf{r}}_{ep\mathrm{i}} \times L_H$
12:         The other H point is at $\mathbf{r}_{\mathrm{ei}} - \hat{\mathbf{r}}_{ep\mathrm{i}} \times L_H$
13: end for
```

guarantee a good stability for either single element or entire system, no matter how we tune the only hyper-parameter $1/Hlratio$ in Algorithm 3.

Using our developed new stability analysis theory, we only need to control the stability of every single triangle. Each dissembled triangle only has 8 **E** points and 10 **H** points, so, the dissembled local system matrix $\mathbf{A}_{\mathrm{i}}$ has a small size 18×18. We can analytically write down $\mathbf{A}_{\mathrm{i}}$. However, even though local system matrix $\mathbf{A}_{\mathrm{i}}$ of single triangle looks simple, the unsymmetrical nature of $\mathbf{A}_{\mathrm{i}}$ makes it hard to analytically find any insights into the eigenvalues.

After many trials, we eventually find one way to control the stability of single triangle of arbitrary shape. The method to get 2-D CS-MFTD is described in Algorithm 4. Compared to 2-D original MFTD where every **H** loop has different loop size, our proposed 2-D CS-MFTD uses a fixed **H** loop size for every **E** point in the entire mesh. Due to the lack of theory to analytically solve eigenvalues of unsymmetrical matrix, we numerically prove the controlled stability of 2-D CS-MFTD in Section 5.4. From our numerical testings, a good choice of the hyper-parameter $1/Hlratio$ in Algorithm 4 is $1/Hlratio = 0.01$.

5.3.3 New 3-D MFTD Method with Controlled Stability

The 3-D original MFTD [65] uses a tetrahedral mesh and first order 3-D vector bases. Each dissembled single element is a tetrahedron. When using first order vector bases, each dissembled tetrahedron has 20 **E** points and at least 8 **H** points. The exact number of **H** points per tetrahedron is determined by if edge **H** falls into this tetrahedron. We tried many methods to control the stability of a single tetrahedron, such as setting a different **H** loop size, uniformly distributing edge **H** in surrounding tetrahedrons, etc. However, none of the methods is able to guarantee the stability of a single tetrahedron of arbitrary shapes.

To develop a 3-D CS-MFTD, we decide to continue the success of the 2-D CS-MFTD. A single triangular patch of an arbitrary shape has well controlled stability in 2-D CS-MFTD as shown in Algorithm 4. Meanwhile, we can view each 3-D tetrahedron element as an assembly of four triangular patches. If we use the first order 2-D vector bases for every triangular patch in the entire 3-D mesh, then the entire 3-D mesh can again be an assembly of 2-D patches, just like that in 2-D CS-MFTD. The new 3-D MFTD following this idea does have well controlled stability and decent accuracy. In such a 3-D CS-MFTD, each 3-D triangular patch is just viewed as a 2-D patch, so, we can directly reuse the method and code in 2-D CS-MFTD, except that the z direction in the 2-D CS-MFTD needs to be set as the normal direction of this 3-D triangular patch. The $\mathbf{S}_{\mathrm{e}}$ matrix in the 3-D CS-MFTD can be generated exactly the same as the $\mathbf{S}_{\mathrm{e}}$ in the 2-D CS-MFTD. $\mathbf{S}_h$ matrix's rows corresponding to the face **E** points are also the same as those in 2-D, while the rows corresponding to the edge **E** points have the following entry

$$\mathbf{S}_{h,\mathrm{ij}} = \pm \frac{2}{L_H \times m}, \tag{5.18}$$

where j denotes the global index of the **H** point associated with this edge $\mathrm{e_i}$. L_H is the fixed **H** loop size obtained from Algorithm 4, and m is the number of tetrahedrons sharing this edge. The $\pm$ sign is determined by right-hand-rule, just as in original MFTD.

Algorithm 5 Numerically Check If System Matrix **A** is Stable

```
1: Numerically calculate all nonzero eigenvalues $\mathbf{\Lambda}$ of $\mathbf{A}$
2: for every nonzero eigenvalue $\lambda_\mathrm{i} \in \mathbf{\Lambda}$ do
3:     if $\lambda_\mathrm{i}.real > 0$ then
4:         $\mathrm{R_i} = \lambda_\mathrm{i}.real/|\lambda_\mathrm{i}.imag|$
5:     else
6:         $\mathrm{R_i} = 0$
7:     end if
8: end for
9: Calculate MR of matrix $\mathbf{A}$: $\mathrm{MR} = \max_\mathrm{i}(\mathrm{R_i})$
10: if MR is very small, e.g. $\mathrm{MR} < 10^{-6}$ then
11:     say system matrix $\mathbf{A}$ is stable
12: else
13:     say system matrix $\mathbf{A}$ is unstable
14: end if
```

The new 3-D CS-MFTD uses the time marching scheme developed in [65]. Since the system matrix has well controlled stability, there is only one upper bound for the time step Δt [65]

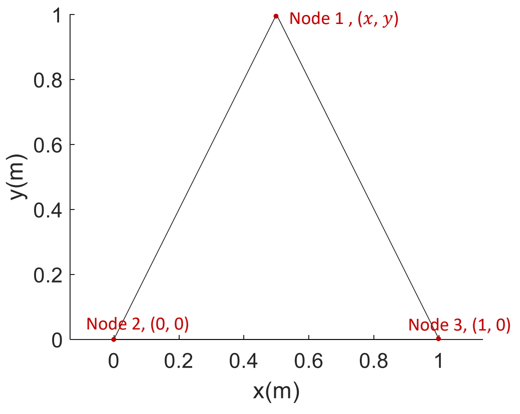
$$\Delta t < \sqrt{\frac{4}{3}} \frac{1}{\sqrt{\lambda_{max}}} = \sqrt{\frac{4}{3}} \frac{1}{\omega_{max}}, \tag{5.19}$$

where λ_{max} is the largest eigenvalue of **S** matrix. ω_{max} is the largest resonant frequency of the numerical system, which can be obtained from the largest imaginary part of system matrix **A**.

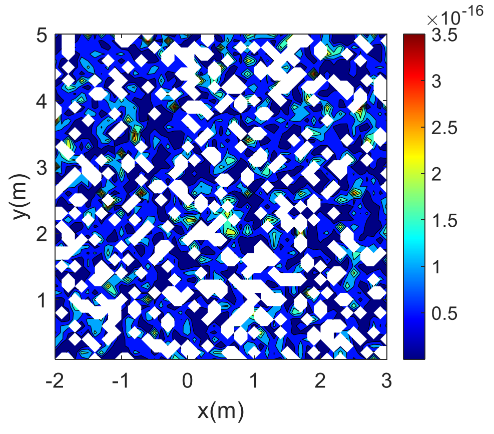
5.4 Numerical Results of 2-D CS-MFTD Method

5.4.1 Controlled Stability of 2-D Triangular Meshes of Arbitrary Shape

Here, we numerically prove that our the proposed 2-D CS-MFTD can control the stability of MFTD in an arbitrary triangle. To do so, we fix two nodes as shown in Fig. 5.2 (a), then, test 2-D CS-MFTD when node 1 is at different locations. In most meshing tools like the DistMesh [73] used in this paper, very skewed triangles will be further partitioned into better structured triangles. So, without losing much generosity, we sample and test some triangle shapes to represent arbitrary triangles. For example, here, x coordinate of node 1 is sampled



(a)



(b)

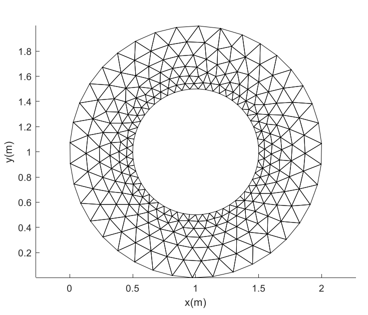
Figure 5.2. Single triangle. (a) Node positions of this triangle. (b) MR of system matrix **A** vs. different x-y locations of node 1.

every 0.1 m from -2 m to 3 m, and y coordinate of node 1 is sampled every 0.1 m from 0.1 m to 5 m. The sampled structures include most triangular shapes used in a 2-D meshing.

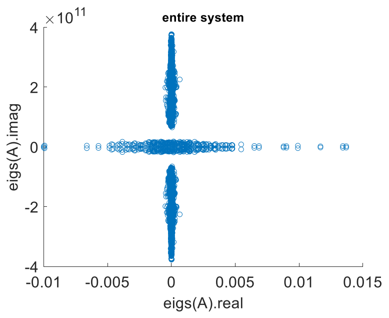
For each sampled triangular shape, we generate its system matrix **A** according to the 2-D CS-MFTD. The scaling factor of fixed **H** loop size is $1/Hlratio = 0.01$. The MR of system matrix **A** is numerically calculated following Algorithm 5, which is then plotted in Fig. 5.2 (b). The left blank area in white color in Fig. 5.2 (b) means MR is less than 10^{-20}. Compared to the machine error of 10^{-15} level, we can safely say the system matrix **A** of each patch has pure imaginary eigenvalues. Since the above sampled triangles numerically represent all regular shapes that are used in 2-D meshing, we thereby numerically prove that 2D triangles of arbitrary shapes always have a guaranteed stability.

5.4.2 Wave Propagation in a 2-D Ring Mesh

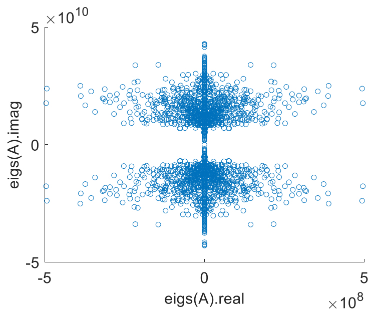
A 2-D ring centered at (1.0 m, 1.0 m) with inner radius 0.5 m and outer radius 1.0 m is simulated in free space. The triangular mesh is shown in Fig. 5.3 (a). The discretization results in 826 edges and 519 triangular patches. The same mesh is solved with both 2-D original MFTD and 2-D CS-MFTD. In 2-D CS-MFTD, the scaling factor of fixed **H** loop size is $1/Hlratio = 0.01$. In 2-D original MFTD, $1/Hlratio = 0.5$.



(a)

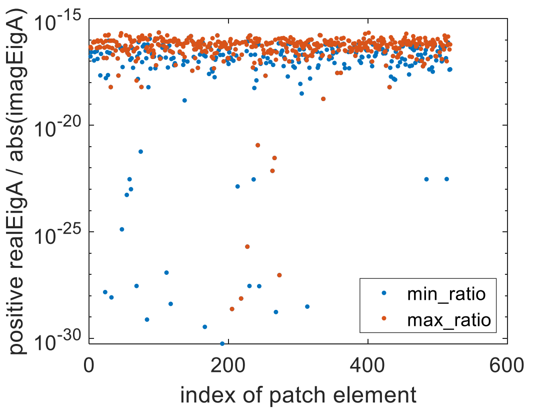


(b)

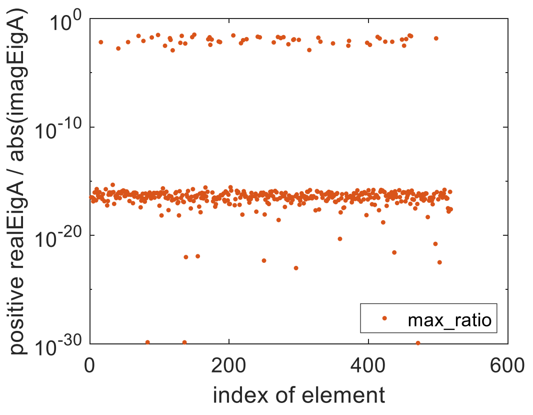


(c)

Figure 5.3. 2-D ring. (a) Triangular mesh of a 2-D ring. (b) Eigenvalues of entire system matrix **A** in CS-MFTD. (c) Eigenvalues of entire system matrix **A** in original MFTD.



(a)

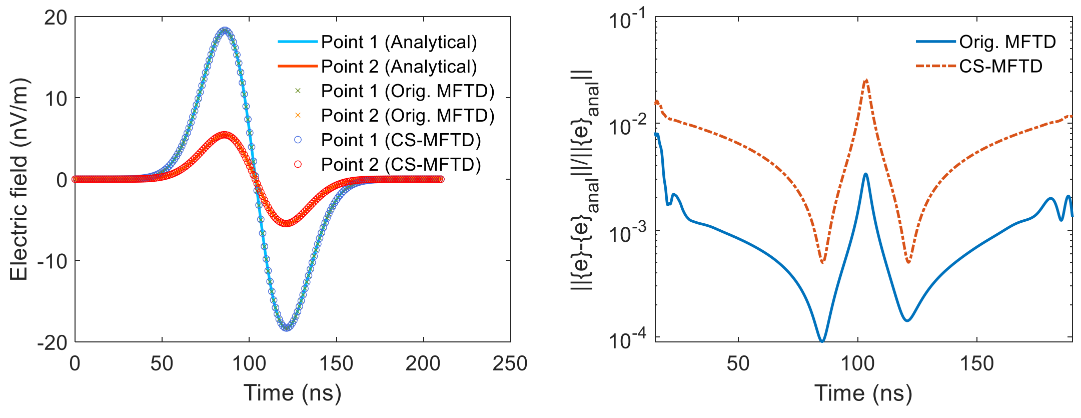


(b)

Figure 5.4. MR of each patch element of the 2-D ring. (a) CS-MFTD. (b) Original MFTD.

To show the stability of the numerical system, we directly solve all the nonzero eigenvalues of entire system matrix **A**. Fig. 5.3 (b) is all the 3,114 nonzero eigenvalues of system matrix **A** in 2-D CS-MFTD. Clearly, the real part of the eigenvalues is much smaller than the imaginary part, thus, we can numerically say that $\mathbf{\Lambda}.real \leq 0$. Therefore, 2-D CS-MFTD is stable in this ring mesh. Fig. 5.3 (c) is all the nonzero eigenvalues of system matrix **A** in 2-D original MFTD. Many eigenvalues in 2-D original MFTD have large positive real part, which correspond to unstable modes in the numerical system.

In Fig. 5.4, we further show the MR of each dissembled patch when using proposed 2-D CS-MFTD or 2-D original MFTD. In original MFTD, a few dissembled patches have bad stability. In contrast, with CS-MFTD, every patch has controlled stability. Following Theorem 5.2.5, we can prove that 2-D CS-MFTD in the entire ring mesh is stable.

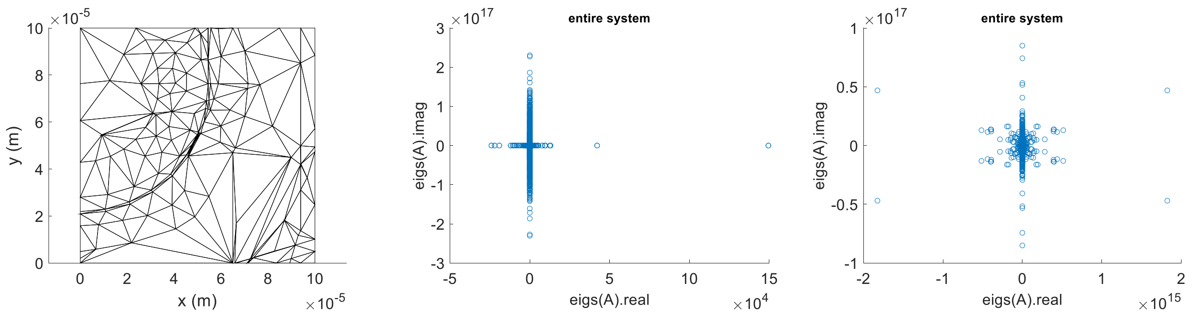


(a) (b)

Figure 5.5. Accuracy of the proposed CS-MFTD in 2-D ring example. (a) Simulated two electric fields in comparison with analytical results. (b) Entire **E** field solution error as a function of time.

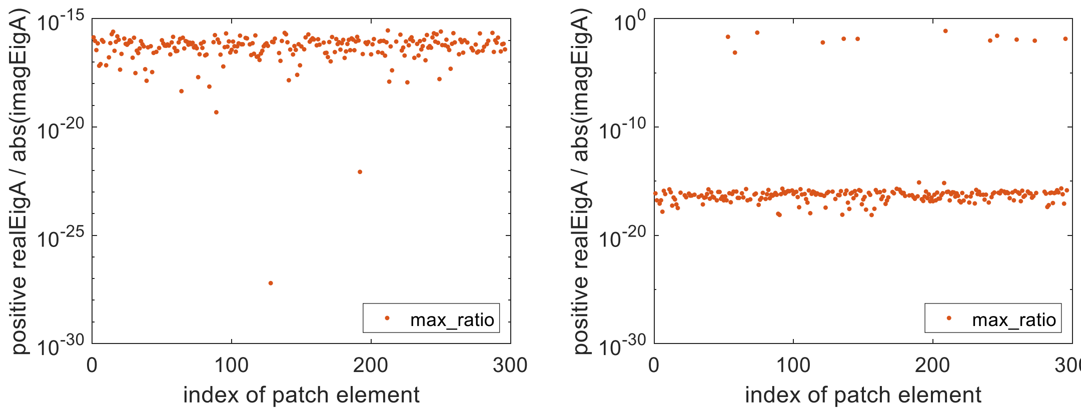
To investigate the accuracy of the proposed method in such a mesh, we consider that the most convincing comparison is a comparison with analytical solution. Although the structure is irregular, we can use it to study a free-space wave propagation problem whose analytical solution is known. To do so, we impose an analytical boundary condition, i.e. the known value of tangential **E**, on the boundary of the problem, which comprises the innermost and outermost circles; we then numerically simulate the fields inside the computational domain and correlate the results with the analytical solution. The incident **E**, which is also the total field in the given problem, is specified as $\mathbf{E} = \hat{y}f(t - x/c)$, where $f(t) = 2(t - t_0)\exp(-(t - t_0)^2/\tau^2)$, $\tau = 2.5 \times 10^{-8}$ s, $t_0 = 4\tau$, and c denotes the speed of light. The time step used in original MFTD is $\Delta t = 2.0 \times 10^{-11}$ s, and $\Delta t = 2.0 \times 10^{-12}$ s in CS-MFTD. We sample two points and plot their electric fields in Fig. 5.5 (a). The relative error of the whole solution vector is shown in Fig. 5.5 (b). It can be seen clearly that the electric fields solved from both original MFTD and CS-MFTD have an excellent agreement with analytical results. The center peak in Fig. 5.5 (b) is due to the comparison with close to zero fields. Note that, compared to original MFTD, CS-MFTD in this ring mesh has slightly worse overall accuracy, which could be from more skewed discretization in terms of **H** points.

5.4.3 Wave Propagation in a 2-D Cavity Discretized Into a Highly Unstructured Mesh



(a) (b) (c)

Figure 5.6. 2-D cavity. (a) Highly unstructured triangular mesh of the 2-D cavity. (b) Eigenvalues of entire system matrix **A** in CS-MFTD. (c) Eigenvalues of entire system matrix **A** in original MFTD.



(a) (b)

Figure 5.7. MR of each patch element of the 2-D cavity. (a) CS-MFTD. (b) Original MFTD.

To examine the robustness of the proposed 2-D CS-MFTD in handling unstructured meshes, we simulate a cavity discretized into a highly irregular mesh as shown in Fig. 5.6 (a). The discretization results in 463 edges and 296 triangular patches. The same mesh is solved with both 2-D original MFTD and 2-D CS-MFTD. In 2-D CS-MFTD, the scaling factor of fixed **H** loop size is $1/Hlratio = 0.01$. In 2-D original MFTD, $1/Hlratio = 2/3$.

To show the stability of the numerical system, we directly solve all the nonzero eigenvalues of entire system matrix **A**. Fig. 5.6 (b) is all the 1,776 nonzero eigenvalues of system matrix **A** in 2-D CS-MFTD. Clearly, the real part of the eigenvalues is much smaller than the imaginary part, thus, we can numerically say that $\mathbf{\Lambda}.real \leq 0$. Therefore, 2-D CS-MFTD is stable in this highly unstructured mesh. Fig. 5.6 (c) is all the nonzero eigenvalues of system matrix **A** in 2-D original MFTD. Many eigenvalues in 2-D original MFTD have large positive real part, which correspond to unstable modes in the numerical system.

In Fig. 5.7, we further show the MR of each dissembled patch when using proposed 2-D CS-MFTD or 2-D original MFTD. In original MFTD, a few dissembled patches have bad stability. In contrast, with CS-MFTD, every patch has controlled stability. Following Theorem 5.2.5, we can prove that 2-D CS-MFTD in the entire mesh is stable.

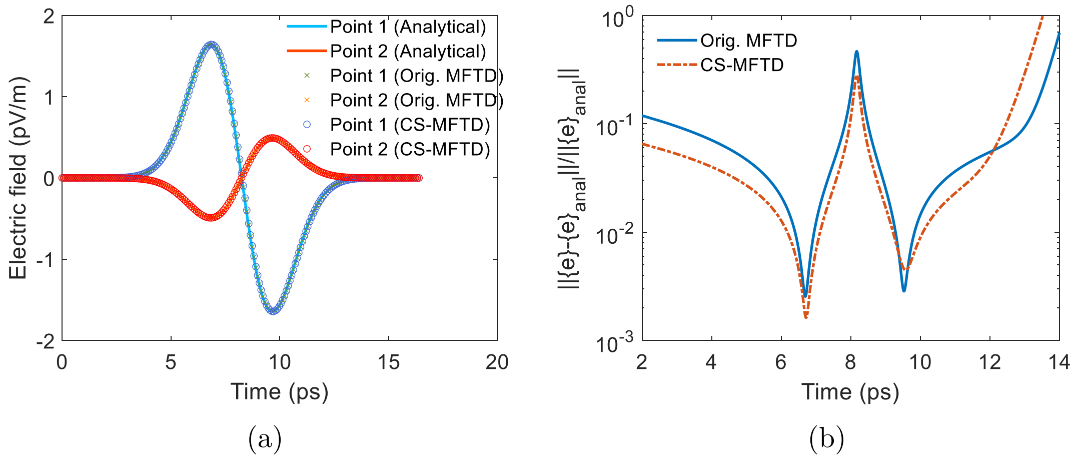


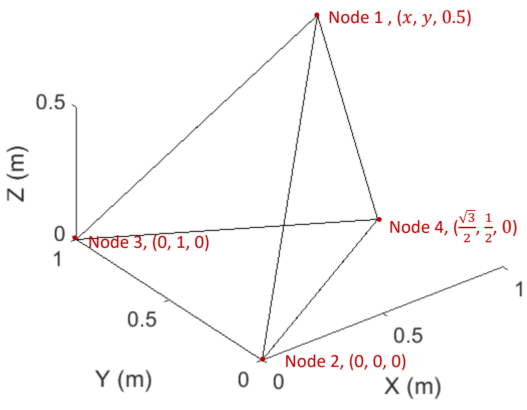
Figure 5.8. Accuracy of the proposed CS-MFTD in 2-D cavity example. (a) Simulated two electric fields in comparison with analytical results. (b) Entire **E** field solution error as a function of time.

To investigate the accuracy, we study a free-space wave propagation problem similar to the one in the ring mesh, except that $\tau = 2.0 \times 10^{-12}$ s. Both original MFTD and CS-MFTD use the same time step $\Delta t = 1.0 \times 10^{-14}$ s. We sample two points and plot their electric fields in Fig. 5.8 (a). The relative error of the whole solution vector is shown in Fig. 5.8 (b).

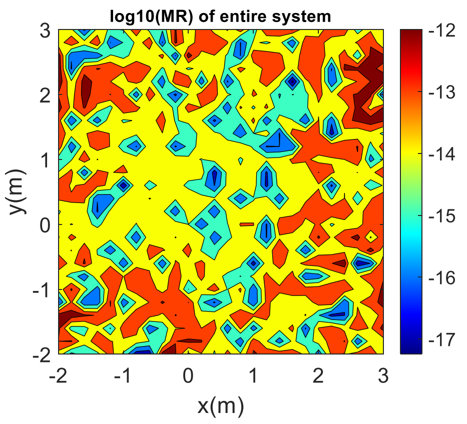
It can be seen clearly that the electric fields solved from both original MFTD and CS-MFTD have an excellent agreement with analytical results.

5.5 Numerical Results of 3-D CS-MFTD Method

5.5.1 3-D Single Tetrahedron of Arbitrary Shape



(a)



(b)

Figure 5.9. Single tetrahedron. (a) Node positions of this tetrahedron. (b) MR of the entire system matrix **A** vs. different x-y locations of node 1.

Here, we apply the new 3-D CS-MFTD to a single tetrahedron as shown in Fig. 5.9 (a). To test the robustness of the new MFTD method for different shapes of single tetrahedron, we set node 2, 3, and 4 at fixed locations as shown in Fig. 5.9 (a), then change the x-y coordinates of node 1. Both x-y coordinates of node 1 are sampled from the range of -2 m to 3 m, with a sampling interval of 0.2 m. The scaling factor of fixed **H** loop size is 0.01. For each location of node 1 representing a new tetrahedron structure, its system matrix **A** is generated to retrieve the MR following Algorithm 5. Fig. 5.9 (b) shows the MR of entire system matrix when node 1 are at different locations. The data show that the eigenvalues of the entire system matrix are almost pure imaginary, therefore, the entire system is always stable in this new 3-D CS-MFTD method. In Fig. 5.10, we further show the MR of each dissembled patch. The left blank area in white color in Fig. 5.10 means MR is less than 10^{-20}. Compared to the machine precision of 10^{-15} level, we can safely say each

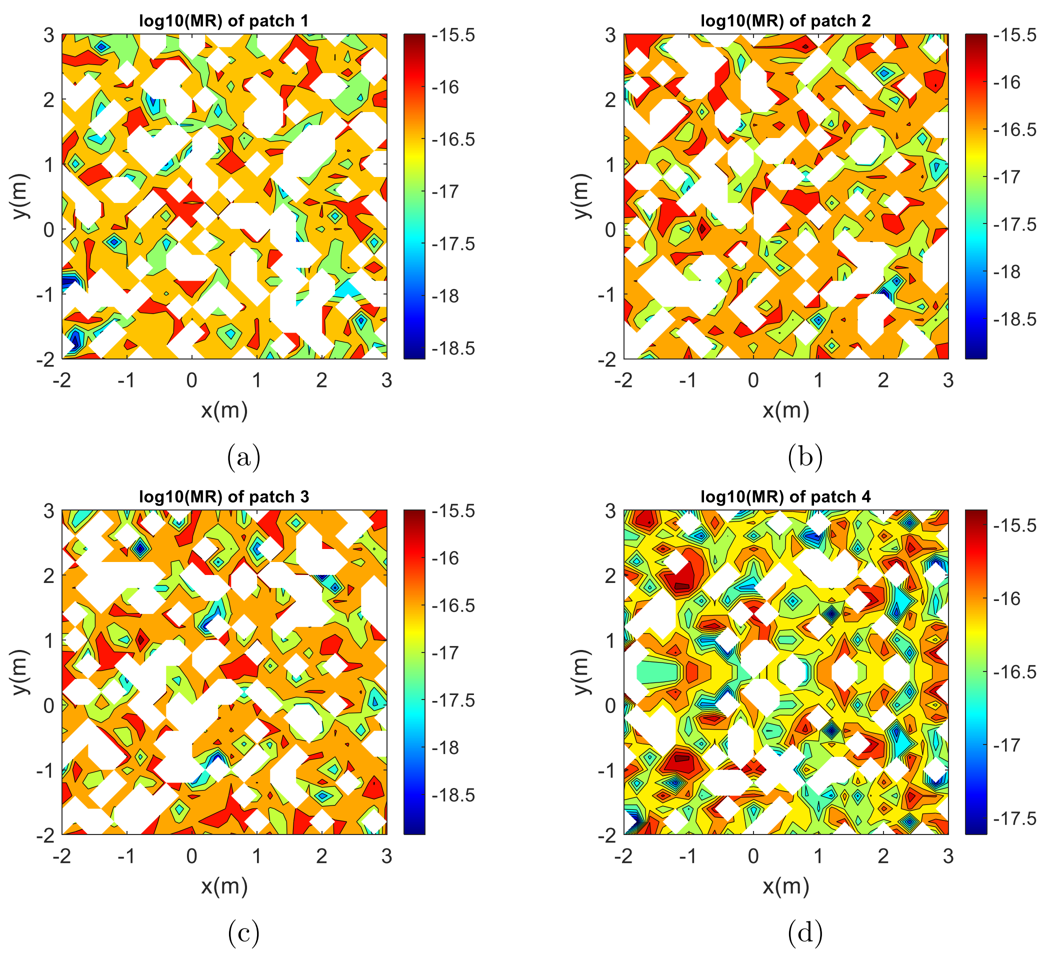


Figure 5.10. MR of the local system matrix in each dissembled patch. (a) Patch 1 (node 1-2-3). (b) Patch 2 (node 1-2-4). (c) Patch 3 (node 1-3-4). (d) Patch 4 (node 2-3-4).

patch and corresponding dissembled subsystem has pure imaginary eigenvalues, therefore has guaranteed stability.

5.5.2 A 3-D Box Discretized into Tetrahedral Mesh

This example is a 3-D box discretized into tetrahedral elements shown in Fig. 5.11 (a). The total size is 1.0 m in x-direction, 0.5 m in y-direction, and 0.75 m in z-direction. The discretization results in 113 nodes, 544 edges, 782 patches, and 350 tetrahedron elements. The scaling factor of fixed $\mathbf{H}$ loop size is 0.01. The resultant entire system matrix $\mathbf{A}$ has

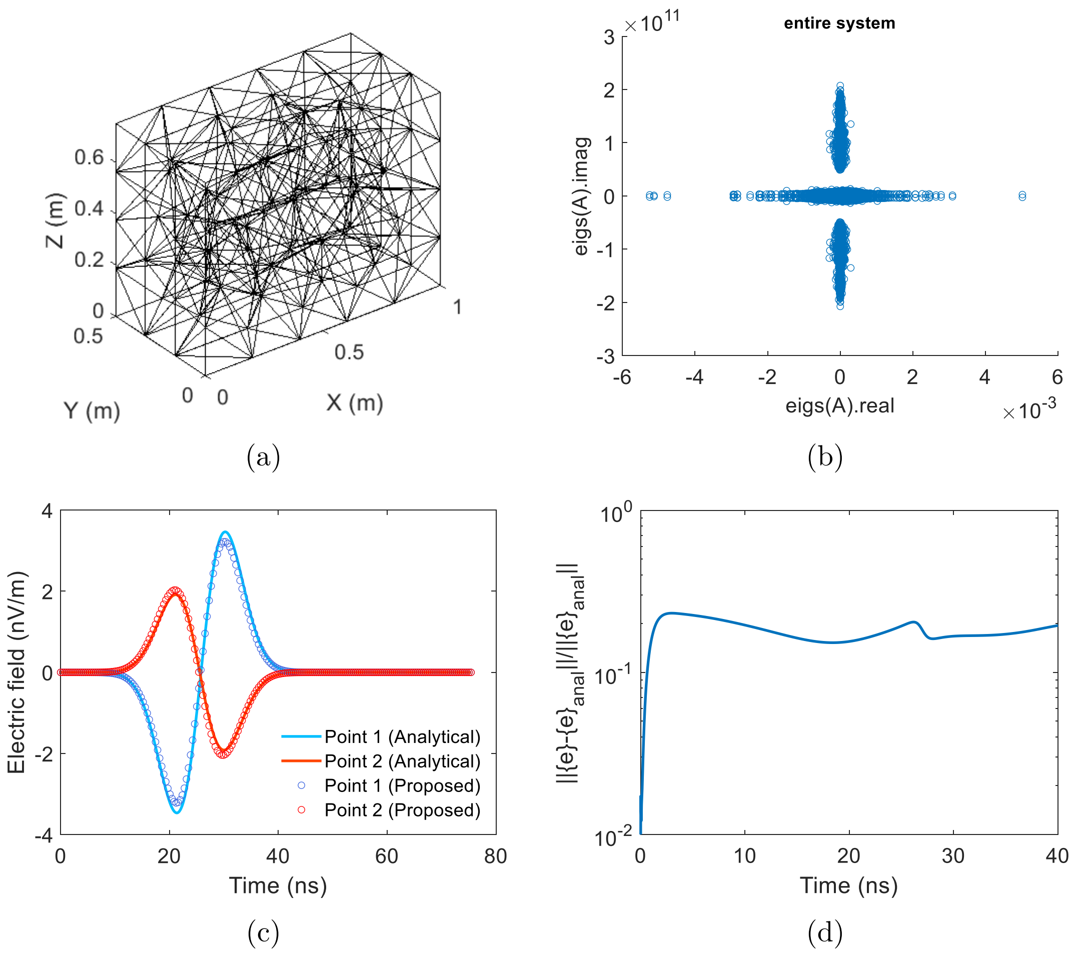


Figure 5.11. 3-D parallel plate. (a) Tetrahedron mesh of a 3-D parallel plate. (b) Eigenvalues of entire system matrix **A**. (c) Simulated two electric fields in comparison with analytical results. (d) Entire **E** field solution error as a function of time.

3,992 nonzero eigenvalues, which are numerically solved and plotted in Fig. 5.11 (b). Clearly, all eigenvalues of **A** are almost pure imaginary, therefore, the entire system is stable.

We also set up a free-space wave propagation problem in the given mesh to validate the accuracy of the proposed method against analytical results. The incident **E** has the same form as that of the previous example, but with $\tau = 6.28$ ns in accordance with the new 3-D structure's dimension. The time step used in the proposed method is $\Delta t = 2.60$ ps, which is determined from (5.19). In Fig. 5.11 (c), we plot the electric fields of the 1st (edge **E**) and the 602-th (face **E**) entry from the unknown $\{e\}$ vector, and compare them with analytical solutions. Good agreement can be observed. We also plot the entire solution error shown in Fig. 5.11 (d) versus time. It is evident that the proposed method is not just accurate at certain points, but accurate at all points in the computational domain for all time instants simulated.

5.5.3 A 3-D Sphere Discretized into Tetrahedral Mesh

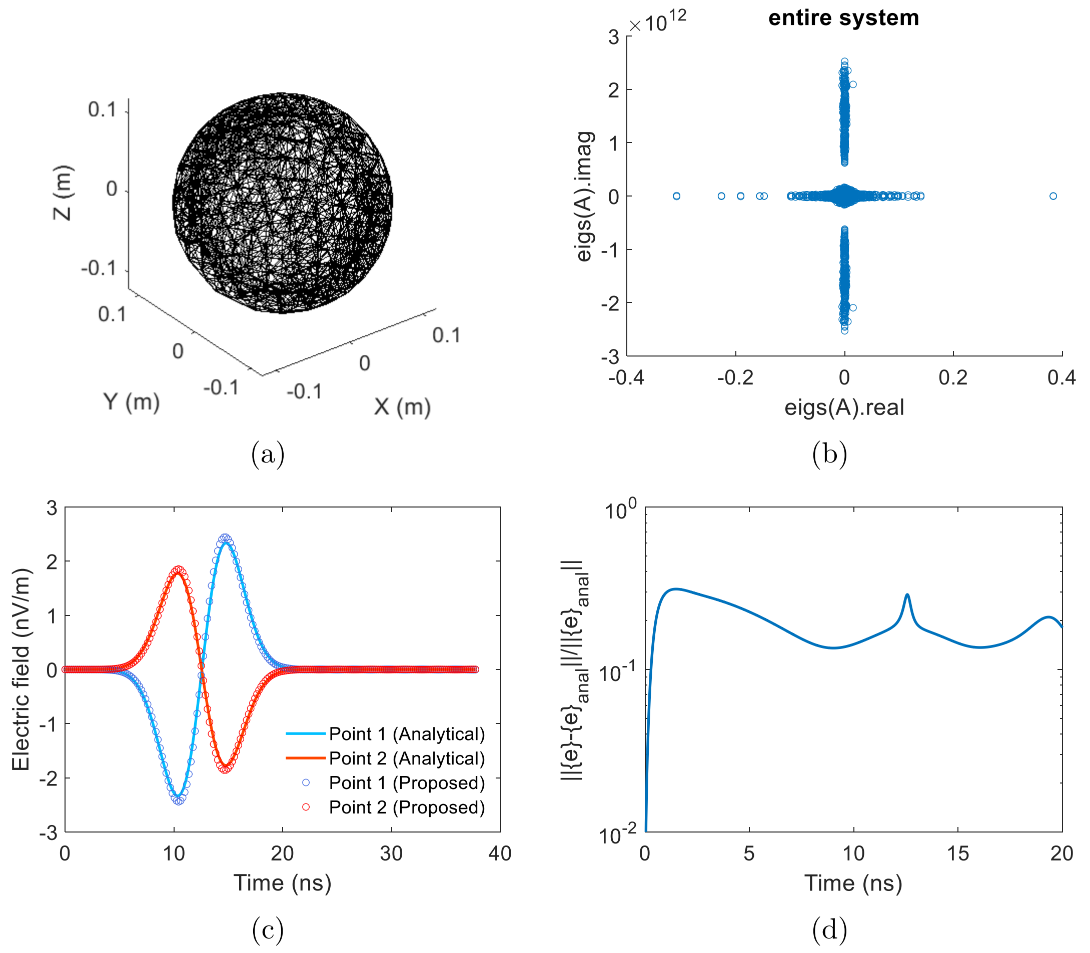


Figure 5.12. 3-D sphere. (a) Tetrahedron mesh of a 3-D sphere. (b) Eigenvalues of entire system matrix **A**. (c) Simulated two electric fields in comparison with analytical results. (d) Entire **E** field solution error as a function of time.

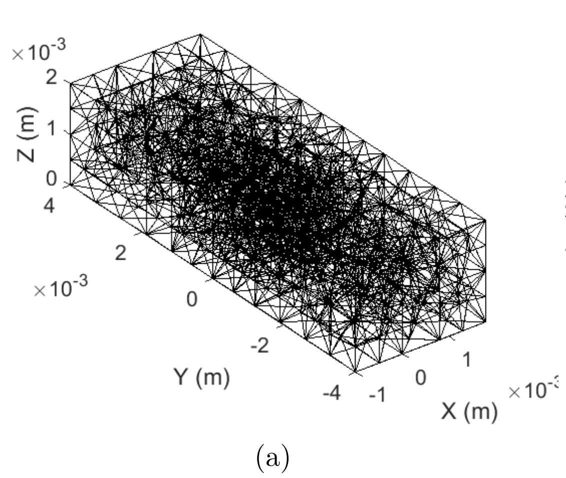
This example is a 3-D sphere discretized into tetrahedral elements shown in Fig. 5.12 (a). The radius is 0.1 m. The discretization results in 629 nodes, 3,183 edges, 4,543 patches, and 1,987 tetrahedron elements. The scaling factor of fixed **H** loop size is 0.01. The resultant entire system matrix **A** has 16,458 nonzero eigenvalues, which are numerically solved and plotted in Fig. 5.12 (b). Clearly, all eigenvalues of $\mathbf{A}$ are almost pure imaginary, therefore, the entire system is stable.

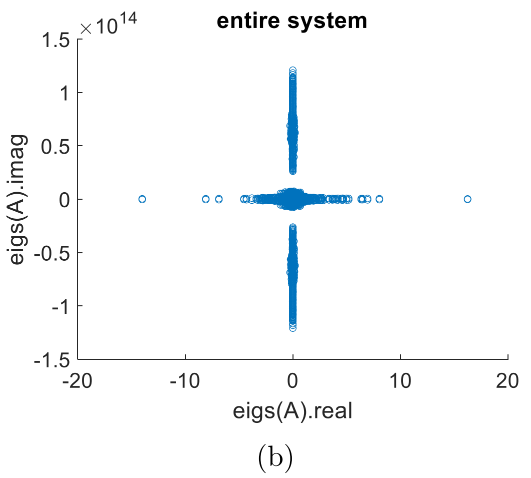
We also set up a free-space wave propagation problem in the given mesh to validate the accuracy of the proposed method against analytical results. The incident $\mathbf{E}$ has the same form as that of the previous example, but with $\tau = 3.14$ ns in accordance with the new 3-D structure's dimension. The time step used in the proposed method is $\Delta t = 0.23$ ps, which is determined from (5.19). In Fig. 5.12 (c), we plot the electric fields of the 1st (edge $\mathbf{E}$) and the 2,955-th (face $\mathbf{E}$) entry from the unknown $\{e\}$ vector, and compare them with analytical solutions. Good agreement can be observed. We also plot the entire solution error shown in Fig. 5.12 (d) versus time. It is evident that the proposed method is not just accurate at certain points, but accurate at all points in the computational domain for all time instants simulated. The center peak in Fig. 5.12 (d) is due to the comparison with close to zero fields.

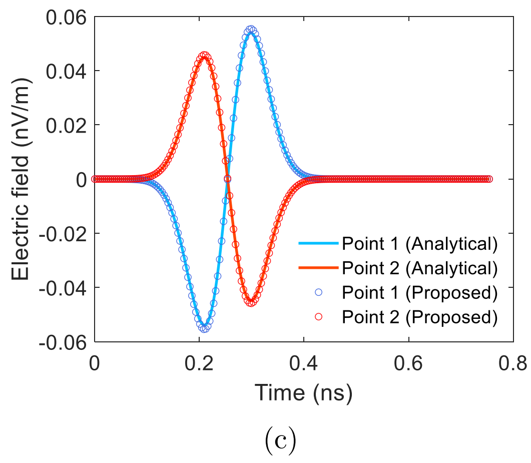
5.5.4 A 3-D Stripline Discretized into Tetrahedral Mesh

This example is a 3-D stripline discretized into tetrahedral elements shown in Fig. 5.13 (a). The total size is 2.795 μm in x-direction, 8.0 μm in y-direction, and 2.0 μm in z-direction. The discretization results in 702 nodes, 4,070 edges, 6,411 patches, and 3,042 tetrahedron elements. The scaling factor of fixed $\mathbf{H}$ loop size is 0.01. The resultant entire system matrix $\mathbf{A}$ has 28,460 nonzero eigenvalues, which are numerically solved and plotted in Fig. 5.13 (b). Clearly, all eigenvalues of $\mathbf{A}$ are almost pure imaginary, therefore, the entire system is stable.

We also set up a free-space wave propagation problem in the given mesh to validate the accuracy of the proposed method against analytical results. The incident $\mathbf{E}$ has the same form as that of the previous example, but with $\tau = 62.8$ ps in accordance with the new 3-D structure's dimension. The time step used in the proposed method is $\Delta t = 5.56$ fs, which is determined from (5.19). In Fig. 5.13 (c), we plot the electric fields of the 1st (edge $\mathbf{E}$) and the 6,184-th (face $\mathbf{E}$) entry from the unknown $\{e\}$ vector, and compare them with analytical solutions. Good agreement can be observed. We also plot the entire solution error shown







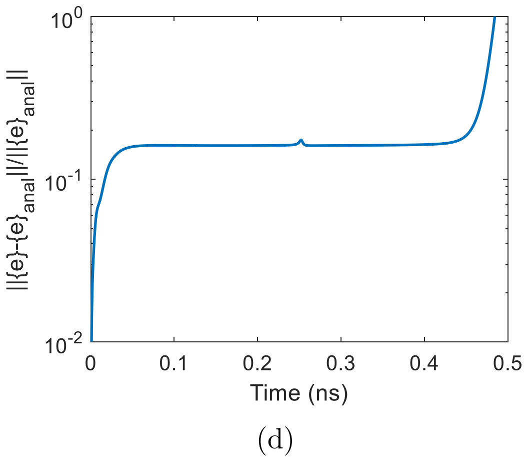


Figure 5.13. 3-D stripline. (a) Tetrahedron mesh of a 3-D stripline. (b) Eigenvalues of entire system matrix **A**. (c) Simulated two electric fields in comparison with analytical results. (d) Entire **E** field solution error as a function of time.

in Fig. 5.13 (d) versus time. It is evident that the proposed method is not just accurate at certain points, but accurate at all points in the computational domain for all time instants simulated. The later rising error in Fig. 5.13 (d) is due to the comparison with close to zero fields.

6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In this work, we develop multiphysics and large-scale modeling and simulation methods for advanced integrated circuit design as follows:

- First, we study the high-frequency performance of Cu-Graphene hybrid interconnects with a proposed multiphysics model. We develop a multiphysics-based model and an efficient simulation algorithm to co-simulate directly in time domain Maxwell's equations, equations characterizing graphene materials, and Boltzmann equation from direct current (DC) to high frequencies. To enable the simulation of nano-interconnects within a feasible run time, the entire numerical system is further made unconditionally stable in time marching. We show the multiphysics modeling and simulation algorithm for analyzing Cu-G interconnects, prove the time-domain stability of the coupled simulation, validate the proposed work against measured data, and also apply it to predict the crosstalk and propagation delay of Cu-G interconnects.

- Second, we study the difference between the first-principles based method and commonly used simplified models such as the Drude-model based approach for analyzing general on-chip Cu-graphene hybrid systems, from both theoretical and numerical perspectives. To do so, we first develop a Drude model based simulation algorithm, including the derivation of the Drude model from the Boltzmann transport equation, the numerical representation of the Drude model, and an efficient algorithm for simulating the Drude model in conjunction with the FDTD to analyze a Cu-graphene interconnect. We then compare it with the first-principles based multiphysics model and the resulting simulation algorithm both theoretically and numerically through extensive numerical experiments performed on the Cu-G nano-interconnects. We find that the first-principles based analysis is necessary to capture the physical process happening in a Cu-G interconnect at microwave frequencies and in the sub-nm regime.

- Third, to address the large-scale simulation challenge, we develop a new parallelized domain decomposition (DD) algorithm for solving Maxwell's equations that minimizes the communication between subdomains, while having a fast convergence of the global solution. In this algorithm, unlike prevailing domain decomposition methods which treat the interface field as a whole and use it to build a transmission condition between subdomains, we split the interface field into multiple components, and let each component be solved from one subdomain. In this way, we transform the original coupled system to p decoupled subsystems to solve iteratively with guaranteed convergence, where p is the number of subdomains. Only one addition (communication) of the interface unknowns needs to be performed after the computation in each subdomain is finished at each time step. More important, the algorithm has a guaranteed fast convergence and permits the use of a large time step irrespective of space step. It possesses the advantages of both the direct solver (used to solve the numerical system in each subdomain) and the iterative solver (used to capture the coupling between subdomains, whose convergence is made guaranteed). Numerical experiments on large-scale on-chip and package layout analysis have demonstrated the capability of the new DD parallelization algorithm.

- In the last part of the thesis, to tackle the challenge of efficient simulation of irregular structures, we develop a method for the stability analysis and control of unsymmetrical numerical systems in time domain. In our method, we reduce the stability analysis of a large system into the stability analysis of dissembled elements, therefore provides a feasible way to control the stability of large-scale systems regardless of whether the system is symmetrical or unsymmetrical. We then apply the proposed method to prove and control the stability of an unsymmetrical matrix-free method that solves Maxwell's equations in general unstructured meshes while not requiring a matrix solution.

6.2 Future Work

The future work of this research includes:

- Balance stability and accuracy in 3-D MFTD. Now, the stability can be controlled. For 2-D MFTD, our developed 2-D CS-MFTD can control the stability while not sacrificing the accuracy. However, in a 3-D mesh, the accuracy of our developed 3-D CS-MFTD is worse than the original 3-D MFTD. Therefore, new approaches to approximate $\mathbf{S}_h$ might be studied to balance stability and accuracy in the 3-D MFTD.

- Include more physics effects, such as the intraband transition in graphene and the surface scattering at the Cu-graphene interface to further enrich the multiphysics model and enhance the prediction power.

REFERENCES

[1] R. C. Munoz and C. Arenas, "Size effects and charge transport in metals: Quantum theory of the resistivity of nanometric metallic structures arising from electron scattering by grain boundaries and by rough surfaces," *Appl. Phys. Rev.*, vol. 4, no. 1, p. 011 102, 2017. DOI: 10.1063/1.4974032.

[2] D. Josell, S. H. Brongersma, and Z. Tőkei, "Size-dependent resistivity in nanoscale interconnects," *Annu. Rev. Mater. Res.*, vol. 39, no. 1, pp. 231–254, 2009. DOI: 10.1146/annurev-matsci-082908-145415.

[3] R. Mehta, S. Chugh, and Z. Chen, "Enhanced electrical and thermal conduction in graphene-encapsulated copper nanowires," *Nano Lett.*, vol. 15, no. 3, pp. 2024–2030, 2015. DOI: 10.1021/nl504889t.

[4] A. Naeemi and J. D. Meindl, "Compact physics-based circuit models for graphene nanoribbon interconnects," *IEEE Trans. Electron Devices*, vol. 56, no. 9, pp. 1822–1833, 2009, ISSN: 0018-9383.

[5] V. P. Gusynin, S. G. Sharapov, and J. P. Carbotte, "Magneto-optical conductivity in graphene," *J. Phys. Condens. Matter*, vol. 19, no. 2, p. 026 222, Dec. 2006. DOI: 10.1088/0953-8984/19/2/026222.

[6] G. W. Hanson, "Dyadic green's functions and guided surface waves for a surface conductivity model of graphene," *J. Appl. Phys.*, vol. 103, no. 6, p. 064 302, 2008, ISSN: 0021-8979. DOI: 10.1063/1.2891452.

[7] A. G. D. Aloia, W. Zhao, G. Wang, and W. Yin, "Near-field radiated from carbon nanotube and graphene-based nanointerconnects," *IEEE Trans. Electromagn. Compat.*, vol. 59, no. 2, pp. 646–653, 2017, ISSN: 0018-9375. DOI: 10.1109/TEMC.2016.2639319.

[8] V. Nayyeri, M. Soleimani, and O. M. Ramahi, "Modeling graphene in the finite-difference time-domain method using a surface boundary condition," *IEEE Trans. Antennas Propag.*, vol. 61, no. 8, pp. 4176–4182, 2013, ISSN: 0018-926X. DOI: 10.1109/TAP.2013.2260517.

[9] R. M. S. d. Oliveira, N. R. N. M. Rodrigues, and V. Dmitriev, "Fdtd formulation for graphene modeling based on piecewise linear recursive convolution and thin material sheets techniques," *IEEE Antennas Wireless Propag. Lett.*, vol. 14, pp. 767–770, 2015, ISSN: 1536-1225. DOI: 10.1109/LAWP.2014.2378174.

[10] A. Vakil and N. Engheta, "Transformation optics using graphene," *Science*, vol. 332, no. 6035, p. 1291, 2011.

[11] D. Sarkar, C. Xu, H. Li, and K. Banerjee, "High-frequency behavior of graphene-based interconnects-part i: Impedance modeling," *IEEE Trans. Electron Devices*, vol. 58, no. 3, pp. 843–852, Mar. 2011, ISSN: 0018-9383. DOI: 10.1109/TED.2010.2102031.

[12] R. Wang, X.-G. Ren, Z. Yan, L.-J. Jiang, W. E. I. Sha, and G.-C. Shan, "Graphene based functional devices: A short review," *Front. Phys.*, vol. 14, no. 1, p. 13 603, Oct. 2018, ISSN: 2095-0470. DOI: 10.1007/s11467-018-0859-y.

[13] B. Sensale-Rodriguez, R. Yan, M. M. Kelly, T. Fang, K. Tahy, W. S. Hwang, D. Jena, L. Liu, and H. G. Xing, "Broadband graphene terahertz modulators enabled by intraband transitions," *Nat. Commun.*, vol. 3, p. 780, 2012.

[14] J. Horng, C.-F. Chen, B. Geng, C. Girit, Y. Zhang, Z. Hao, H. A. Bechtel, M. Martin, A. Zettl, M. F. Crommie, Y. R. Shen, and F. Wang, "Drude conductivity of dirac fermions in graphene," *Phys. Rev. B*, vol. 83, p. 165 113, 16 Apr. 2011. DOI: 10.1103/PhysRevB.83.165113.

[15] C. Berger, Z. Song, X. Li, X. Wu, N. Brown, C. Naud, D. Mayou, T. Li, J. Hass, A. N. Marchenkov, E. H. Conrad, P. N. First, and W. A. de Heer, "Electronic confinement and coherence in patterned epitaxial graphene," *Science*, vol. 312, no. 5777, pp. 1191–1196, 2006, ISSN: 0036-8075. DOI: 10.1126/science.1125925. eprint: http://science.sciencemag.org/content/312/5777/1191.full.pdf.

[16] E. H. Hwang and S. Das Sarma, "Single-particle relaxation time versus transport scattering time in a two-dimensional graphene layer," *Phys. Rev. B*, vol. 77, p. 195 412, 19 May 2008. DOI: 10.1103/PhysRevB.77.195412.

[17] A. Toselli and O. Widlund, *Domain Decomposition Methods-Algorithms and Theory*. Springer Science & Business Media, 2006, vol. 34.

[18] T. Hemmi, F. Costen, S. Garcia, R. Himeno, H. Yokota, and M. Mustafa, "Efficient parallel LOD-FDTD method for Debye-dispersive media," *IEEE Trans. Antennas Propag.*, vol. 62, no. 3, pp. 1330–1338, Mar. 2014.

[19] M. Yi, Z. Qian, A. Aydiner, and M. Swaminathan, "Transient simulation of multiscale structures using the nonconformal domain decomposition Laguerre-FDTD method," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 5, no. 4, pp. 532–540, Apr. 2015.

[20] J. Yan and D. Jiao, "Time-domain method having a naturally diagonal mass matrix independent of element shape for general electromagnetic analysis—2-D formulations," *IEEE Trans. Antennas Propag.*, vol. 65, no. 3, pp. 1202–1214, Mar. 2017. DOI: 10.1109/TAP.2017.2653078.

[21] C. G. Kang, S. K. Lim, S. Lee, S. K. Lee, C. Cho, Y. G. Lee, H. J. Hwang, Y. Kim, H. J. Choi, S. H. Choe, M.-H. Ham, and B. H. Lee, "Effects of multi-layer graphene capping on Cu interconnects," *Nanotechnology*, vol. 24, no. 11, p. 115 707, 2013.

[22] P. Goli, H. Ning, X. Li, C. Y. Lu, K. S. Novoselov, and A. A. Balandin, "Thermal properties of graphene–copper–graphene heterogeneous films," *Nano Lett.*, vol. 14, no. 3, pp. 1497–1503, 2014. DOI: 10.1021/nl404719n.

[23] N. T. Cuong and S. Okada, "Suppression of conductivity deterioration of copper thin films by coating with atomic-layer materials," *Appl. Phys. Lett.*, vol. 110, no. 13, p. 131 601, 2017. DOI: 10.1063/1.4979038.

[24] S. Sun and D. Jiao, "Multiphysics modeling and simulation of 3-D cu-graphene hybrid nanointerconnects," *IEEE Trans. Microw. Theory Tech.*, vol. 68, no. 2, pp. 490–500, 2020. DOI: 10.1109/TMTT.2019.2955123.

[25] S. Sun and D. Jiao, "First-principles based multiphysics modeling and simulation of on-chip Cu-graphene hybrid nano-interconnects in comparison with simplified model based analysis," *IEEE J. Multiscale Multiphys. Comput. Tech.*, vol. 4, pp. 374–382, Dec. 2019. DOI: 10.1109/JMMCT.2020.2964655.

[26] C. Kittel, *Introduction to Solid State Physics*, 8th ed. Wiley, 2005, pp. 656–661.

[27] H. Peng, N. B. M. Schröter, J. Yin, H. Wang, T.-F. Chung, H. Yang, S. Ekahana, Z. Liu, J. Jiang, L. Yang, T. Zhang, C. Chen, H. Ni, A. Barinov, Y. P. Chen, Z. Liu, H. Peng, and Y. Chen, "Substrate doping effect and unusually large angle van hove singularity evolution in twisted Bi- and multilayer graphene," *Adv. Mater.*, vol. 29, no. 27, p. 1 606 741, 2017, ISSN: 1521-4095. DOI: 10.1002/adma.201606741.

[28] M. Jablan, H. Buljan, and M. Solja či ć, "Plasmonics in graphene at infrared frequencies," *Phys. Rev. B*, vol. 80, p. 245 435, 24 Dec. 2009. DOI: 10.1103/PhysRevB.80.245435.

[29] J. Yan and D. Jiao, "Fast explicit and unconditionally stable FDTD method for electromagnetic analysis," *IEEE Trans. Microw. Theory Tech.*, vol. 65, no. 8, pp. 2698–2710, Aug. 2017. DOI: 10.1109/TMTT.2017.2686862.

[30] C. Jungemann and B. Meinerzhagen, "Analysis of the stochastic error of stationary monte carlo device simulations," *IEEE Trans. Electron Devices*, vol. 48, no. 5, pp. 985–992, May 2001, ISSN: 0018-9383. DOI: 10.1109/16.918247.

[31] P. W. Rambo and J. Denavit, "Time stability of monte carlo device simulation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 11, pp. 1734–1741, Nov. 1993, ISSN: 0278-0070. DOI: 10.1109/43.248084.

[32] L. Varani, L. Reggiani, T. Kuhn, T. Gonzalez, and D. Pardo, "Microscopic simulation of electronic noise in semiconductor materials and devices," *IEEE Trans. Electron Devices*, vol. 41, no. 11, pp. 1916–1925, Nov. 1994, ISSN: 0018-9383. DOI: 10.1109/16.333807.

[33] S.-M. Hong, A.-T. Pham, and C. Jungemann, *Deterministic Solvers for the Boltzmann Transport Equation*. Springer Science & Business Media, 2011.

[34] K. Zhao, S. Hong, C. Jungemann, and R. Han, "Stable implementation of a deterministic multi-subband boltzmann solver for silicon double-gate nmosfets," in *Int. Conf. Simulation Semiconductor Processes Devices*, Sep. 2010, pp. 303–306. DOI: 10.1109/SISPAD.2010.5604500.

[35] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007, p. 342, ISBN: 978-0-89871-629-0. DOI: doi:10.1137/1.9780898717839.

[36] J. G. Maloney and G. S. Smith, "The efficient modeling of thin material sheets in the finite-difference time-domain (fdtd) method," *IEEE Trans. Antennas Propag.*, vol. 40, no. 3, pp. 323–330, 1992, ISSN: 0018-926X. DOI: 10.1109/8.135475.

[37] M. J. Kobrinsky, S. Chakravarty, D. Jiao, M. C. Harmes, S. List, and M. Mazumder, "Experimental validation of crosstalk simulations for on-chip interconnects using S-parameters," *IEEE Trans. Adv. Packag.*, vol. 28, no. 1, pp. 57–62, Feb. 2005. DOI: 10.1109/TADVP.2004.841672.

[38] S. Sun and D. Jiao, "Multiphysics simulation of high-speed graphene-based interconnects in time domain," in *Proc. IEEE Int. Symp. Antennas Propag.*, Jul. 2018, pp. 1169–1170. DOI: 10.1109/APUSNCURSINRSM.2018.8608859.

[39] Feng Xu and Wei Hong, "Domain decomposition FDTD algorithm for the analysis of a new type of E-plane sectorial horn with aperture field distribution optimization," *IEEE Trans. Antennas Propag.*, vol. 52, no. 2, pp. 426–434, Feb. 2004.

[40] Z. Lai, J. Kiang, and R. Mittra, "A domain decomposition finite difference time domain (FDTD) method for scattering problem from very large rough surfaces," *IEEE Trans. Antennas Propag.*, vol. 63, no. 10, pp. 4468–4476, Oct. 2015.

[41] S. Yang, Z. Chen, Y. Yu, and W. Yin, "An unconditionally stable one-step arbitrary-order leapfrog ADI-FDTD method and its numerical properties," *IEEE Trans. Antennas Propag.*, vol. 60, no. 4, pp. 1995–2003, Apr. 2012. DOI: 10.1109/TAP.2012.2186249.

[42] J. Shibayama, M. Muraki, J. Yamauchi, and H. Nakano, “Efficient implicit FDTD algorithm based on locally one-dimensional scheme,” *Electron. Lett.*, vol. 41, no. 19, pp. 1046–1047, Sep. 2005. DOI: 10.1049/el:20052381.

[43] Y. Duan, B. Chen, D. Fang, and B. Zhou, “Efficient implementation for 3-D Laguerre-based finite-difference time-domain method,” *IEEE Trans. Microw. Theory Tech.*, vol. 59, no. 1, pp. 56–64, Jan. 2011. DOI: 10.1109/TMTT.2010.2091206.

[44] G. Sun and C. W. Trueman, “Approximate Crank-Nicolson schemes for the 2-D finite-difference time-domain method for TE$_z$ waves,” *IEEE Trans. Antennas Propag.*, vol. 52, no. 11, pp. 2963–2972, Nov. 2004. DOI: 10.1109/TAP.2004.835142.

[45] H. Bao and R. Chen, “An efficient domain decomposition parallel scheme for leapfrog ADI-FDTD method,” *IEEE Trans. Antennas Propag.*, vol. 65, no. 3, pp. 1490–1494, Mar. 2017.

[46] F. Zheng and Z. Chen, “Numerical dispersion analysis of the unconditionally stable 3-D ADI-FDTD method,” *IEEE Trans. Microw. Theory Tech.*, vol. 49, no. 5, pp. 1006–1009, May 2001. DOI: 10.1109/22.920165.

[47] I. Ahmed, E. Chua, E. Li, and Z. Chen, “Development of the three-dimensional unconditionally stable LOD-FDTD method,” *IEEE Trans. Antennas Propag.*, vol. 56, no. 11, pp. 3596–3600, Nov. 2008. DOI: 10.1109/TAP.2008.2005544.

[48] G. He, W. Shao, X. Wang, and B. Wang, “An efficient domain decomposition Laguerre-FDTD method for two-dimensional scattering problems,” *IEEE Trans. Antennas Propag.*, vol. 61, no. 5, pp. 2639–2645, May 2013. DOI: 10.1109/TAP.2013.2242836.

[49] X. Wei, W. Shao, and H. Ou, “Domain decomposition CN-FDTD method for analyzing dispersive metallic gratings,” *IEEE Photon. J.*, vol. 9, no. 4, pp. 1–18, Aug. 2017. DOI: 10.1109/JPHOT.2017.2722459.

[50] Y. Li and J. M. Jin, “A vector dual-primal finite element tearing and interconnecting method for solving 3-D large-scale electromagnetic problems,” *IEEE Trans. Antennas Propag.*, vol. 54, no. 10, pp. 3000–3009, Oct. 2006. DOI: 10.1109/TAP.2006.882191.

[51] D. Jiao, S. Chakravarty, and C. Dai, “A layered finite element method for electromagnetic analysis of large-scale high-frequency integrated circuits,” *IEEE Trans. Antennas Propag.*, vol. 55, no. 2, pp. 422–432, Feb. 2007.

[52] Y. Shao, Z. Peng, and J. Lee, “Signal integrity analysis of high-speed interconnects by using nonconformal domain decomposition method,” *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 2, no. 1, pp. 122–130, Jan. 2012.

[53] S. Wang, Y. Shao, and Z. Peng, "A parallel-in-space-and-time method for transient electromagnetic problems," *IEEE Trans. Antennas Propag.*, vol. 67, no. 6, pp. 3961–3973, Jun. 2019.

[54] M. Discacciati, P. Gervasio, and A. Quarteroni, "The interface control domain decomposition (ICDD) method for elliptic problems," *SIAM J. Control Optim.*, vol. 51, no. 5, pp. 3434–3458, Sep. 2013.

[55] M. J. Gander, "Schwarz methods over the course of time," *Electron. Trans. Numer. Anal.*, vol. 31, pp. 228–255, 2008.

[56] *The OpenROAD Project*, Jun. 2021. [Online]. Available: https://theopenroadproject.org/.

[57] L. T. Clark, V. Vashishtha, D. M. Harris, S. Dietrich, and Z. Wang, "Design flows and collateral for the ASAP7 7nm FinFET predictive process design kit," in *Proc. IEEE Int. Conf. Microelectron. Syst. Edu. (MSE)*, May 2017, pp. 1–4. DOI: 10.1109/MSE.2017.7945071.

[58] X. Jia, F. Yang, X. Liu, M. Li, and S. Xu, "Fast nonuniform metasurface analysis in FDTD using surface susceptibility model," *IEEE Trans. Antennas Propag.*, vol. 68, no. 10, pp. 7121–7130, Oct. 2020. DOI: 10.1109/TAP.2019.2957317.

[59] H. B. Wang, Y. J. Cheng, and Z. N. Chen, "Wideband and wide-angle single-layered-substrate linear-to-circular polarization metasurface converter," *IEEE Trans. Antennas Propag.*, vol. 68, no. 2, pp. 1186–1191, Feb. 2020. DOI: 10.1109/TAP.2019.2938683.

[60] T. Ohtani, Y. Kanai, and J. B. Cole, "A stability improvement technique using PML condition for the three-dimensional nonuniform mesh nonstandard FDTD method," *IEEE Trans. Magn.*, vol. 49, no. 5, pp. 1569–1572, May 2013. DOI: 10.1109/TMAG.2013.2238613.

[61] J. Yan and D. Jiao, "An unsymmetric FDTD subgridding algorithm with unconditional stability," *IEEE Trans. Antennas Propag.*, vol. 66, no. 8, pp. 4137–4150, May 2018. DOI: 10.1109/TAP.2018.2835561.

[62] K. Zeng and D. Jiao, "Symmetric positive semi-definite FDTD subgridding algorithms in both space and time for accurate analysis of inhomogeneous problems," *IEEE Trans. Antennas Propag.*, vol. 68, no. 4, pp. 3047–3059, Jan. 2020. DOI: 10.1109/TAP.2020.2964943.

[63] A. A. Ijjeh, M. Cueille, J.-L. Dubard, and M. M. Ney, "Dispersion and stability analysis for TLM unstructured block meshing," *IEEE Trans. Microw. Theory Tech.*, vol. 69, no. 10, pp. 4352–4365, Oct. 2021. DOI: 10.1109/TMTT.2021.3093417.

[64] J. Yan and D. Jiao, "Matrix-free time-domain method for general electromagnetic analysis in 3-D unstructured meshes—modified-basis formulation," *IEEE Trans. Microw. Theory Tech.*, vol. 64, no. 8, pp. 2371–2382, Jul. 2016. DOI: 10.1109/TMTT.2016.2584047.

[65] K. Zeng and D. Jiao, "Explicit matrix-free time-domain method in unstructured meshes and its application to stable simulation of general unsymmetrical systems," *IEEE Trans. Microw. Theory Tech.*, vol. 67, no. 12, pp. 4821–4832, Dec. 2019. DOI: 10.1109/TMTT.2019.2951669.

[66] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*. SIAM, 2004.

[67] M. Moradi, V. Nayyeri, and O. M. Ramahi, "An unconditionally stable single-field finite-difference time-domain method for the solution of Maxwell equations in three dimensions," *IEEE Trans. Antennas Propag.*, vol. 68, no. 5, pp. 3859–3868, May 2020. DOI: 10.1109/TAP.2020.2975675.

[68] D. Jiao and J.-M. Jin, "A general approach for the stability analysis of the time-domain finite-element method for electromagnetic simulations," *IEEE Trans. Antennas Propag.*, vol. 50, no. 11, pp. 1624–1632, Nov. 2002. DOI: 10.1109/TAP.2002.803965.

[69] R. Bellman, *Stability Theory of Differential Equations*. Dover Publications, 2013, ISBN: 9780486150130.

[70] P. J. Schmid, "Nonmodal stability theory," *Annu. Rev. Fluid Mech.*, vol. 39, no. 1, pp. 129–162, Jan. 2007. DOI: 10.1146/annurev.fluid.38.050304.092139.

[71] Y. Saad, *Numerical Methods for Large Eigenvalue Problems: Revised Edition*. SIAM, 2011.

[72] J. Yan and D. Jiao, "Accurate and stable matrix-free time-domain method in 3-D unstructured meshes for general electromagnetic analysis," *IEEE Trans. Microw. Theory Tech.*, vol. 63, no. 12, pp. 4201–4214, Dec. 2015, ISSN: 0018-9480.

[73] P.-O. Persson and G. Strang, "A simple mesh generator in MATLAB," *SIAM Rev.*, vol. 46, no. 2, pp. 329–345, 2004. DOI: 10.1137/S0036144503429121.

VITA

Shuzhan Sun received the B.S. degree in physics from the School of Special Class for the Gifted Young, University of Science and Technology of China (USTC), Hefei, China, in 2016. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering (with a minor M.S. degree in physics in 2018) in the On-Chip Electromagnetics Group, Purdue University, West Lafayette, IN, USA. Since Sept. 2021, Shuzhan works as a Lead Software Engineer in the Spectre circuit simulation team in Cadence Design Systems, San Jose, CA.

His current research focuses on simulating next-generation Cu-graphene hybrid nanointerconnects and developing novel electromagnetic algorithms for large-scale simulation.

Mr. Sun received the Best Student Paper Finalist Award at the IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (IEEE AP-S/URSI) in 2019, and 2021, respectively.