Interior Penalty Discontinuous Galerkin Finite Element Method for the Time-Domain Maxwell’s Equations

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Stylianos Dosopoulous, B.S., M.S.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2012

Dissertation Committee:
Professor Jin-Fa Lee, Advisor
Professor Fernando Teixeira
Professor Ashok Krishnamurthy
Abstract

This dissertation investigates a discontinuous Galerkin (DG) methodology to solve Maxwell’s equations in the time-domain. More specifically, we focus on an Interior Penalty (IP) approach to derive a DG formulation. In general, discontinuous Galerkin methods decompose the computational domain into a number of disjoint polyhedral (elements). For each polyhedron, we define local basis functions and approximate the fields as a linear combination of these basis functions. To ensure equivalence to the original problem the tangentially continuity of the electric and magnetic fields need to be enforced between polyhedra interfaces. This condition is applied in the weak sense by proper penalty terms on the variational formulation also known as numerical fluxes. Due to this way of coupling between adjacent polyhedra DG methods offer great flexibility and a nice set of properties such as, explicit time-marching, support for non-conformal meshes, freedom in the choice of basis functions and high efficiency in parallelization.

Here, we first introduce an Interior Penalty (IP) approach to derive a DG formulation and a physical interpretation of such an approach. This physical interpretation will provide a physical insight into the IP method and link important concepts like the duality pairing principle to a physical meaning. Furthermore, we discuss the
time discretization and stability condition aspects of our scheme. Moreover, to address the issue of very small time steps in multi-scale applications we employ a local time-stepping (LTS) strategy which can greatly reduce the solution time.

Secondly, we present an approach to incorporate a conformal Perfectly Matched Layer (PML) in our interior penalty discontinuous Galerkin time-domain (IPDGTDT) framework. From a practical point of view, a conformal PML is easier to model compared to a Cartesian PML and can reduce the buffer space between the structure and the truncation boundary, thus potentially reducing the number of unknowns.

Next, we discuss our approach to combine EM and circuit simulation into a single framework. We show how we incorporate passive lumped elements such as resistors, capacitors and inductors in the IPDGTDT framework. Practically, such a capability is useful since EM applications may often include lumped elements.

Following, we present our design of a scalable parallel implementation of IPDGTDT in order to exploit the inherit DG parallelism and significantly speed up computations. Our parallelization, is aimed to multi-core CPUs and/or graphics processor units (GPUs), for shared and/or distributed memory systems. In this way all of MPI/CPU, MPI/GPU and MPI/OpenMP configurations can be used.

Finally, we extend our IPDGTDT to further include the case of non-conformal meshes. Since, in DG methods the tangentially continuity of the fields is enforced in a weak sense, DG methods naturally support non-conformal meshes. In cases of complicated geometries where a conformal mesh is nearly impossible to get, the ability to handle non-conformal meshes is important. The original geometry in divided into smaller pieces and each piece is meshed independently. Thus, meshing requirements can be greatly relaxed and a final mesh can be obtained for computation.
This is dedicated to my parents, my sister and my friends
Acknowledgments

I would like to begin by expressing my gratitude to my advisor, Prof. Jin-Fa Lee for his continuous guidance and for being a mentor during my PhD studies. His zeal for the field of electromagnetics and science is inspiring for any young scientist. Additionally, I would like to thank my undergraduate advisor Prof. George Kyriacou for encouraging me to pursue a PhD degree.

Moreover, I would like to thank the dissertation committee members, Prof. Fernando Teixeira and Prof. Ashok Krisnamurthy whose useful and constructive comments on my dissertation contributed to produce a high quality document.

Furthermore, I thank all my colleagues in the Computational Science Group, Dr. Peng Zhen, Dr. Yang Shao, Josh Mahaffey, Matt Stephanson, Xiaochuan Wang, Jianguo Wei, Jue Wang and Yuanhong Zhao, for many fruitful discussions. Also, I would like to acknowledge Dr. Judith Gardiner from OSC, for a wonderful collaboration and all her support. Additionally, I would like to thank Ioannis Tzanidis, Georgios Tri-chopoulos, Erdinc Irci and Mustafa Kuloglou for being good friends during my time at ElectroScience Laboratory. Moreover, I thank ANSYS corp for supporting my research throughout my PhD studies.

Last but most importantly, I would like to thank my parents and sister. Their love, support and constant encouragement has played a crucial role throughout my PhD studies.
Vita

March 8, 1984 .......................... Born - Kozani, Greece

2006 .............................. B.S. Electrical and Computer Engineering, Democritus University Of Thrace, Xanthi, Greece

2011 .............................. M.S. Electrical and Computer Engineering, The Ohio State University, Columbus , OH, USA

2006-present ........................ Graduate Reasearch Associate,
The Ohio State University, Columbus , OH, USA

Publications

Research Publications


Fields of Study

Major Field: Electrical and Computer Engineering
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>Vita</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xii</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Contribution &amp; Organization</td>
<td>4</td>
</tr>
<tr>
<td>2. Interior Penalty Discontinuous Galerkin Time Domain Formulation</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Original Initial Value Problem (IVP)</td>
<td>9</td>
</tr>
<tr>
<td>2.2 IPDGT Formulation and Physical Interpretation</td>
<td>9</td>
</tr>
<tr>
<td>2.2.1 Trace operators and Notation</td>
<td>9</td>
</tr>
<tr>
<td>2.2.2 Formulation-Space Discretization</td>
<td>10</td>
</tr>
<tr>
<td>2.2.3 Formulation-Time Discretization</td>
<td>17</td>
</tr>
<tr>
<td>2.2.4 Stability Analysis and Local Time-Stepping (LTS)</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Convergence Properties of DGTD</td>
<td>23</td>
</tr>
<tr>
<td>2.4 Discussion</td>
<td>25</td>
</tr>
</tbody>
</table>
3. Perfectly Matched Layer for IPDGT

3.1 Perfectly Matched Layer

3.1.1 Formulation

3.1.2 DG Space and Time Discretization

3.1.3 Late Time Instability and PML performance

3.2 Numerical Results

3.2.1 Coated Sphere

3.2.2 Scattering from a F-16 aircraft

3.3 Discussion

4. Combined EM/Circuit Simulation in Interior
Penalty Discontinuous Galerkin Time-Domain Methods

4.1 IPDGT Formulation-Lumped Elements

4.1.1 Resistor

4.1.2 Capacitor

4.1.3 Inductor

4.1.4 Interior Port

4.2 Numerical Results

4.2.1 Interconnect device with Passive Lumped Elements

4.3 Discussion

5. Parallelization of Interior Penalty Discontinuous Galerkin Time-Domain Method

5.1 Introduction and Organization

5.2 CUDA Implementation

5.2.1 FEM Mesh to CUDA Grid Mapping

5.2.2 Finite Element to CUDA thread block Mapping

5.2.3 DOF to CUDA thread Mapping

5.2.4 Description of the leap-frog update kernels

5.3 MPI/GPU Parallelization

5.3.1 Combined MPI/GPU/LTS Strategy

5.3.2 MPI System Design

5.3.3 Partitioning and Communication Setup

5.4 Numerical Examples

5.4.1 Performance Analysis

5.4.2 3D SRR Metamaterial Cloaking Device

5.5 Discussion

6.1 DGTD Formulation-Support for non-conformal meshes ................. 83
6.2 MPI Parallelization ........................................................................ 87
    6.2.1 Balancing Strategy ................................................................. 88
6.3 Numerical examples ....................................................................... 91
    6.3.1 Simple Via Example ............................................................... 92
    6.3.2 IBM (IC) Model ..................................................................... 93
    6.3.3 Numerical Results ................................................................. 96
6.4 Discussion ..................................................................................... 101

Appendices ..................................................................................... 102

A. Appendix 1 ................................................................................... 102

    A.1 Stability Analysis by an Energy Technique ............................... 102
        A.1.1 Energy Conservation .......................................................... 102
        A.1.2 Definite Positivity ............................................................. 107

Bibliography .................................................................................... 116
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Physical Interpretation of the Duality Pairing</td>
<td>13</td>
</tr>
<tr>
<td>3.1 Element Partitioning by classes for the F16-aircraft</td>
<td>35</td>
</tr>
<tr>
<td>3.2 CPU time gain obtained with the local time-stepping versus the leap-frog scheme</td>
<td>35</td>
</tr>
<tr>
<td>4.1 Element Partitioning by classes for the Interconnect</td>
<td>52</td>
</tr>
<tr>
<td>4.2 CPU time gain obtained with the local time-stepping versus the leap-frog scheme</td>
<td>53</td>
</tr>
<tr>
<td>5.1 Average iteration time of one complete LTS update (4 classes) for the SRR cloaking example</td>
<td>78</td>
</tr>
<tr>
<td>6.1 Computational statistics for the IBM model.</td>
<td>99</td>
</tr>
<tr>
<td>6.2 Distribution of elements into the LTS classes for the IBM model.</td>
<td>99</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 (a)-(b) Geometrical illustration of the DG discretization.</td>
<td>12</td>
</tr>
<tr>
<td>2.2 Pictorial illustration of the LTS for a two class (class 0, class 1) examples. (a) E-Update (b) H-Update.</td>
<td>22</td>
</tr>
<tr>
<td>2.3 Convergence results using central and upwind flux. The red and blue dashed lines show the evolution of the computed error and the black and green solid lines represent the asymptotic behavior. (a) $</td>
<td></td>
</tr>
<tr>
<td>3.1 Late time instability in the PML region for a parallel plate waveguide example. The PML parameters where set to $m = 4$, $\sigma_{\text{max}} = 20$ and the PML is 4-layers thick ($\delta = 3.5m$). (a) Magnitude of the E-field without stabilization at time $T = 680\delta t$ (b) Magnitude of the E-field with stabilization at time $T = 680\delta t$.</td>
<td>30</td>
</tr>
<tr>
<td>3.2 Reflection coefficient for the parallel plate waveguide of Figure 3.3a with and without the stabilization technique. The stabilization, removes instability without significantly altering the PML properties. The PML was 4-layers thick.</td>
<td>31</td>
</tr>
<tr>
<td>3.3 (a) The red shaded region indicates the PML region and the gray region corresponds to the parallel plate waveguide region. (b) PML reflection coefficients for the geometry illustrated at 3.3a. The minimum reflection is obtained when constant profiling is used (i.e. $m = 0$).</td>
<td>32</td>
</tr>
<tr>
<td>3.4 Snapshot of the magnitude of the electric field distribution for the coated sphere.</td>
<td>33</td>
</tr>
</tbody>
</table>
3.5 RCS(dB) of a coated sphere with inner radius $a = 3.0m$ and outer radius $b = 3.25m$ at 300 MHz. (blue-DGTD, pink-Mie series). (a) H-plane (b) E-plane.

3.6 (a) Surface mesh of the F16-aircraft. (b) Geometry of F16-aircraft together with the PML truncation. The pink region represents the conformal PML used to truncate the computational domain. The green and red regions correspond to the glass canopy and dielectric radome accordingly.

3.7 Electric field distribution on the surface of the aircraft at different time locations. (a) Top view. (b) Bottom view. (c)-(d) Snapshots of the magnitude of the electric field distribution in the computational domain. The F-16 aircraft is illuminated by a Gaussian pulse, on-nose incidence.

4.1 Geometrical illustration of the impedance surface used in the modeling of $R, L, C$ elements.

4.2 A parallel plate waveguide ($\varepsilon_r = \mu_r = 1$) of width $a = 4m$ and height $b = 4m$, loaded with a resistor, capacitor and inductor was used to study the effect of the proposed resistor, capacitor and inductor formulation on the stability condition of the IP-DGTD method.

4.3 Study of the effect of the resistor in the stability condition for three different values of $R$, with $w = 4m$ and $l = 4m$, same to the parallel plate waveguide dimensions. The simulation was performed based on the stability condition of the unloaded case for 17 periods of time and no instability was observed for all three cases. In (b)-(d) we shown the last 7 periods and a good agreement is observed when comparing with the analytical solution. Therefore, for the proposed average approximation the stability condition between the loaded and the unloaded case remains unchanged.
4.4 Study of the effect of the capacitor in the stability condition for three different values of $C$, with $w = 4m$ and $l = 4m$, same to the parallel plate waveguide dimensions. The simulation was performed based on the stability condition of the unloaded case for 17 periods of time and no instability was observed for all three cases. In (b)-(d) we shown the last 7 periods and good agreement is observed when comparing with the analytical solution. Therefore, for the proposed time discretization the stability condition between the loaded and the unloaded case remains unchanged.

4.5 Study of the effect of the inductor in the stability condition for three different values of $L$, with $w = 4m$ and $l = 4m$, same to the parallel plate waveguide dimensions. The simulation was performed based on the stability condition of the unloaded case for 17 periods of time and no instability was observed for all three cases. In (b)-(d) we shown the last 7 periods and a good agreement is observed when compared with the analytical solution. Therefore, for the proposed time discretization the stability condition between the loaded and the unloaded case remains unchanged.

4.6 Geometry of the interconnect (3mm×3mm×0.6mm) and snapshot of the magnitude of the E-field.

4.7 (a) Modeling of the SMT components as equivalent $R, L, C$ circuit. (b) Partial view of the distribution of the mesh elements after application of the local time stepping algorithm in the interconnect example. In the colorbar the values 0-4 denote class 0-4 respectively.

4.8 Comparison of magnitude and phase of $S_{11}$ and $S_{12}$ with measurement and our frequency domain FEM code. A very good agreement is observed with both measurement and frequency domain data.

5.1 (a) Mapping of DGTD to CUDA programming model. (b) Implementation of one update step of the LTS algorithm in the GPU. Here, $N_{elements}$ is the numbers of elements in partition $M_i$ of the class that is currently updated.
5.2 (a) Partitioning and assignment of MPI processes to the partitions. (b)-(c) Proposed MPI/LTS strategy. A number of MPI processes $P_0, P_1, ..., P_N$ work in parallel for every step in the LTS algorithm. Here LE denotes a leap-frog E update and LH denotes a leap-frog H update.

5.3 (a) Use MPI and run one MPI process per GPU. (b) Use host threads to run multiple GPUs on each cluster node, and MPI for inter-node communications.

5.4 (a) Pre-processing step to set up all necessary buffers for communication between subdomains. (b) Flowchart of how one leap-frog E or leap-frog H update of the LTS algorithm shown in Figure 2 is performed in the MPI/GPU implementation.

5.5 Coated sphere used for the performance analysis.

5.6 Scalability and parallelization efficiency of MPI/CPU implementation.

5.7 Scalability and parallelization efficiency of MPI/GPU implementation.

5.8 Speed up in iteration time when comparing one CPU against one CPU core.

5.9 (a)-(b) Geometry of the 3D SRR cloaking device. (c) Partial view of the mesh. The colored regions show the different METIS partitions.

5.10 A pulse with 8GHz-11GHz 3dB bandwidth propagating through the cloaking device. Small but noticeable scattered field in the forward region and ripples, due to time delay, as the pulse exits the cloak, reduce the clocking effectiveness.

6.1 (a) Generic illustration of the domain partitioning into non-overlapping subdomains. The subdomains must have planar interfaces but they can also be geometrically non-conformal. (b) The dashed red line shows the intersection where the coupling is enforced. Example of conformal case coupling. (c) The dashed red line shows the intersection where the coupling is enforced. Example of non-conformal case coupling.

6.2 (a) Example of non-conformal planar interface. (b) Calculation of interface matrix on the non-conformal interface.
6.3 (a)-(c) Pictorial illustration of the three steps of Algorithm 1 for a parallel plate waveguide example. The original five unbalanced subdomains are regrouped into five new balanced subdomains.

6.4 (a) FLO distribution for original subdomains. One can clearly notice the original subdomains are not well balanced for computation. (b) FLO distribution for new subdomains. The new subdomains are better balanced for computation.

6.5 (a) Geometry of the problem. The computational domain is decomposed into 8 subdomains. (b) Comparison of magnitude of $S_{11}$ parameter. (c) Comparison of phase of $S_{11}$ parameter. (d) Comparison of magnitude of $S_{12}$ parameter. (e) Comparison of phase of $S_{12}$ parameter.

6.6 (a) IC package provided by IBM. Here, we focus our attention on a signal integrity analysis for the part enclosed in the red box (courtesy of [32]). (b) Schematic illustration of the eight layers in IBM model structure.

6.7 (a)-(h) Top view of the metalization of the 8 layers that the IBM model consists of: (ground/mounting pads (SURFACE), signal (FC3), ground (FC2), signal (FC1), signal (BC1), power (BC2), signal (BC3), and ground/mounting pads (BASE)). Each layer is decomposed into a number of subdomains resulting in a total of 149 subdomains.

6.8 (a) Distribution of the $FLO_{SD_1}$ for the original/unbalanced 149 subdomains. (b) Distribution of the $FLO_{SD_1}$ for the new balanced 149 subdomains.

6.9 (a) Simulation set up. The input of line DAT06 is excited with a step source with 80$ps$ rise time and the voltage at the output of DATA06 is computed. (b) Snapshot of the distribution of electric field magnitude at domain 41 where the input port is located. (c) Comparison between computed and measured voltages. A fairly good agreement with the measurement data is observed. (d) Snapshot of the distribution of electric field magnitude at the signal layer FC3.
Chapter 1: Introduction

1.1 Background

The complexity of modern electromagnetic applications demand sophisticated and efficient numerical methods. This requirement are even more critical for time domain simulations. Schemes such as Finite Difference Time Domain (FDTD) [35] are simple and efficient, but can suffer from dispersion and have some difficulty to handle very complex geometries. Other types of methods, like Finite Element Time-Domain (FETD) [23] methods can deal with complex geometries, and unstructured grids. However, they require heavy computation since a solution of a linear system is needed at each time iteration unless lumping of the mass matrices is applied to facilitate an explicit scheme [23]. Moreover, hardware architectures such as multi-core CPUs and/or Graphics Processor Units (GPUs) are widely available for computation. Therefore, numerical methodologies that exhibit high parallelism and can be mapped effectively to the latest hardware architectures are also desirable.

Discontinuous Galerkin (DG) finite element methods offer an alternative for time-domain simulations. In general, discontinuous Galerkin methods decompose the computational domain into a number of disjoint polyhedral (elements). For each polyhedron, local basis functions are defined and the fields are approximated as a linear
combination of these basis functions. To ensure equivalence to the original problem the tangential continuity of the electric and magnetic fields need to be enforced between polyhedra interfaces. This condition is applied in the weak sense by proper penalty terms on the variational formulation also known as numerical fluxes.

Hence, one can notice that by nature, DG methods can support various types and shapes of elements, non-conformal meshes and non-uniform orders of approximation. Additionally, for uniform cartesian grids and when \( kh << 1 \) (\( k \) is the exact wavenumber) a dispersion error of \( O((hk)^{2p+3}) \) [18, 4] can be shown, where \( p \) is the polynomial approximation order. Moreover, since the tangential continuity of the fields and boundary conditions can be enforced in the weak sense, significant freedom in the choice of basis functions is available. In this way, a great amount of flexibility is available and this is a major strength of DG methods. Furthermore, the resulting mass matrix is a block diagonal matrix with the block size equal to the number of degrees of freedom in the element. Therefore, the method can lead to a fully explicit time-marching scheme for the solution in time. Finally, information exchange is required only between neighboring elements regardless of approximation order and shape, which then leads to high efficiency in parallelization.

In electromagnetism, DG methods were recently applied for the solution of the time dependent Maxwell’s equations [17, 11, 26, 13]. In [17], Heathhaven and WARBURTON developed a low storage, high-order Runge-Kutta DGTD method based on upwind fluxes. The authors showed that upwind fluxes, which closely resemble a Robin type transmission condition in the frequency domain, will result in optimal \( h \)-convergence rate \( O(h^{p+1}) \). Moreover, in [18, 4] for uniform cartesian grids with
$kh \ll 1$ ($k$ is the exact wavenumber), the authors present a $O((hk)^{2p+3})$ error estimate for the dispersion error, and a $O((hk)^{2p+2})$ for the dissipation error respectively, again for an upwind flux DGTD.

On the other hand, Fezoui et al. [11] formulated an energy conservative DGTD method based on central fluxes and leap-frog discretization in time, but with sub-optimal $h$-convergence rate $O(h^p)$. Also, in Fahs. et al. [10] the authors studied a high order DGTD based on central fluxes and non-conformal meshes and results were shown only for two dimensional problems. Furthermore, Montseny et al. [26] proposed a DGTD method for hexahedra with the option of either central or upwind flux. The authors used Gauss-Lobatto quadrature formulas to obtain a method with low cost in memory and CPU time. Montseny et al. [26] present a $O(h^{p-1})$ convergence rate for the central flux option and a $O(h^{p-1/2})$ convergence rate for the upwind flux case.

As mentioned earlier, DG methods can result in an explicit but also conditionally stable time marching scheme. After the space discretization is complete, a semi-discrete system of ODEs is obtained. The stability condition for the fully discrete system will then depend on the method used for the time discretization of the system of ODEs. Although, different methods (i.e. Runge-Kutta, leap-frog, etc) may lead to different stability conditions there is one common bottleneck that is shared. In applications where the generated unstructured meshes contain very small or distorted elements, the stability condition will result in a very small time step $\delta t$. Consequently, the CPU time will significantly increase. However, since in DGTD methods each element has it is own local stability limit, more sophisticated time marching schemes could be developed. Such approaches were studied in [11, 26] where the authors developed a local time stepping algorithm based on the Verlet and the leap-frog.
schemes respectively. Another approach based on a hybrid explicit/implicit scheme was proposed in [7]. In that work the locally refined regions were updated with an implicit and unconditionally stable scheme, whereas the rest of the elements were updated in an explicit fashion. Both strategies have shown that there is significant gain in the solution time and that the gain is also problem dependent.

Finally, as previously mentioned, another important feature of DG methods is that they offer high capacity for parallelism. That makes DG methods a good candidate for modern hardware like multi-core CPUs and graphics processor units (GPUs). In the context of DG methods, GPU computing was initially introduced in [21], where a single precision GPU implementation of DGTD was presented. The authors presented a 40x-60x speedup when comparing one GPU versus one CPU core. Furthermore, [15] presented their approach to implementing DGTD in GPU clusters, and the authors reported a 20x speedup in the solution time for a single precision implementation. Also, in [15] the GPU communication was done in a shared memory architecture, since all GPU devices physically reside on the same cluster node and scalability results were reported for up to 8 GPUs. Furthermore, in [16] a multi-rate GPU implementation of DGTD was presented, but there was a restriction to only two classes for time marching. In the following we discuss the specifics of our DG formulation and also present the contributions that this research work has to offer.

1.2 Contribution & Organization

In this dissertation, we focus on a Interior Penalty (IP) approach [5, 19, 29] as one of the ways to derive a DG formulation for the first order Maxwell’s equations in the time-domain. First in Chapter 2, we formulate our Interior Penalty Discontinuous
Galerkin Time-Domain (IPDGTD) method in a unique way based on a general IP approach. Moreover, we present a physical interpretation of the IP approach, which we find to be interesting. The goal of this physical interpretation is to provide a physical insight into the IP method and link very important concepts like the duality pairing principle to a physical meaning. Furthermore, we discuss the time discretization based on a leap-frog scheme and the stability analysis of the fully discrete system. Additionally, we present how we can employ a local time-stepping strategy similar to the one in [26] into our framework, to make the time marching scheme more efficient. Finally, we numerically study the convergence rate for both central and upwind fluxes and compare with the theoretical estimates derived in [11, 17] accordingly.

In Chapter 3, we present our approach to incorporate a conformal Perfectly Matched Layer within the IPDGTD method. From a practical point of view, a conformal PML is easier to model compared to a Cartesian PML and can reduce the buffer space between the structure and the truncation boundary, thus potentially reducing the number of unknowns. Moreover, we study and discuss the issues concerning the PML’s stability and performance and present our conclusions.

Following, in Chapter 4 we present an approach to combine EM and circuit simulation within the IPDGTD framework. This approach is unique and is the main contribution of Chapter 4. Namely, we discuss how to incorporate passive lumped elements such as resistors, capacitors and inductors in DGTD. Since the lumped elements used in practical applications are small compared to the wavelength, we can assume that the electric and magnetic fields are constant on the surface of the lumped element. We begin with the voltage and current relationships and derive the equivalent relationships that describe each of the $R, L, C$ in terms of the electric and
magnetic fields. Next, these field expressions are weakly enforced through the IPDG formulation.

Continuing, in Chapter 5 we present our design of a scalable parallel implementation of our IPDGTD method. Our parallelization, is aimed at multi-core CPUs and/or graphics processor units (GPUs), for shared and/or distributed memory systems. In this way, both the MPI/CPU and MPI/GPU configurations can be used. In our approach, we exploit the inherent DGTD parallelism and describe a combined MPI/GPU and local time-stepping implementation. This combination is aimed at increasing efficiency and reducing computational time, especially for multi-scale applications. The main contributions of this Chapter are summarized in the following: (a) we present our approach for an MPI/GPU implementation of DGTD, on conformal meshes with uniform degrees of approximation. The proposed approach is applicable on large GPU clusters with distributed-memory architecture. Moreover, we report a 10x speedup compared to CPU clusters (one GPU against one CPU core) in double precision arithmetic using Quadro FX 5800 cards; (b) we combine the local time stepping (LTS) [26] algorithm with MPI/GPU to increase efficiency and reduce the computational time for multi-scale applications, and (c) we show good scalability and parallelization efficiency of 85% with up to 40 GPUs and 80% with up to 160 CPU cores, on the Glenn Cluster at the Ohio Supercomputer Center.

Finally, in Chapter 6 we present an approach to incorporate non-conformal meshes in IPDGTD. In cases of complicated geometries where a conformal mesh is very hard to obtain, the ability to handle non-conformal meshes is important. Firstly, the computational domain is decomposed into non-overlapping sub-domains. Then, each sub-domain is meshed independently resulting in non-conformal domain interfaces
but simultaneously providing great flexibility in the meshing process. In this way a final mesh can be obtained for computation. The non-conformal triangulations at sub-domain interfaces are naturally supported within the IPDGTD framework. Additionally, a MPI parallelization together with a LTS is applied again to significantly increase the efficiency of the method. To the best of our knowledge, a parallel and non-conformal (both geometrically and mesh-wise) DGTD method for unstructured meshes in three dimensions has not been presented before in the literature and this is the main contribution of this chapter. Finally, some interesting numerical examples demonstrate the capabilities of the proposed approach.
Chapter 2: Interior Penalty Discontinuous Galerkin Time Domain Formulation

In this chapter, we follow a interior penalty (IP) based approach [8] to derive our discontinuous Galerkin formulation. Also, we discuss a physical interpretation of our approach in order to provide a more intuitive and deeper physical insight to the IP method and link mathematical principles like the duality pairing principle to a physical meaning. This physical interpretation is a key feature that is also used in later chapters of this dissertation. Furthermore, we study the stability analysis of the fully discrete system showing that the proposed scheme is explicit and conditionally stable. Additionally, we discuss about a local time-stepping strategy that will provide a more efficient updating scheme by significantly reducing the CPU time, especially for multi-scale applications. Finally, we study the convergence rate ($h-$convergence) of the derived IPDGTd. As will be shown, the optimality (optimal or sub-optimal) of the convergence rate depends on the choice of some parameters in the IP formulation. These parameters will define what is commonly referred to as a numerical flux which essentially represents the way the tangential electric and magnetic fields are coupled through element interfaces.
2.1 Original Initial Value Problem (IVP)

We consider the time-dependent Maxwell’s equations in three dimensions. The electric permittivity $\bar{\epsilon}(r)$ and the magnetic permeability $\bar{\mu}(r)$ are symmetric positive definite tensors varying in space.

\[
\nabla \times \mathbf{E} = -\bar{\mu} \frac{\partial \mathbf{H}}{\partial t} \quad \text{in } \Omega \times [0, T] \tag{2.1}
\]

\[
\nabla \times \mathbf{H} = \bar{\epsilon} \frac{\partial \mathbf{E}}{\partial t} \quad \text{in } \Omega \times [0, T] \tag{2.2}
\]

\[
\mathbf{E}(r, t = 0) = \mathbf{E}_0, \quad \mathbf{H}(r, t = 0) = \mathbf{H}_0 \tag{2.3}
\]

The above equations are solved in a bounded domain $\Omega \subset \mathbb{R}^3$. Moreover, we apply boundary conditions that are:

\[
\hat{n} \times \mathbf{E} = 0 \quad \text{on } \Gamma_{PEC}
\]

\[
\hat{n} \times \mathbf{H} = 0 \quad \text{on } \Gamma_{PMC}
\]

or some kind of absorbing boundary condition, applied on the boundary $\partial \Omega$ of $\Omega$, for the truncation of the computational domain.

2.2 IPDGTDT Formulation and Physical Interpretation

2.2.1 Trace operators and Notation

Let $\Omega$ be the computational domain of interest and $\mathcal{T}_h$ the discretization of $\Omega$ into polyhedra $K$ ($\Omega \approx \bigcap K_i$). Continuing, let $K_i$ and $K_j$ be two adjacent elements of $\mathcal{T}_h$. We denote by $\mathcal{F}^I_h$ the set of all interior faces, $\partial K_i \cap \partial K_j$, of $\mathcal{T}_h$. Moreover, by $\mathcal{F}^B_h$ we notate the set of all boundary faces $\partial K_i \cap \partial \Omega$, such that $\mathcal{F}_h = \mathcal{F}^I_h \cup \mathcal{F}^B_h$. Next, we introduce the notations for the trace operators used in our analysis. Define the tangential trace and projection (“components trace”) operators, $\gamma_r(\cdot)$ and $\pi_r(\cdot)$. 
respectively, as $\gamma(u_i) = \mathbf{n}_i \times u_i|_{\partial K_i}$ and $\pi(u_i) = \mathbf{n}_i \times (u_i \times \mathbf{n}_i)|_{\partial K_i}$, where $\mathbf{n}_i$ is the boundary normal pointing out of the element $K_i$. We define the following traces operators:

\[
\begin{align*}
\{u\}_\gamma &= (\pi(u_i) + \pi(u_j))/2 \\
[u]_\gamma &= \gamma(u_i) + \gamma(u_j) \\
[u]_\pi &= \pi(u_i) - \pi(u_j)
\end{align*}
\]

(2.4) on $\mathcal{F}_h^I$

\[
\begin{align*}
\{u\}_\gamma &= \gamma(u) \\
[u]_\gamma &= \gamma(u) \\
[u]_\pi &= \pi(u)
\end{align*}
\]

(2.5) on $\mathcal{F}_h^B$

Moreover, $(u, v)_K = \int_K u \cdot v \, dv$ and $\langle u, v \rangle_S = \int_S u \cdot v \, ds$. Finally, $\int_\Omega \psi \, dv$ in this chapter is used in the sense $\int_\Omega \psi \, dv = \sum_{K_i \in \mathcal{T}_h} \int_{K_i} \psi \, dv$. Likewise, $\int_{\mathcal{F}_h} \psi \, ds = \sum_{f \in \mathcal{F}_h} \int_f \psi \, ds$.

### 2.2.2 Formulation-Space Discretization

To begin with, let us introduce the general principles of DG methods which can be summarized in the following three steps:

- Partition the computational domain into a number of polyhedra (elements) $K_i$.

- Inside each polyhedron $K_i$, define a set of local basis functions and approximate the fields as a linear combination of these basis functions. The approximate fields are allowed to be discontinuous between neighboring elements. In this work complete order hierarchical edge elements are used to approximate the fields.

- Add proper penalty terms on the variational formulation to weakly enforce continuity at element interfaces.

For simplicity, but without any loss of generality, let us consider the case where the domain is discretized into two adjacent polyhedra $K_i$ and $K_j$, and let $\Gamma = \partial K_i \cap \partial K_j \ (\in \mathcal{F}_h^T)$. Note that $K_i$ and $K_j$, can also be two arbitrary shaped, adjacent
subdomains of the computational domain and the analysis presented here remains the same. However, in the following, we assume each element is a subdomain and therefore, $K_i$ and $K_j$ are two adjacent elements as shown in Figure 2.1. Moreover, a metallic boundary condition is assumed on the boundary of the domain. Then the original BVP statement can be restated formally as:

\[
\begin{aligned}
\nabla \times E_i &= -\vec{\mu}_i \frac{\partial H_i}{\partial t} \quad \text{in } K_i \\
\nabla \times H_i &= \vec{\varepsilon}_i \frac{\partial E_i}{\partial t} \quad \text{in } K_i \\
\nabla \times E_j &= -\vec{\mu}_j \frac{\partial H_j}{\partial t} \quad \text{in } K_j \\
\nabla \times H_j &= \vec{\varepsilon}_j \frac{\partial E_j}{\partial t} \quad \text{in } K_j \\
\hat{n}_i \times E_i &= -\hat{n}_j \times E_j \quad \text{on } \Gamma = \mathcal{F}_h \\
\hat{n}_i \times H_i &= -\hat{n}_j \times H_j \quad \text{on } \Gamma = \mathcal{F}_h \\
\hat{n}_{ext} \times E_{i(j)} &= 0 \quad \text{on } \mathcal{F}_h^S
\end{aligned}
\] (2.6-2.10)

At this point, let us note that (2.8)-(2.9) together with the uniqueness theorem ensure that the decomposed BVP described above and the original BVP are equivalent. Finally, before we proceed to the weak formulation let us define the following function spaces [25],

\[
\begin{aligned}
\mathbf{H}(\text{curl}, K_{i(j)}) &= \{ u \in L^2(K_{i(j)}) \mid \nabla \times u \in L^2(K_{i(j)}) \} \\
\mathbf{H}(\text{div}, K_{i(j)}) &= \{ u \in L^2(K_{i(j)}) \mid \nabla \cdot u \in L^2(K_{i(j)}) \} \\
\mathbf{H}(\text{curl}_\tau, \partial K_{i(j)}) &= \{ \pi_\tau (u) \mid u \in \mathbf{H}(\text{curl}, K_{i(j)}) \} \\
\mathbf{H}(\text{div}_\tau, \partial K_{i(j)}) &= \{ \gamma_\tau (u) \mid u \in \mathbf{H}(\text{curl}, K_{i(j)}) \}
\end{aligned}
\]

The Galerkin Statement

In the previous BVP statement, equations (2.6)-(2.9) are not satisfied exactly. Therefore, we proceed to introduce the residuals and their related function spaces.
Namely,

\[
\begin{align*}
\mathbf{R}_{K_i}^{(1)} &= \nabla \times \mathbf{E}_i + \bar{\mu}_i \frac{\partial \mathbf{H}_i}{\partial t} \quad &\in &\quad \mathbf{H}(\text{div}, K_i) \\
\mathbf{R}_{K_i}^{(2)} &= \nabla \times \mathbf{H}_i - \bar{\epsilon}_i \frac{\partial \mathbf{E}_i}{\partial t} \quad &\in &\quad \mathbf{H}(\text{div}, K_i) \\
\mathbf{R}_{K_j}^{(3)} &= \nabla \times \mathbf{E}_j + \bar{\mu}_j \frac{\partial \mathbf{H}_j}{\partial t} \quad &\in &\quad \mathbf{H}(\text{div}, K_j) \\
\mathbf{R}_{K_j}^{(4)} &= \nabla \times \mathbf{H}_j - \bar{\epsilon}_j \frac{\partial \mathbf{E}_j}{\partial t} \quad &\in &\quad \mathbf{H}(\text{div}, K_j) \\
\mathbf{R}_\Gamma^{(5)} &= \hat{\mathbf{n}}_i \times \mathbf{E}_i + \hat{\mathbf{n}}_j \times \mathbf{E}_j \quad &\in &\quad \mathbf{H}(\text{div}, \Gamma) \\
\mathbf{R}_\Gamma^{(6)} &= \hat{\mathbf{n}}_i \times \mathbf{H}_i + \hat{\mathbf{n}}_j \times \mathbf{H}_j \quad &\in &\quad \mathbf{H}(\text{div}, \Gamma)
\end{align*}
\]

To understand the method better, we associate each of these residuals with a corresponding physical meaning as shown in Table 2.1. The residual \(\mathbf{R}_{K_i}^{(1)}\) can be interpreted as the time changing volume error \(\mathbf{B}_i^{\text{err}}\) of the magnetic flux density. Likewise, \(\mathbf{R}_{K_i}^{(2)}\) can be viewed as the time changing volume error \(\mathbf{D}_i^{\text{err}}\) of the electric flux density.
Table 2.1: Physical Interpretation of the Duality Pairing

<table>
<thead>
<tr>
<th>Residual</th>
<th>Physical Meaning</th>
<th>Duality Testing</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{K_i}^{(1)}$</td>
<td>$\partial B_i^{err} / \partial t$</td>
<td>$w_i \in H(\text{curl}, K_i)$</td>
<td>$H \cdot \partial B_i^{err} / \partial t$</td>
</tr>
<tr>
<td>$R_{K_i}^{(2)}$</td>
<td>$\partial D_i^{err} / \partial t$</td>
<td>$v_i \in H(\text{curl}, K_i)$</td>
<td>$E \cdot \partial D_i^{err} / \partial t$</td>
</tr>
<tr>
<td>$R_{K_j}^{(3)}$</td>
<td>$\partial B_j^{err} / \partial t$</td>
<td>$w_j \in H(\text{curl}, K_j)$</td>
<td>$H \cdot \partial B_j^{err} / \partial t$</td>
</tr>
<tr>
<td>$R_{K_j}^{(4)}$</td>
<td>$\partial D_j^{err} / \partial t$</td>
<td>$v_j \in H(\text{curl}, K_j)$</td>
<td>$E \cdot \partial D_j^{err} / \partial t$</td>
</tr>
<tr>
<td>$R_{\Gamma}^{(5)}$</td>
<td>$\mathbf{M}^{err}(\gamma_\tau(H))$</td>
<td>$\pi_\tau(w_i) \in H(\text{curl}_\tau, \Gamma)$</td>
<td>$H \cdot \mathbf{M}^{err}$</td>
</tr>
<tr>
<td>$R_{\Gamma}^{(6)}$</td>
<td>$\mathbf{J}^{err}(\gamma_\tau(H))$</td>
<td>$\pi_\tau(v_i) \in H(\text{curl}_\tau, \Gamma)$</td>
<td>$E \cdot \mathbf{J}^{err}$</td>
</tr>
</tbody>
</table>

The same arguments hold for $R_{K_j}^{(3)}$ and $R_{K_j}^{(4)}$. Furthermore, $R_{\Gamma}^{(5)}$ and $R_{\Gamma}^{(6)}$ represent the surface error magnetic and electric currents respectively.

Continuing, with the Galerkin procedure we need appropriate testing functions that form proper dual pairs. To identify the proper testing, we can use both mathematical and physical arguments. From a physical point of view $R_{K_i}^{(1)}$ should be tested with a magnetic field $H$ to form the energy density $H \cdot B^{err}$. However, $H$ is tangential continuous which means it belongs to $H(\text{curl}, K_i)$ and in this way we have identified the proper testing functions to be $w_i \in H(\text{curl}, K_i)$. Moreover, $R_{K_i}^{(2)}$ should be tested with an electric field $E$ to form the energy density $E \cdot D^{err}$. Furthermore, $E$ belongs to $H(\text{curl}, K_i)$, so the proper testing functions should be $v_i \in H(\text{curl}, K_i)$. Likewise, we can identify the proper testing for $R_{K_j}^{(3)}$ and $R_{K_j}^{(4)}$ as shown in Table 2.1. In a similar manner, $R_{\Gamma}^{(5)}$ should be tested with a surface $H$ to give rise to the energy density $H \cdot \mathbf{M}^{err}$. The surface magnetic field lies in $H(\text{curl}_\tau, \Gamma)$ and thus, we choose $\pi_\tau(w_i)$ as testing functions. Lastly, $R_{\Gamma}^{(6)}$ should be tested with a surface $E$ to form $E \cdot \mathbf{J}^{err}$. The surface electric field lies in $H(\text{curl}_\tau, \Gamma)$ and thus, we choose $\pi_\tau(v_i)$ as testing functions. On the interface $\Gamma$ there are two possibilities for the testing from
either side of the interface. We choose an average of the two possibilities as a testing function. Now we are ready to obtain a weak formulation by testing each residual with a function in its dual space. A linear combination of the weighted residuals can be written as:

\[
\begin{align*}
& a_i(w_i, R_{K_i}^{(1)})_{K_i} + b_i(v_i, R_{K_j}^{(2)})_{K_j} + a_j(w_j, R_{K_j}^{(3)})_{K_j} + b_j(v_j, R_{K_j}^{(4)})_{K_j} \\
& + c \langle \pi_\tau(v_i) + \pi_\tau(v_j), R_\Gamma^{(6)} \rangle_{\Gamma} + d \langle \pi_\tau(w_i) + \pi_\tau(w_j), R_\Gamma^{(5)} \rangle_{\Gamma} \\
& + e \langle \pi_\tau(v_i) - \pi_\tau(v_j), R_\Gamma^{(5)} \times \hat{n}_i \rangle_{\Gamma} + f \langle \pi_\tau(w_i) - \pi_\tau(w_j), R_\Gamma^{(6)} \times \hat{n}_i \rangle_{\Gamma} = 0 \\
& \forall (w_i, v_i) \in H(\text{curl}, K_{i(j)}) \times H(\text{curl}, K_{i(j)})
\end{align*}
\]

where \((w, v)_K = \int_K w \cdot v \, dv \quad \text{and} \quad \langle w, v \rangle_{\Gamma} = \int_{\Gamma} w \cdot v \, ds\). The physical interpretation of (2.17) is that \(M_{\text{err}}, J_{\text{err}}, B_{\text{err}} \), \(D_{\text{err}} \) when tested with every possible \(\pi_\tau(E), \pi_\tau(H), H_{i(j)}\) and \(E_{i(j)}\) respectively, they produce zero energy. Considering the surface testing individually, we see that

\[
\begin{align*}
& \langle \pi_\tau(w_i), dR_\Gamma^{(5)} \rangle_{\Gamma} + f R_\Gamma^{(6)} \times \hat{n}_i \rangle_{\Gamma} = 0 \quad \forall w_i \in H(\text{curl}, K_i) \quad (2.18) \\
& \langle \pi_\tau(w_j), dR_\Gamma^{(5)} \rangle_{\Gamma} - f R_\Gamma^{(6)} \times \hat{n}_i \rangle_{\Gamma} = 0 \quad \forall w_j \in H(\text{curl}, K_j) \quad (2.19)
\end{align*}
\]

and

\[
\begin{align*}
& \langle \pi_\tau(v_i), cR_\Gamma^{(5)} + eR_\Gamma^{(5)} \times \hat{n}_i \rangle_{\Gamma} = 0 \quad \forall v_i \in H(\text{curl}, K_i) \quad (2.20) \\
& \langle \pi_\tau(v_j), cR_\Gamma^{(6)} - eR_\Gamma^{(5)} \times \hat{n}_i \rangle_{\Gamma} = 0 \quad \forall v_j \in H(\text{curl}, K_j) \quad (2.21)
\end{align*}
\]

From the above we see that for non-zero values of \(c, d, e\) and \(f\), both \(R_\Gamma^{(5)}\) and \(R_\Gamma^{(6)}\) are zero which implies the correct enforcement of the boundary conditions on the interface \(\Gamma\). Moreover, note that (2.18)-(2.19) and (2.20)-(2.21) closely resemble the Robin transmission conditions used in domain decomposition methods [24, 36].

The constants \(a_i, a_j, b_i\) and \(b_j\) are chosen as \(a_i = a_j = 1\) and \(b_i = b_j = -1\).
Moreover, the constants $c$, $d$, $e$ and $f$ are yet to be determined. The choices of $c$, $d$, $e$ and $f$, will define a corresponding numerical flux. The type of the numerical flux can drastically affect the numerical properties of the final formulation such as the convergence rate and the numerical dispersion and dissipation. In this work, the effect of this parameters on the convergence rate is studied numerically in section 2.3. whereas the effect on the numerical dispersion and dissipation is left as future work.

**Energy Conservation**

In this section we find the constants $c$, $d$, $e$ and $f$ for which an energy conservative formulation can be obtained. Let $v_i(j) \rightarrow E_i(j)$ and $w_i(j) \rightarrow H_i(j)$. Then making use of the divergence theorem one can rewrite (2.17) as

$$
\sum_{K \in T_h} \int_K \nabla \times \mu \frac{\partial H}{\partial t} + \nabla \times \varepsilon \frac{\partial E}{\partial t} dK = (1 - c + d) \sum_{f \in F_h^e} \int_f E_{K_i} \cdot (\hat{n}_i \times H_{K_i}) ds
$$

$$
- (c + d) \sum_{f \in F_h^e} \int_f E_{K_i} \cdot (\hat{n}_j \times H_{K_j}) ds
$$

$$
- (c + d) \sum_{f \in F_h^e} \int_f E_{K_j} \cdot (\hat{n}_i \times H_{K_i}) ds
$$

$$
- e \sum_{f \in F_h^e} \int_f [ E ]_\pi \cdot [ E ]_\pi ds
$$

$$
- f \sum_{f \in F_h^e} \int_f [ H ]_\pi \cdot [ H ]_\pi ds
$$

(2.22)

From Poynting’s theorem, it is well known that in a region $\Omega$ with no free sources and with the boundary condition $\hat{n} \times E = 0$ on $\partial \Omega$ (metallic cavity)

$$
\int_\Omega \nabla \times \mu \frac{\partial H}{\partial t} + \nabla \times \varepsilon \frac{\partial E}{\partial t} d\Omega = \sum_{K \in T_h} \int_K \nabla \times \mu \frac{\partial H}{\partial t} + \nabla \times \varepsilon \frac{\partial E}{\partial t} dK = 0
$$

(2.23)

(2.23) must hold in order to have energy conservation. Therefore, from (2.23) the parameters $c$, $d$, $e$ and $f$ must satisfy $1 - c + d = 0$, $c + d = 0$ and $e = f = 0$ in
order to result in an energy conserved system. Solving the systems of equations we find \( c = -d = 1/2 \) and \( e = f = 0 \). Finally, in the following we will assume constant material properties within each element. Therefore, one can define the following finite-dimensional space: \( V_h^k = \{ v \in \mathbb{L}^2(\Omega)^3 : v|_K \in [\mathbb{P}^k(K)]^3, \forall K \in \mathcal{T}_h \} \) as the discrete trial space.

Central Flux

The central flux formulation corresponds to the special case of \( c = -d = 1/2 \) and \( e = f = 0 \) of the general IPDG method described previously. As shown in the previous section, this choice of constants will give rise to a conservative formulation. On the other hand, a central flux has a suboptimal (\( O(h^p) \)) rate of convergence as discussed in [11, 8] and also shown numerically in section 2.3. The final formulation using a central flux can be formally stated as:

\[
\text{Find } (H, E) \in V_h^k \times V_h^k \text{ such that } \]
\[
\int_{\Omega} w \cdot (\nabla \times E + \mu \frac{\partial H}{\partial t}) \, d\Omega - \int_{\Omega} v \cdot (\nabla \times H - \epsilon \frac{\partial E}{\partial t}) \, d\Omega \\
+ \int_{\Gamma_h} \{v\} \cdot \{H\}_{\gamma} \, ds - \int_{\Gamma_h} \{w\} \cdot \{E\}_{\gamma} \, ds = 0 \quad \forall (w, v) \in V_h^k \times V_h^k \tag{2.24}
\]

Upwind Flux

In this case we choose \( c = -d = 1/2 \) and \( e \neq f \neq 0 \). More specifically, \( e = \frac{1}{2Z_\Gamma} \) and \( f = \frac{1}{2Y_\Gamma} \) with \( Z_\Gamma = \frac{1}{2} \left( \sqrt{\frac{\mu_i}{\epsilon_i}} + \sqrt{\frac{\mu_j}{\epsilon_j}} \right) \), \( Y_\Gamma = \frac{1}{2} \left( \sqrt{\frac{\epsilon_i}{\mu_i}} + \sqrt{\frac{\epsilon_j}{\mu_j}} \right) \). For these choices of constants, one obtains a slightly lossy formulation. However, upwind flux provides an optimal (\( O(h^{p+1}) \)) rate of convergence as discussed in [17, 8] and also shown numerically in section 2.3. The final formulation using a upwind flux can be formally stated as:
Find \((\mathbf{H}, \mathbf{E}) \in V_h^k \times V_h^k\) such that
\[
\begin{align*}
\int_{\Omega} \mathbf{w} \cdot (\nabla \times \mathbf{E} + \frac{\mu}{\varepsilon} \frac{\partial \mathbf{H}}{\partial t}) d\Omega &- \int_{\Omega} \mathbf{v} \cdot (\nabla \times \mathbf{H} - \frac{1}{\varepsilon} \frac{\partial \mathbf{E}}{\partial t}) d\Omega \\
+ \int_{\partial \Omega} \{\mathbf{v}\} \cdot [\mathbf{H}]_{\gamma} ds - \int_{\partial \Omega} \{\mathbf{w}\} \cdot [\mathbf{E}]_{\gamma} ds &+ \frac{1}{2Z_{\Gamma}} \int_{\partial \Omega} [\mathbf{v}]_{\pi} \cdot [\mathbf{E}] ds + \frac{1}{2Y_{\Gamma}} \int_{\partial \Omega} [\mathbf{w}]_{\pi} \cdot [\mathbf{H}] ds = 0
\end{align*}
\]
\(\forall (\mathbf{w}, \mathbf{v}) \in V_h^k \times V_h^k\)

### 2.2.3 Formulation-Time Discretization

Let us expand the electric and magnetic fields within element \(K_i\) in terms of basis functions \(\mathbf{w}, \mathbf{v} \in V_h^k\) as, \(\mathbf{E}^{(i)}(r, t) = \mathbf{E}(r, t)|_{K_i} \approx \sum_{n=1}^{d_i} e_{in}(t) \mathbf{v}_{in}(r)\) and \(\mathbf{H}^{(i)}(r, t) = \mathbf{H}(r, t)|_{K_i} \approx \sum_{n=1}^{d_i} h_{in}(t) \mathbf{w}_{in}(r)\), where \(d_i\) is the number of the degrees of freedom in element \(K_i\). Separating the \(\mathbf{w}, \mathbf{v}\) testings in (2.25) we get in matrix form a semidiscrete system within each element

\[
\begin{align*}
\mathbf{M}_e \frac{\partial \mathbf{e}_i}{\partial t} &= \mathbf{S}_e \mathbf{h}_i - \mathbf{F}_e^{ij} \mathbf{h}_j + \epsilon \mathbf{P}_e^{ij} \mathbf{e}_i - \mathbf{F}_e^{ij} \mathbf{h}_j + \epsilon \mathbf{P}_e^{ij} \mathbf{e}_j \quad \text{(2.26)} \\
\mathbf{M}_h \frac{\partial \mathbf{h}_i}{\partial t} &= -\mathbf{S}_h \mathbf{e}_i + \mathbf{F}_h^{ij} \mathbf{e}_j + \mathbf{P}_h^{ij} \mathbf{h}_i + \mathbf{F}_h^{ij} \mathbf{e}_j + \mathbf{P}_h^{ij} \mathbf{h}_j \quad \text{(2.27)}
\end{align*}
\]

where \(\mathbf{e}_{i(j)}\) and \(\mathbf{h}_{i(j)}\) are the coefficient vectors for the electric and magnetic field, respectively, and
Therefore, the backward approximations in the upwind flux formulation will become globally implicit due to the coupling terms from the neighboring elements. We use the backward approximation \( (e^{n+1})_{i} \approx e^{n}_{i+j} \) and \( h^{n+1}_{i} \approx h^{n+1}_{i+j} \). If an average approximation was used i.e. \( (e^{n+1}_{i}) \approx \frac{e^{n}_{i} + e^{n}_{i+1}}{2} \) and \( h^{n+1}_{i} \approx \frac{h^{n}_{i} + h^{n+1}_{i+1}}{2} \), then the system will become globally implicit due to the coupling terms from the neighboring elements. Therefore, the backward approximations in the upwind flux formulation

\[
\begin{align*}
(M_e)_{n,m} &= \int_K v_{in} \bar{v}_i \cdot v_{im} dK \\
(S_e)_{n,m} &= \int_K v_{in} \cdot (\nabla \times w_{im}) dK \\
(F_e^{ii})_{n,m} &= \frac{1}{2} \int_{F_h^{ii}} \pi_i (v_{in}) \cdot \gamma_i (w_{jm}) ds \\
(F_e^{ij})_{n,m} &= \frac{1}{2} \int_{F_h^{ij}} \pi_i (v_{in}) \cdot \gamma_i (w_{jm}) ds \\
(P_e^{ii})_{n,m} &= - \int_{F_h^{ii}} \pi_i (v_{in}) \cdot \pi_i (w_{im}) ds \\
(P_e^{ij})_{n,m} &= \int_{F_h^{ij}} \pi_i (v_{in}) \cdot \pi_i (w_{jm}) ds
\end{align*}
\]

are the local matrices. The system of ODE (2.26) - (2.27) can be discretized in time in a variety of ways. We use a leap-frog scheme, which is second order accurate. The electric field unknowns are evaluated at \( t_n = n \Delta t \) and the magnetic field unknowns are evaluated at \( t_{n+\frac{1}{2}} = (n + \frac{1}{2}) \Delta t \).

\[
M_e \frac{\partial e_i}{\partial t} = S_e h_i - F_e^{ii} h_i + eP_e^{ii} e_i - F_e^{ij} h_j + eP_e^{ij} e_i \bigg|_{t=n+\frac{1}{2}} \quad (2.28)
\]

\[
M_\mu \frac{\partial h_i}{\partial t} = -S_h e_i + F_h^{ii} e_i + fP_h^{ii} h_i + F_h^{ij} e_j + fP_h^{ij} e_j \bigg|_{t=n+\frac{1}{2}} \quad (2.29)
\]

The first order time derivatives will be approximated using the central difference method, which is second order accurate. Namely,

\[
\frac{\partial e_i}{\partial t} \bigg|_{n+\frac{1}{2}} = \frac{e^{n+1}_i - e^n_i}{\delta t} + O(\delta t^2) \quad (2.30)
\]

Moreover, for the two extra penalty terms arising from the upwind flux formulation, we use the backward approximation \( e^{n+1}_{i} \approx e^{n}_{i+j} \) and \( h^{n+1}_{i} \approx h^{n+1}_{i+j} \). If an average approximation was used i.e. \( (e^{n+1}_{i}) \approx \frac{e^{n}_{i} + e^{n}_{i+1}}{2} \) and \( h^{n+1}_{i} \approx \frac{h^{n}_{i} + h^{n+1}_{i+1}}{2} \), then the system will become globally implicit due to the coupling terms from the neighboring elements. Therefore, the backward approximations in the upwind flux formulation
are necessary if we want the time marching scheme to remain explicit. In this way, the fully discretized local system of equations can be obtained from the semi-discrete system as

$$M_{\epsilon}e_i^{n+1} = (M_\epsilon + e\delta t P_{\epsilon}^{ii})e_i^n + \delta t (S_\epsilon - F_{\epsilon}^{ii})h_i^{n+1/2} + e\delta t P_{\epsilon}^{ij}e_j^n$$

(2.31)

$$M_{\mu}h_i^{n+3/2} = (M_{\mu} + f\delta t P_{\mu}^{ii})h_i^{n+1} + \delta t(-S_{h} + F_{h}^{ii})e_i^{n+1} + f\delta t P_{h}^{ij}h_j^{n+1/2}$$

(2.32)

### 2.2.4 Stability Analysis and Local Time-Stepping (LTS)

The fully discretized update scheme of (2.31)-(2.32) is conditionally stable. A more detailed description of the stability analysis in given in Appendix A. In this section we give only the final conclusions. By following an approach similar to the energy technique presented in [9, 10], one can obtain the following stability condition:

$$\forall i, \forall j \in \text{Neigh}(i)$$

$$\delta t_1 \left[ 2\alpha_i c_i + \beta_{ij} c_i + 4\beta_{ij} \frac{e_{ij}}{\epsilon_i} \right] < \frac{4V_i}{P_i}$$

(2.33)

$$\delta t_2 \left[ 2\alpha_j c_j + \beta_{ji} c_j + 4\beta_{ji} \frac{e_{ij}}{\epsilon_j} \right] < \frac{4V_j}{P_j}$$

(2.34)

$$\delta t_3 \left[ 2\alpha_i c_i + \beta_{ij} c_i + 4\beta_{ij} \frac{f_{ij}}{\mu_i} \right] < \frac{4V_i}{P_i}$$

(2.35)

$$\delta t_4 \left[ 2\alpha_j c_j + \beta_{ji} c_j + 4\beta_{ji} \frac{f_{ij}}{\mu_j} \right] < \frac{4V_j}{P_j}$$

(2.36)

where $\text{Neigh}(i)$ is the set of indices of the neighbors of element $K_i$. In order to ensure stability we need to choose $\delta t_i = \min_{j \in \text{Neigh}(i)} (\delta t_1, \delta t_2, \delta t_3, \delta t_4)$. Note that $f_{ij(ji)} = \frac{\kappa}{\sqrt{\epsilon_i/\mu_i + \sqrt{\epsilon_j/\mu_j}}}$ and $e_{ij(ji)} = \frac{\kappa}{1/2(\mu_i/\epsilon_i + \mu_j/\epsilon_j)}$ with $\kappa = 0$ for central flux and $\kappa = 1/2$ for the upwind flux case. Moreover, $\alpha_i, \beta_{ij}$ are given below:

$$\alpha_i = \sqrt{V_i^2 ||M^{-1/2}S_1M^{-1/2}||/P_i^2}, \quad \beta_{ij} = V_i ||M^{-1/2}S_2M^{-1/2}||/P_i$$

(2.37)
where,

\[
(M)_{nm} = \int_{K_i} w_{in} \cdot w_{im}, \quad (S_1)_{nm} = \int_{K_i} \nabla \times w_{in} \cdot \nabla \times w_{im}
\]

\[
(S_2)_{nm} = \int_{\partial K_i \cap \partial K_j} w_{in} \cdot (\hat{n}_i \times w_{jm})
\]

Moreover, \( V_i \) is the volume of element \( K_i \), \( P_i \) is the perimeter of \( K_i \) defined as \( P_i = \sum_{f_i} S_{f_i} \), \( (S_{f_i} \) is the area of face \( i \)) and \( c_i = \frac{1}{\sqrt{\mu_i \epsilon_i}} \). In this way, we can compute for each element \( K_i \) its own local \( \delta t_i \). Furthermore, we would like to emphasize that the stability conditions (2.33) - (2.36) are can be used for both conformal and non-conformal meshes within the IPDGT method framework and potentially with arbitrary basis as well. Finally, only for conformal meshes and \( p = 1 \) approximation for all elements one may use the simpler formula \( \forall i, \forall j \in Neigh(i) \), \( c_i \delta t_i \left[ \frac{2\sqrt{5}}{3} + \frac{8}{3} \max\left(\sqrt{\frac{\mu_i}{\mu_j}}, \sqrt{\frac{\epsilon_i}{\epsilon_j}}\right) \right] < \frac{4V_i}{P_i} \) [11] to compute local \( \delta t_i \), which is computationally more efficient since it does not require any matrix norm evaluation.

In the above stability condition, the ratio \( \frac{4V_i}{P_i} \) effectively represents the diameter \( h_i \) of the finite element. As \( \frac{4V_i}{P_i} \) becomes smaller the stability condition provides a smaller \( \delta t_i \). In practical applications, the generated unstructured meshes contain very small or distorted elements and consequently the stability condition will result in a very small time step \( \delta t \). For a standard leap-frog scheme, to guarantee stability we must choose \( \delta t = \delta t_{min} = \min(\delta t_i) \), i.e. \( \delta t_{min} \) is the minimum of all the local \( \delta t_i \). Consequently, CPU time will significantly increase. Most of the time these small elements are not easy to remove and therefore more sophisticated marching schemes need to be developed. To mitigate this problem a local time-stepping strategy proposed in [26] is applied to increase efficiency and reduce the computational time.
detailed description of the method can be found in [26]. Here we will only summarize
the fundamental concepts of the method. We have that:

- The set of elements is partitioned into \( N \) classes. This partition is done before
the time-marching begins and is based on the stability condition.

- For the \( k^{th} \) class choose \( \delta t_k = (2m + 1)^k \delta t_{\text{min}}, \ k = 0, 1, \ldots, N - 1 \). We choose
\( m = 1 \) so that each class has a time step three times larger than the previous
class. Consequently, the effective global time step of the algorithm is the one of
the \( N^{th} \) class and equal to \( \delta t_{\text{global, eff}} = \delta t_{N-1} = (2m + 1)^{N-1} \delta t_{\text{min}} \).

In an attempt to clarify things let us assume two classes \( (N = 2) \), class 0 and class 1,
with \( \delta t_0 = \delta t_{\text{min}} \) and \( \delta t_1 = 3 \delta t_{\text{min}} \) as shown in Figure 2.2. Then, the update scheme
applying the local time-stepping is summarized below:

E-update

Step 1: \( M^1_{e} e_1^{n+1} = M^1_{e} e_1^n + e \delta t_1 P^1_{e} e_1^n \)
\[ + \delta t_1 (S^1_{e} - F^1_{e}) h_1^{n+\frac{1}{2}} - \delta t_1 F^1_{e} h_0^{n+\frac{1}{2}} + e \delta t_1 P^1_{e} e_0^n \]

Step 2: \( M^0_{e} e_0^{n+\frac{1}{6}} = M^0_{e} e_0^{n+\frac{2}{6}} + e \delta t_0 P^0_{e} e_0^{n+\frac{3}{6}} \)
\[ + \delta t_0 (S^0_{e} - F^0_{e}) h_0^{n+\frac{3}{6}} - \delta t_0 F^0_{e} h_1^{n+\frac{3}{6}} + e \delta t_0 P^0_{e} e_1^{n+1*} \]

Step 3: \( M^0_{\mu} h_0^{n+\frac{5}{6}} = M^0_{\mu} h_0^{n+\frac{4}{6}} + f \delta t_0 P^0_{h} h_0^{n+\frac{5}{6}} \)
\[ + \delta t_0 (-S^0_{h} + F^0_{h}) e_0^{n+\frac{5}{6}} + \delta t_0 F^0_{h} e_1^{n+1*} + f \delta t_0 P^0_{h} e_1^{n+1*} \]

Step 4: \( M^0_{e} e_0^{n+1} = M^0_{e} e_0^{n+\frac{1}{6}} + e \delta t_0 P^0_{e} e_0^{n+\frac{2}{6}} e_0^{n+1*} \)
\[ + \delta t_0 (S^0_{e} - F^0_{e}) h_0^{n+\frac{7}{6}} - \delta t_0 F^0_{e} h_1^{n+\frac{7}{6}} + e \delta t_0 P^0_{e} e_1^{n+1*} \]
H-update

Step 5: $M^1_{\mu} h_1^{n+\frac{3}{2}} = M^1_{\mu} h_1^{n+\frac{1}{2}} + f \delta t_1 P^1_{h} h_1^{n+\frac{1}{2}}$

$+ \delta t_1 (-S^1_h + F^1_{h}) e_1^{n+1} + \delta t_1 F^0_{h} e_0^{n+1} + f \delta t_1 P^0_{h} h_0^{n+\frac{1}{2}}$

Step 6: $M^0_{\mu} h_0^{n+\frac{7}{6}} = M^0_{\mu} h_0^{n+\frac{5}{6}} + f \delta t_0 P^0_{h} h_0^{n+\frac{5}{6}}$

$+ \delta t_0 (-S^1_h + F^0_{h}) e_0^{n+1} + \delta t_0 F^0_{h} e_0^{n+1} + f \delta t_0 P^0_{h} h_0^{n+\frac{3}{2}}$

Step 7: $M^0_e e_0^{n+\frac{8}{6}} = M^0_e e_0^{n+1} + e \delta t_0 P^0_{e} e_0^{n+1}$

$+ \delta t_0 (S^0_{e} - F^0_{e}) h_0^{n+\frac{7}{6}} - \delta t_0 F^0_{e} h_0^{n+\frac{3}{2}} + e \delta t_0 P^0_{e} e_0^{n+1}$

Step 8: $M^0_{\mu} h_0^{n+\frac{3}{2}} = M^0_{\mu} h_0^{n+\frac{7}{6}} + f \delta t_0 P_{h} h_0^{n+\frac{7}{6}}$

$+ \delta t_0 (-S^1_h + F^0_{h}) e_0^{n+\frac{8}{6}} + \delta t_0 F^0_{h} e_0^{n+1} + f \delta t_0 P^0_{h} h_0^{n+\frac{3}{2}}$

Figure 2.2: Pictorial illustration of the LTS for a two class (class 0, class 1) examples. (a) E-Update (b) H-Update.

The local time-stepping algorithm described previously can be generalized for N-classes and can be implemented as a recursive process [26]. At this point, let us note
that for the elements that are not located at the interface between the two classes 0 and 1, (2.31)-(2.32) are applied with the time step of the current class without any problems. On the other hand, for the elements that are located at the interface between the two classes the field values at unknown times are replaced by the last known value denoted by $^*$ in (2.38)-(2.39). Finally, let us point out that the stability study of appendix A is for the non local time-stepping case. The stability study with the local time-stepping strategy is a more difficult task and has not been done yet. Based on numerical experiments, in some cases the stability condition derived for the non local time-stepping case needs to be strengthened to ensure stability. As mentioned also in [26] a factor of 0.8 on the $\delta t$ obtained for the non local time-stepping case could be sufficient to guarantee stability in the local time-stepping case.

2.3 Convergence Properties of DGTD

In this section, we study numerically the asymptotic convergence in $L^2$-norm for the electric and magnetic field for both central and upwind fluxes. The polynomial order is assumed to be fixed and all the meshes are unstructured. If $(E, H)$ is the exact solution and $(E_h, H_h)$ is the approximate solution, we are interested of finding the largest real constants $p_1$ and $p_2$ such that $\|E - E_h\|_{L^2(\Omega)^3} \leq C_1 h^{p_1}$ and $\|H - H_h\|_{L^2(\Omega)^3} \leq C_2 h^{p_1}$ for all $C_1, C_2 > 0$, where $h$ is the mesh size. The problem under consideration is the one of a TEM wave propagating in a parallel plate waveguide, for which an analytical solution is known. For this example linear $(p = 1)$ basis were used within each element. Moreover, $\mathbf{k} = (-k_x, 0, 0)$, $\mathbf{E} = (0, 0, E_z)$ and $\mathbf{H} = (0, H_y, 0)$. The excitation is given by $E^{inc} = (0, 0, 1)cos(\omega t + k_x x)$ and $H^{inc} = \frac{1}{\eta}(0, 1, 0)cos(\omega t + k_x x)$. The material properties are $\epsilon_r = 1$ and $\mu_r = 1$. The waveguide was terminated
with a first order Silver-Muller absorbing boundary condition (i.e. $\hat{n} \times E = -c \mu \hat{n} \times \hat{n} \times H$ and $\hat{n} \times H = c e \hat{n} \times \hat{n} \times E$), which is exact for this problem. In Figure 2.3, we present numerical results for the convergence studies using central and upwind fluxes for both electric and magnetic fields. The results show that central flux provides a suboptimal convergence a $O(h^p)$ convergence rate for central flux and a $O(h^{p+1})$ convergence rate for an upwind flux, as predicted by the theoretical estimate derived in [11, 17] accordingly. Moreover, let us note that the overall error can be written as: $e = O(h^{p+1}) + O(\delta t^2) \approx O(h^{p+1}) + O(h^2)$. Therefore, since the leap frog scheme is second order accurate, to maintain the optimal $O(h^{p+1})$ rate for higher order basis in space, higher order time discretizations like the ones proposed in [9, 14] are also

Figure 2.3: Convergence results using central and upwind flux. The red and blue dashed lines show the evolution of the computed error and the black and green solid lines represent the asymptotic behavior. (a) $||E - E_h||_{L^2(\Omega)^3}$. (b) $||H - H_h||_{L^2(\Omega)^3}$. 

24
needed. Finally, a higher order time-integrator will not be presented in this work and is left as future research work.

2.4 Discussion

In this chapter, we have presented an interior penalty (IP) approach to derive a general IPDGTD formulation. Moreover, we discussed a physical interpretation of the IP approach, which we find to be interesting. The goal of this physical interpretation is to provide a physical insight into the IP method and link very important concepts like the duality pairing principle to a physical meaning. We presented that the choice of parameters in the IP method, can give rise to a specific numerical flux which then will affect the numerical properties of the method (i.e convergence rate, numerical dissipation). For the time discretization, a leap-frog scheme was employed, resulting in an explicit and conditionally stable time-marching algorithm. Additionally, we discussed about a local time-stepping strategy that will provide a more efficient updating scheme by significantly reducing the CPU time, especially for multi-scale applications. Moreover, we study the convergence rate ($h$—convergence) of the derived IPDGTD. Our experiments show optimal convergence for the upwind flux and sub-optimal for central flux which is in agreement with the theoretical estimates discussed in [17, 11] accordingly. Furthermore, we note that higher order time discretization schemes that go along with higher order space discretization could prove to be an interesting research topic for the future related to this chapter. Finally, in the following chapters we present various numerical examples that validate our approach.
Chapter 3: Perfectly Matched Layer for IPDGT

In this chapter, we discuss an approach to incorporate a conformal Perfectly Matched Layer within the IPDGT method. Practically, it can be stated, that a conformal PML is more flexible and more convenient to model compared to a Cartesian PML. Moreover, because it conforms to the geometry of the problem, we can possibly reduce the buffer space between the structure and the truncation boundary and thus reduce the number of unknowns. Furthermore, we study and discuss the issues concerning the PML’s stability and performance and present our conclusions. Finally, some numerical examples are provided to illustrate and validate our approach.

3.1 Perfectly Matched Layer

3.1.1 Formulation

For any PDE numerical method, there is a need to truncate the infinite domain for exterior wave propagation problems. For this reason an absorbing boundary condition of some sort is necessary. A first order Silver-Muller absorbing boundary condition (i.e. $\hat{n} \times \mathbf{E} = -c\mu\hat{n} \times \hat{n} \times \mathbf{H}$ and $\hat{n} \times \mathbf{H} = c\epsilon\hat{n} \times \hat{n} \times \mathbf{E}$) is usually the simplest form of an absorbing boundary condition and is easy to implement. However, such a boundary condition is exact only at normal incidence and in order to avoid reflection,
the truncation surface needs to be set further away from the scatterer. This in turn will increase the computational resources.

The Perfectly Matched Layer (PML) introduced by [6] offers an alternative way to truncate the computational domain. The PML is shown to be reflectionless for any angle of incidence, polarization and frequency. The PML proposed in [6] used a split field formulation, which was shown to be weakly ill-posed. However other authors [30, 12] derived an unsplit formulation that is well posed. In frequency domain, [30] formulated the PML as an anisotropic medium which is very convenient for finite element implementations. In this work a standard unsplit anisotropic PML [13] is enforced in time domain with the introduction of the auxiliary vector fields $\mathbf{P}_{e,h}$. Moreover, a conformal PML [33] is utilized to reduce the buffer space. These auxiliary fields exist only in the PML region and thus the additional computational cost is small. Within each element $K_i$ in the PML region, we have

$$\frac{\mu_i \bar{\alpha} \cdot \partial \mathbf{H}_i}{\partial t} = - \nabla \times \mathbf{E}_i - \mu_i \bar{\beta} \cdot \mathbf{H}_i - \mu_i \bar{\gamma} \cdot \mathbf{P}_{hi}$$  
$$\frac{\epsilon_i \bar{\alpha} \cdot \partial \mathbf{E}_i}{\partial t} = \nabla \times \mathbf{H}_i - \epsilon_i \bar{\beta} \cdot \mathbf{E}_i - \epsilon_i \bar{\gamma} \cdot \mathbf{P}_{ei}$$  
$$\frac{\partial \mathbf{P}_{hi}}{\partial t} = \bar{\kappa}^{-1} \cdot \mathbf{H}_i - \bar{d} \cdot \mathbf{P}_{hi}$$  
$$\frac{\partial \mathbf{P}_{ei}}{\partial t} = \bar{\kappa}^{-1} \cdot \mathbf{E}_i - \bar{d} \cdot \mathbf{P}_{ei}$$

As shown in [33], the conformal PML can be realized by using a local diagonal anisotropic tensor $\bar{s}_{u_1,u_2,u_3}$ defined in a local orthogonal curvilinear system ($\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$). The tensors $\bar{\alpha}, \bar{\beta}, \bar{\gamma}, \bar{\kappa}^{-1}$ are also diagonal in the system ($\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$). The entries of $\bar{s}$ are given by $s_{11} = s_2 s_3 / s_1$, $s_{22} = s_1 s_3 / s_2$ and $s_{33} = s_1 s_2 / s_3$. Moreover, $s_1 = \kappa_1(\xi_3, r_1) + \sigma_1(\xi_3, r_1) / j \omega \epsilon_o$, $s_2 = \kappa_2(\xi_3, r_2) + \sigma_2(\xi_3, r_2) / j \omega \epsilon_o$, $s_3 = \kappa_3(\xi_3) + \sigma_3(\xi_3) / j \omega \epsilon_o$ where $r_1$ and $r_2$ are the local radii of curvature at the principal directions $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$. 

27
respectively. Moreover, attenuation is assumed to be in the $\xi_3$ direction. The diagonal tensors $\bar{a}, \bar{b}, \bar{c}, \bar{d}$ are given by [13] $a_{11} = \frac{\kappa_2 \kappa_3}{\kappa_1}$, $b_{11} = \frac{1}{\kappa_1 \epsilon_o} (\sigma_2 \kappa_3 + \sigma_3 \kappa_2 - a_{11} \kappa_3)$, $c_{11} = \frac{\sigma_2 \sigma_3}{\epsilon_o} - b_{11} \frac{\sigma_1}{\epsilon_o}$ and $d_{11} = \frac{\sigma_1}{\kappa_1 \epsilon_o}$. The other diagonals terms are obtained by simple permutation of the subscripts. Next, since the system of differential equations (3.1) - (3.4) is given in the $(x, y, z)$ coordinate system, we need to calculate the representation of any tensor $\bar{\Lambda}_{u_1, u_2, u_3}$ ($\bar{\Lambda} = \bar{a}, \bar{b}, \bar{c}, \bar{d}, \bar{\kappa}$) in $(x, y, z)$. This is done by applying the following transformation matrix

$$T = \begin{pmatrix} u_{1x} & u_{1y} & u_{1z} \\ u_{2x} & u_{2y} & u_{2z} \\ u_{3x} & u_{3y} & u_{3z} \end{pmatrix}$$

(3.5)
on $\bar{\Lambda}_{u_1, u_2, u_3}$ to get $\bar{\Lambda}_{x, y, z} = T^t \bar{\Lambda}_{u_1, u_2, u_3} T$. Finally, let us note that both $\bar{\Lambda}_{u_1, u_2, u_3}$ and $T$ are functions of position.

### 3.1.2 DG Space and Time Discretization

Within each element the electric and magnetic fields and the auxiliary fields are expanded in terms of trial functions in $V_h^k$ as defined in Chapter 2. The testing functions also span the same function space as the trial functions. Applying the interior penalty method described above in the PML region one obtains the following semi-discrete system of coupled differential equations:

$$M_a \frac{\partial e_i}{\partial t} = -M_b e_i + e P_{ei}^i + M_c p_{ei}$$

$$+ (S_e - F_{ei}) h_i - F_{ei}^j h_j + e P_{ei}^j e_j \big|_{t=n+\frac{1}{2}}$$

$$M_a \frac{\partial h_i}{\partial t} = -M_b h_i + f P_{hi}^i + M_c p_{hi}$$

$$+ (-S_h + F_{hi}^i) e_i + F_{hi}^j e_j + f P_{hi}^j h_j \big|_{t=n+1}$$

$$M \frac{\partial p_{ei}}{\partial t} = M_{e-1} e_i - M_d p_{ei} \big|_{t=n+\frac{1}{2}}$$

$$M \frac{\partial p_{hi}}{\partial t} = M_{e-1} h_i - M_d p_{hi} \big|_{t=n+1}$$

(3.6) - (3.9)
where $\mathbf{e}_i$ and $\mathbf{h}_i$ are the time dependent coefficient vectors for the electric and magnetic field, respectively in the element $K_i$ and $j \in Neigh(i)$. Likewise, $\mathbf{p}_{ei}$ and $\mathbf{p}_{hi}$ are, respectively, the time dependent coefficient vectors for the auxiliary fields in the element $K_i$. The system (3.6) - (3.9) of first-order differential equations is discretized in time with a leap-frog scheme. The electric field unknowns together with the $\mathbf{p}_{ei}^n$ are evaluated at $t_n = n\delta t$ and the magnetic field unknowns with the $\mathbf{p}_{hi}^n$ are evaluated at $t_{n+\frac{1}{2}} = (n + \frac{1}{2})\delta t$. The first order derivatives are approximated by central differences as before. Moreover, in (3.8)-(3.9) we approximate $\mathbf{p}_{ei}^{n+\frac{1}{2}} \approx \frac{\mathbf{p}_{ei}^n + \mathbf{p}_{ei}^{n+1}}{2}$, $\mathbf{p}_{hi}^{n+1} \approx \frac{\mathbf{p}_{hi}^{n+\frac{3}{2}}}{2} + \frac{\mathbf{p}_{hi}^{n+\frac{1}{2}}}{2}$, $\mathbf{e}_i^{n+\frac{1}{2}} \approx \frac{\mathbf{e}_{ei}^n + \mathbf{e}_{ei}^{n+1}}{2}$ and $\mathbf{h}_i^{n+1} \approx \frac{\mathbf{h}_{hi}^{n+\frac{3}{2}}}{2} + \frac{\mathbf{h}_{hi}^{n+\frac{1}{2}}}{2}$. Finally, for the above approximations the updating scheme still remains explicit since there are no coupling terms from neighboring elements in (3.8)-(3.9) as the ones existing in (3.6)-(3.7). The only difference compared to the non-PML region is that local matrices are $2d_i \times 2d_i$ due to the auxiliary variables.

### 3.1.3 Late Time Instability and PML performance

It is apparent that the PML is characterized by the parameter $\sigma$ as well by its thickness. The choice of these parameters is closely related to the performance of the PML. In the case of FDTD [12], numerical studies have shown that one can obtain optimal performance by making $\sigma$ position dependent. Typically, a polynomial distribution $\sigma(\xi) = \sigma_{\max} \frac{\xi}{\delta m}$ is employed where $\delta$ is the thickness of the PML, $\sigma$ and $m$ are free-parameters and the attenuation is assumed to be in the $\xi$ direction. However, when profiling is used, it was pointed out in [3] that the standard unsplit PML formulation suffers from late time instability, and our numerical studies verify that as shown in Figure 3.1a. According to [3], long after the solution has passed
through the PML, a linear growth is observed in the PML region. This linear growth will pollute the rest of the computational domain and eventually lead to instability. On the other hand a constant conductivity profile \( m = 0 \) remains stable. However, if constant conductivity is used, in order to avoid severe reflections at the PML/air interface, one must use a small value of conductivity i.e \( \sigma = 0.01 \text{S/m} \).

To mitigate the late time instability problem, a stabilization technique was proposed in [3] to remove the linear growth. Our numerical experiments have shown that the approach proposed in [3] does remove the linear growth as shown in Figure 3.1b. However, we need to investigate how this stabilization affects the properties of the PML. In order to answer this question, we computed the reflection coefficient for a parallel plate waveguide before and after the stabilization. A mesh with average edge length size \( l_{\text{avg}} = \frac{\lambda_{30\text{MHz}}}{7} \) was used to obtain the numerical results. As shown in Figure 3.2 the stabilization technique removes the linear growth and seems not to alter significantly the PML properties.

Figure 3.1: Late time instability in the PML region for a parallel plate waveguide example. The PML parameters were set to \( m = 4 \), \( \sigma_{\text{max}} = 20 \) and the PML is 4-layers thick \( (\delta = 3.5m) \). (a) Magnitude of the E-field without stabilization at time \( T = 680\delta t \) (b) Magnitude of the E-field with stabilization at time \( T = 680\delta t \).
Figure 3.2: Reflection coefficient for the parallel plate waveguide of Figure 3.3a with and without the stabilization technique. The stabilization, removes instability without significantly altering the PML properties. The PML was 4-layers thick.

As found in [12], the reflection is minimized in FDTD when profiling is applied with $m = 4$ and $\sigma_{max} \approx \frac{m+1}{150\pi\Delta\sqrt{\epsilon_r}}$, where $\Delta$ is the spatial discretization (in meters) along the normal axis of the PML layer. We studied the performance of the PML in DGTD by calculating the reflection coefficient for a simple parallel plate waveguide. The PML was set to be both 4-layers thick, as shown in Figure 3.3a, and was terminated by PEC boundary. We set $m = 4$ and let $\sigma$ to be the only free parameter. The waveguide was excited by a Gaussian pulse $E^{inc} = E_0 e^{-[t-t_0-\hat{k}\cdot(r-r_0)/c]^2/\tau^2}$ with $E_0 = (0,0,1)$, $\hat{k} = (-1,0,0)$, $t_0 = 30.0\text{ns}$, $\tau = 6\text{ns}$. A mesh with average edge length size $l_{avg} = 1.538m \approx \lambda_{30MHz}/7$ was used to obtain the numerical results. The reflection coefficients are shown in Figure 3.3. Other authors [27] who performed a similar study have reached the same conclusion as to which conductivity distribution will give the minimum reflection coefficient in DGTD.
3.2 Numerical Results

Herein, we present two numerical examples that show the application of the conformal PML in practical applications and also highlight the benefits of the local time-stepping strategy. Finally, for all numerical examples, to provide for the optimal converge rate an upwind flux formulation was used.

3.2.1 Coated Sphere

In this example, we calculate the RCS of a coated sphere and compare it with the exact solution. We simulated a coated sphere with inner radius of \( a = 3.0m \), the outer radius is \( b = 3.25m \) and the coating has \( \epsilon_r = 2.0 \) and \( \mu_r = 1.0 \). In this case, the generated mesh has an average edge length \( l_{avg} \approx \frac{\lambda}{7} \) at 300 MHz in free space. The PML/air boundary is set 0.5\( m \) from the coating. The PML has a constant profile...
of $\sigma = 0.02$ and a thickness of $0.4m$. The mesh consisted of 2,242,037 elements of order $p = 1$ and required 26.5GB of memory. The coated sphere is illuminated by a Gaussian pulse $\mathbf{E}^{inc} = \mathbf{E}_0 e^{-\left[ (t-t_0 - \hat{k}(\mathbf{r} - \mathbf{r}_0))/c \right]^2/\tau^2}$ with $\mathbf{E}_0 = (1, 0, 0)$, $\hat{k} = (0, 0, 1)$, $t_0 = 5.0ns$, $\tau = 0.6ns$. By employing the Fourier transform of the time-dependent electric and magnetic currents on the sphere’s surface, we can evaluate the RCS at multiple frequencies. Figure 3.5 shows the computed RCS at the frequency of 300 MHz. A good agreement is observed between the numerical solution and the analytical solution provided by the Mie series. The RCS error defined by (3.10) was computed to be $RCS_{error} = 9.02 \times 10^{-2}$.

$$RCS_{error} = \frac{\sqrt{\int_0^{2\pi} \int_0^\pi |\sigma_{Mie}(\theta, \phi) - \sigma_{Num}(\theta, \phi)|^2 \sin \theta d\theta d\phi}}{\sqrt{\int_0^{2\pi} \int_0^\pi |\sigma_{Mie}(\theta, \phi)|^2 \sin \theta d\theta d\phi}}$$

(3.10)

Figure 3.4: Snapshot of the magnitude of the electric field distribution for the coated sphere.
Figure 3.5: RCS(dB) of a coated sphere with inner radius $a = 3.0m$ and outer radius $b = 3.25m$ at 300 MHz.(blue-DGTD, pink-Mie series). (a) H-plane (b) E-plane.

3.2.2 Scattering from a F-16 aircraft

In this example, we consider the numerical simulation of the scattering from an F-16 aircraft. The aircraft is illuminated by a Gaussian pulse $E^{inc} = E_0 e^{-\frac{(t-t_0-\hat{k}(r-r_0))^2}{\tau^2}}$ with $E_0 = (0, 0, 1)$, $\hat{k} = (1, 0, 0)$, $t_0 = 4.0ns$, $\tau = 0.6ns$ and $r_0 = (8.78, 0, 0) m$. A partial view of the surface mesh is shown in Figure 3.6. Let us note, that the modeling of the F-16 also includes the dielectric radome and the glass canopy as shown in Figure 3.6. The generated mesh has an average edge length $l_{avg} \approx \frac{\lambda}{5}$ at 300 MHz, which is the 3-dB frequency of the incident pulse. The total number of tetrahedra is 1,656,676 and the order of basis was $p = 1$ for all elements. Moreover, the total number of DOF is 52,641,744 with 13,213,392 DOF in the PML region and the total memory was 18GB. The PML is set to be 4-layers thick and has a constant profile of $\sigma = 0.02$. In the outer surface of the PML region, a first order ABC was applied to reduce the reflection due to the finite thickness of the PML. The mesh generated in this case
is an unstructured locally refined mesh with strong element-size disparities, which leads to a minimum time step of $\delta t_{\text{min}} = 7.50 \times 10^{-13}\text{s}$ and a maximum time step of $\delta t_{\text{max}} = 5.33 \times 10^{-11}\text{s}$. The application of the local time-stepping strategy is shown in Table 3.1. The elements are partitioned into 4 classes. Only 0.015% of the elements are the really small elements, that would have dominated in the stability condition and increased significantly the overall CPU time. This simulation was performed using OpenMP parallelization for shared memory architecture and the number of processors was $N_p = 6$. For this example the application of the local time-stepping will provide a gain of approximately 15 times in the CPU time as shown in Table 3.2. Let us note, that in the non local time stepping case the CPU time is estimated time based on the average iteration time. From Table 3.2, one can clearly see the advantages of the

### Table 3.1: Element Partitioning by classes for the F16-aircraft

<table>
<thead>
<tr>
<th>#Class</th>
<th>#Elements</th>
<th>$\delta t$ (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>252</td>
<td>6.002e-13s</td>
</tr>
<tr>
<td>1</td>
<td>36,158</td>
<td>1.801e-12s</td>
</tr>
<tr>
<td>2</td>
<td>395,760</td>
<td>5.402e-12s</td>
</tr>
<tr>
<td>3</td>
<td>1,224,506</td>
<td>1.621e-11s</td>
</tr>
</tbody>
</table>

### Table 3.2: CPU time gain obtained with the local time-stepping versus the leap-frog scheme

<table>
<thead>
<tr>
<th>Solution time with LTS($N_p = 6$)</th>
<th>88.38 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution time without LTS(Standard leap-frog, $N_p = 6$)</td>
<td>1,301 hours</td>
</tr>
<tr>
<td>CPU Gain with LTS</td>
<td>14.7245</td>
</tr>
</tbody>
</table>

in Table 3.1. The elements are partitioned into 4 classes. Only 0.015% of the elements are the really small elements, that would have dominated in the stability condition and increased significantly the overall CPU time. This simulation was performed using OpenMP parallelization for shared memory architecture and the number of processors was $N_p = 6$. For this example the application of the local time-stepping will provide a gain of approximately 15 times in the CPU time as shown in Table 3.2. Let us note, that in the non local time stepping case the CPU time is estimated time based on the average iteration time. From Table 3.2, one can clearly see the advantages of the
local time-stepping algorithm for meshes with strong element-size disparities, which are commonly found in real-life applications. Finally, the field distributions for the electric field at different times is given in Figure 3.7. One can notice the significant scattering created by the air intake as well as by the nose and the cockpit regions.

![Figure 3.6: (a) Surface mesh of the F16-aircraft. (b) Geometry of F16-aircraft together with the PML truncation. The pink region represents the conformal PML used to truncate the computational domain. The green and red regions correspond to the glass canopy and dielectric radome accordingly.](image)

### 3.3 Discussion

In this chapter, we presented an approach to incorporate a conformal Perfectly Matched Layer within the IPDGT D method. From a practical point of view, a conformal PML is more flexible and convenient to model compared to a Cartesian PML. Moreover, because it conforms to the geometry of the problem, we can possibly reduce the buffer space between the structure and the truncation boundary and thus reduce
Figure 3.7: Electric field distribution on the surface of the aircraft at different time locations. (a) Top view. (b) Bottom view. (c)-(d) Snapshots of the magnitude of the electric field distribution in the computational domain. The F-16 aircraft is illuminated by a Gaussian pulse, on-nose incidence.
the number of unknowns. Furthermore, we studied and discussed the issues concerning the PML’s performance and late time instability and present our conclusions. Additionally, some numerical examples were presented to illustrate and validate our approach. The multi-scale nature of some of the presented examples illustrate the ability of a local time-stepping strategy to significantly reduce the solution time. Finally, in the chapters to follow, we will demonstrate how to couple the LTS strategy with distributed parallelization to further increase the efficiency of our methodology.
Chapter 4: Combined EM/Circuit Simulation in Interior Penalty Discontinuous Galerkin Time-Domain Methods

In this chapter, we further enrich the capabilities of the developed IPDGT D by proposing a methodology to model a resistor, a capacitor and an inductor as a planar impedance surface within the IPDGT D framework. This capability, can be of practical importance since EM applications often include lumped elements. Because the lumped elements used in practical applications are small compared to the wavelength, we can assume that the electric and magnetic fields are constant on the surface of the lumped element. In our approach, we start from the voltage and current relationships and derive the equivalent relationships that describe each of the $R, L, C$ in terms of the electric and magnetic fields. Next, these field expressions are weakly enforced through the IP formulation.

Moreover, since no auxiliary variables are introduced to achieve the EM to circuit coupling, it is convenient to integrate the modeling of lumped elements into the LTS strategy and no substantial extra cost is added to memory or CPU time. Furthermore, by numerical experiments we show that for the proposed approach, the stability condition of the loaded case is the same as the one of the unloaded case. This is practically important, since if the opposite was true, it could lead to very small values
for the time step $\delta t$ and increase the CPU time. Finally, some numerical examples illustrate the validity of our approach.

4.1 IPDGTLD Formulation-Lumped Elements

In the following, we use $w$ to denote the width and $l$ to denote the length of the impedance surface. To derive our formulation, we assume two elements as shown in Figure 4.1 where $\partial K_i \cap \partial K_j$ is now a part of an impedance surface that models a resistor, a capacitor or an inductor surface denoted by $\Gamma_R$, $\Gamma_C$ and $\Gamma_L$ respectively.

![Figure 4.1: Geometrical illustration of the impedance surface used in the modeling of $R, L, C$ elements.](image)

4.1.1 Resistor

Space Discretization

For a resistor we have the well known relationship $I_R = V_R/R$ on $\Gamma_R$. To model the resistor as an impedance surface we have to apply the correct conditions for the electric and magnetic fields on that surface. To model this electric current we apply equation (4.1) on $\Gamma_R$ for element $K_i$ and (4.3) on $\Gamma_R$ for element $K_j$. Moreover, since there are no magnetic currents on the resistor’s surface the electric field needs to be
tangentially continuous across the resistor surface. In other words we need to enforce $V_{Ri} = V_{Rj}$ and that is done with equations (4.2), (4.4). In this way, we obtain the following relationships on $\Gamma_R$,

\[
\begin{align*}
\text{Element } K_i & : \hat{n}_i \times H_i + \hat{n}_j \times H_j = -\frac{l}{Rw} \pi_x(E_i) \\
\text{Element } K_j & : \hat{n}_i \times H_i + \hat{n}_j \times H_j = -\frac{l}{Rw} \pi_y(E_j) \\
\hat{n}_i \times E_i + \hat{n}_j \times E_j &= 0 \\
\hat{n}_i \times E_i + \hat{n}_j \times E_j &= 0
\end{align*}
\]

The relationships (4.1)-(4.4) will be enforced weakly through the IP approach in a manner similar to Eqs. (2.8)-(2.9) were enforced. In this way, we get the following residuals

\[
\begin{align*}
\text{Residuals in } K_i & : R_{\Gamma_R}^{(1)} = [H]_\gamma + \frac{l}{Rw} \pi_x(E_i) \\ & \in \mathbf{H}^{\text{div}, \gamma}(\Gamma_R) \\
\text{Residuals in } K_j & : R_{\Gamma_R}^{(2)} = [E]_\gamma \\ & \in \mathbf{H}^{\text{div}, \gamma}(\Gamma_R)
\end{align*}
\]

\[
\begin{align*}
\text{Residuals in } K_i & : R_{\Gamma_R}^{(3)} = [H]_\gamma + \frac{l}{Rw} \pi_y(E_j) \\ & \in \mathbf{H}^{\text{div}, \gamma}(\Gamma_R) \\
\text{Residuals in } K_j & : R_{\Gamma_R}^{(2)} = [E]_\gamma \\ & \in \mathbf{H}^{\text{div}, \gamma}(\Gamma_R)
\end{align*}
\]

The residuals, $R_{\Gamma_R}^{(1),(3)}$ and $R_{\Gamma_R}^{(2)}$ represent the surface error electric and magnetic currents, respectively. The residual, $R_{\Gamma_R}^{(2)}$ should be tested with a surface $\mathbf{H}$ to give rise to the energy density $\mathbf{H} \cdot \mathbf{M}^{\text{err}}$. The surface magnetic field lies in $\mathbf{H}(\text{curl}, \Gamma_R)$ and thus, we choose $\pi_y(w)$ as testing functions. Moreover, both $R_{\Gamma_R}^{(1)}$ and $R_{\Gamma_R}^{(3)}$ should be tested with a surface $\mathbf{E}$ to form $\mathbf{E} \cdot \mathbf{J}^{\text{err}}$. The surface electric field lies in $\mathbf{H}(\text{curl}, \Gamma_R)$ and thus, we choose $\pi_x(v)$ as testing functions. Next, to complete the IP formulation,
(2.17) is replaced by (4.9) for the two elements sharing the resistor surface:

\[
(w_i, R_{K_i}^{(1)})_{K_i} - (v_i, R_{K_i}^{(2)})_{K_i} + (w_j, R_{K_j}^{(3)})_{K_j} - (v_j, R_{K_j}^{(4)})_{K_j}
\]

\[
+ \frac{1}{2} \langle \pi_t(v_i), R_{\Gamma R}^{(1)} \rangle_{\Gamma R} + \frac{1}{2} \langle \pi_t(v_j), R_{\Gamma R}^{(3)} \rangle_{\Gamma R} - \frac{1}{2} \langle \pi_t(w_i) + \pi_t(w_j), R_{\Gamma R}^{(2)} \rangle_{\Gamma R} = 0
\]

(4.9)

**Time Discretization**

Separating the \( w, v \) testing in (4.9), the semi-discrete system for element \( K_i \) can then be written as

\[
M_{\epsilon} \frac{\partial e_i}{\partial t} = S_{\epsilon} h_i - F^{ij}_{\epsilon} e_i - \frac{l}{R_w} B^R_{\epsilon} e_i - F^{ij}_{\epsilon} h_j \left|_{t=n+\frac{1}{2}} \right.
\]

(4.10)

\[
M_{\mu} \frac{\partial h_i}{\partial t} = -S_{h} e_i + F^{ij}_{h} e_i + F^{ij}_{h} e_j \left|_{t=n+1} \right.
\]

(4.11)

where \( e_{i(j)} \) and \( h_{i(j)} \) are the coefficient vectors for the electric and magnetic field, respectively, and \( (B^R_{\epsilon})_{nm} = \frac{1}{2} \int_{F^R_{h}} \pi_t(v_{in}) \cdot \pi_t(v_{im}) \) ds. A leap-frog scheme is used for discretization in time. The first derivatives are approximated using the central differences. Moreover, we use an average approximation \( e_i^{n+\frac{1}{2}} \approx \frac{e_i^n + e_i^{n+1}}{2} \). Since a rigorous analysis of the stability condition for the loaded case is not yet available by the authors, we study the effect of the resistor to the stability of the method by means of numerical examples as shown in Figure 4.3. A parallel plate waveguide \((\epsilon_r = \mu_r = 1)\) of width \( a = 4m \) and height \( b = 4m \), loaded with a resistor was used as shown in Figure 4.2. A time harmonic TEM wave at 30 MHz was used as the port excitation. The corresponding transmission line model is shown in Fig 4.2.

For this model an exact solution can be calculated for the voltages as \( V^{tot}(x,t) = (V_0 e^{-\gamma x} + \Gamma^{ref} V_0 e^{\gamma x}) e^{j\omega t} \) with \( \gamma = j\beta \) and \( \Gamma^{ref} = (Z||Z_0 - Z_0)/(|Z||Z_0 + Z_0) \). Three different values \( R = 10^{-9} \text{ Ohm}, R = 100 \text{ Ohm} \) and \( R = 10^9 \text{ Ohm} \), that have strong variation were studied. The width of the impedance surface is \( w = 4m \) and the height
is \( l = 4\text{m} \), same to the parallel plate waveguide dimensions. In all three cases, the same mesh and consequently the same \( \delta t \) was used. Moreover, in all three cases, the \( \delta t \) was calculated using the stability condition for the unloaded case given in subsection 2.2.4 of Chapter 2, which has no dependence on the value of \( R \). The simulation was performed for 17 periods of time and no instability was observed for all three cases. Therefore, for the proposed average approximation the stability condition between the loaded and the unloaded case remains unchanged. Furthermore, in Figure 4.3 we show the calculated numerical voltage at the input port for the last 7 periods of time. A good agreement is observed when comparing with the analytical solution. Moreover, one can notice the correct behavior when the model approaches the OC case in Fig 4.3 (b) and SC case in Figure 4.3 (d) respectively. Finally, the fully discrete system of equations can be written as

\[
(M_{\epsilon} + \frac{\delta t l}{2Rw} B_{e}^{R}) e_{i}^{n+1} = (M_{\epsilon} - \frac{\delta t l}{2Rw} B_{e}^{R}) e_{i}^{n} + \delta t (S_{e} - F_{e}^{ii}) h_{i}^{n+\frac{1}{2}} - \delta t F_{e}^{ij} h_{j}^{n+\frac{1}{2}} \tag{4.12}
\]

\[
M_{\mu} h_{i}^{n+\frac{1}{2}} = M_{\mu} h_{i}^{n+\frac{1}{2}} + \delta t (-S_{h} + F_{h}^{ii}) e_{i}^{n+1} + \delta t F_{h}^{ij} e_{j}^{n+1} \tag{4.13}
\]

### 4.1.2 Capacitor

**Space Discretization**

For the capacitor, we follow the same steps as in the resistor’s case, starting from the fundamental relationship \( I_C = C \frac{dV_C}{dt} \) on \( \Gamma_C \). To model this electric current we apply equation (4.14) on \( \Gamma_C \) for element \( K_i \) and (4.16) on \( \Gamma_C \) for element \( K_j \). Moreover, since there are no magnetic currents on the capacitor’s surface the electric field needs to be tangentially continuous across the capacitor’s surface. In other words we need to enforce \( V_{Ci} = V_{Cj} \) with equations (4.15), (4.17). In this way, we obtain the following relationships on \( \Gamma_C \),
Figure 4.2: A parallel plate waveguide ($\epsilon_r = \mu_r = 1$) of width $a = 4m$ and height $b = 4m$, loaded with a resistor, capacitor and inductor was used to study the effect of the proposed resistor, capacitor and inductor formulation on the stability condition of the IP-DGTD method.

As before, Eqs. (4.14)-(4.17) will be enforced weakly through the IP approach. In this way, we get the following residuals

Residuals in $K_i$

$$R_{iC}^{(1)} = \lbrack [H] \rbrack_{\gamma} + \frac{Cl}{w} \frac{\partial(\pi_r(E_i))}{\partial t}$$

Function Space

$$\in H(\text{div}, \Gamma_C) \quad (4.18)$$

$$R_{iC}^{(2)} = \lbrack [E] \rbrack_{\gamma}$$

$$\in H(\text{div}, \Gamma_C) \quad (4.19)$$
Residuals in $K_j$ Function Space

$$R_{\Gamma C}^{(3)} = [H]_\gamma + \frac{Cl}{w} \frac{\partial (\pi_\tau(E_j))}{\partial t} \in H(\text{div}, \Gamma_C)$$ (4.20)

$$R_{\Gamma C}^{(2)} = [E]_\gamma \in H(\text{div}, \Gamma_C)$$ (4.21)

The residuals, $R_{\Gamma C}^{(1),(3)}$ and $R_{\Gamma C}^{(2)}$, represent the surface error electric and magnetic currents respectively. The residual, $R_{\Gamma C}^{(2)}$, should be tested with a surface $H$ to give rise to the energy density $H \cdot M_{err}$. The surface magnetic field lies in $H(\text{curl}, \Gamma_C)$ and thus, we choose $\pi_\tau(w)$ as testing functions. Moreover, $R_{\Gamma C}^{(1)}$ and $R_{\Gamma C}^{(3)}$ should be tested with a surface $E$ to form $E \cdot J_{err}$. The surface electric field lies in $H(\text{curl}, \Gamma_C)$ and thus, we choose $\pi_\tau(v)$ as testing functions. The IP formulation for the two elements sharing the capacitor surface now reads:

$$(w_i, R_{K_j}^{(1)})_{K_i} - (v_i, R_{K_j}^{(2)})_{K_i} + (w_j, R_{K_j}^{(3)})_{K_j} - (v_j, R_{K_j}^{(4)})_{K_j}$$

$$+ \frac{1}{2} \langle \pi_\tau(w_i), R_{\Gamma C}^{(1)} \rangle_{\Gamma_C} + \frac{1}{2} \langle \pi_\tau(v_j), R_{\Gamma C}^{(3)} \rangle_{\Gamma_C} - \frac{1}{2} \langle \pi_\tau(w_i) + \pi_\tau(w_j), R_{\Gamma C}^{(2)} \rangle_{\Gamma_C} = 0$$

(4.22)

**Time Discretization**

Separating the $w$, $v$ testings in (4.22), the semi-discrete system for element $K_i$ can then be written as

$$M_e \frac{\partial e_i}{\partial t} = S_e h_i - F_{ei}^{ij} h_i - \frac{Cl}{w} B_{ei}^{C} \frac{\partial e_i}{\partial t} - F_{ei}^{ij} h_j \big|_{t=n+\frac{1}{2}}$$ (4.23)

$$M_\mu \frac{\partial h_i}{\partial t} = -S_h e_i + F_h^{ii} e_i + F_h^{ij} e_j \big|_{t=n+1}$$ (4.24)

where $(B_e^C)_{nm} = \frac{1}{2} \int_{F_h^C} \pi_\tau(v_{in}) \cdot \pi_\tau(v_{im}) \, ds$. Again, the first order time derivatives are approximated with central differences. Similar to the resistor’s case, three different values of $C$: $C = 10^{-13}$ F, $C = 10^{-11}$ F and $C = 10^{0}$ F, that have strong variation were studied as shown in Figure 4.4. Likewise to the resistor example, the same
parallel plate waveguide was used and a time harmonic TEM wave at 30 MHz was set as the port excitation. In all three cases the $\delta t$ used, was the one provided by the stability condition for the unloaded case given in subsection 2.2.4 of Chapter 2, which has no dependence on the value of $C$. The width of the impedance surface is $w = 4\text{m}$ and the height is $l = 4\text{m}$, same to the parallel plate waveguide dimensions. Again, the simulation was performed for 17 periods of time and no instability was observed for all three cases. Therefore, for the proposed time discretization the stability condition between the loaded and the unloaded case remains unchanged. Moreover, in Fig 4.4 we show the calculated numerical voltage at the input port for the last 7 periods of time. A good agreement is observed when compared with the analytical solution. Moreover, one can notice the correct behavior when the model approaches the OC case in Fig 4.4 (b) and SC case in Figure 4.4 (d) respectively. Using a leap-frog scheme and central differences for the first derivatives, the fully discrete system is written as:

$$
(M_\epsilon + \frac{C_\ell}{w} B_e^C) e_t^{n+1} = (M_\epsilon + \frac{C_\ell}{w} B_e^C) e_t^n + \delta t(S_e - F_{eii}^n) h_i^{n+\frac{1}{2}} - \delta t F_{eij}^n h_j^{n+\frac{1}{2}} \quad (4.25)
$$

$$
M_h h_i^{n+\frac{3}{2}} = M_h h_i^{n+\frac{1}{2}} + \delta t(-S_h + F_{hii}^n) e_t^{n+1} + \delta t F_{hij}^n e_j^{n+1} \quad (4.26)
$$

### 4.1.3 Inductor

**Space Discretization**

For an inductor we have an electric current given by $I_L = \frac{1}{L} \int_0^t V_L \, dt$ on $\Gamma_L$. To model this electric current we apply equation (4.27) on $\Gamma_L$ for element $K_i$ and (4.29) on $\Gamma_L$ for element $K_j$. Since there are no magnetic currents on the inductor’s surface the electric field needs to be tangentially continuous. In other words we need to enforce $V_{Li} = V_{Lj}$ and that is done with equations (4.28) and (4.30). In this way, we obtain the following relationships on $\Gamma_L$:
Again, (4.27)-(4.30) will be enforced weakly through the IP approach. We proceed with the following residuals,

Residuals in \( K_i \)

\[
R_{\Gamma_L}^{(1)} = [H]_\gamma + \frac{l}{Lw} \int_0^t \pi_\tau(E_i) \, dt \quad \in H(\text{div}_r, \Gamma_L) \quad (4.31)
\]

\[
R_{\Gamma_L}^{(2)} = [E]_\gamma \quad \in H(\text{div}_r, \Gamma_L) \quad (4.32)
\]

Residuals in \( K_j \)

\[
R_{\Gamma_L}^{(3)} = [H]_\gamma + \frac{l}{Lw} \int_0^t \pi_\tau(E_j) \, dt \quad \in H(\text{div}_r, \Gamma_L) \quad (4.33)
\]

\[
R_{\Gamma_L}^{(2)} = [E]_\gamma \quad \in H(\text{div}_r, \Gamma_L) \quad (4.34)
\]

The residuals, \( R_{\Gamma_L}^{(1),(3)} \) and \( R_{\Gamma_L}^{(2)} \) represent the surface error electric and magnetic currents respectively. The residual, \( R_{\Gamma_L}^{(2)} \) should be tested with a surface \( H \) to give rise to the energy density \( H \cdot M^{err} \). The surface magnetic field lies in \( H(\text{curl}_r, \Gamma) \) and thus, we choose \( \pi_\tau(w) \) as testing functions. Moreover, \( R_{\Gamma_L}^{(1)} \) and \( R_{\Gamma_R}^{(3)} \) should be tested with a surface \( E \) to form \( E \cdot J^{err} \). The surface electric field lies in \( H(\text{curl}_r, \Gamma_L) \) and thus, we choose \( \pi_\tau(v) \) as testing functions. The IP formulation for the two elements sharing the inductor surface is written as:

\[
(w_i, R_{K_j}^{(1)})_{K_i} - (v_i, R_{K_j}^{(2)})_{K_i} + (w_j, R_{K_j}^{(3)})_{K_j} - (v_j, R_{K_j}^{(4)})_{K_j} + \frac{1}{2} \langle \pi_\tau(v_i), R_{\Gamma_L}^{(1)} \rangle_{\Gamma_L} + \frac{1}{2} \langle \pi_\tau(v_j), R_{\Gamma_L}^{(3)} \rangle_{\Gamma_L} - \frac{1}{2} \langle \pi_\tau(w_i) + \pi_\tau(w_j), R_{\Gamma_L}^{(2)} \rangle_{\Gamma_L} = 0
\]

(4.35)
Time Discretization

The semi-discrete system for element $K_i$ can then be written as

$$M_i \frac{\partial \mathbf{e}_i}{\partial t} = S_i \mathbf{h}_i - \mathbf{F}^{ii}_e \mathbf{h}_i - \frac{l}{Lw} B^L_e \int_0^t \mathbf{e}_i \, dt - \mathbf{F}^{ij}_e \mathbf{h}_j \mid_{t=n+\frac{1}{2}} \tag{4.36}$$

$$M_i \mu \frac{\partial \mathbf{h}_i}{\partial t} = -S_i \mathbf{e}_i + \mathbf{F}^{ii}_h \mathbf{e}_i + \mathbf{F}^{ij}_h \mathbf{e}_j \mid_{t=n+1} \tag{4.37}$$

where $(B^L_e)_{nm} = \frac{1}{2} \int_{F^L_h} \pi_T(v_{im}) \cdot \pi_T(v_{im}) \, ds$. The integration in time is approximated by midpoint integration. Namely, $\int_{0}^{n+1/2} e_{i(j)} \, dt \approx \delta t \sum_{k=0}^{n} \frac{e_{i}^{k} + e_{i}^{k+1}}{2}$. The integration is performed up to $(n+1)dt$. Likewise to the resistor’s and capacitor’s example, the same parallel plate waveguide with a time harmonic TEM wave at 30 MHz was used to study the effect of the inductor on the stability. Similarly, three different values of $L$: $L = 10^{-14}$ H, $L = 10^{-5}$ H and $L = 10^9$ H, with strong variations were studied as shown in Figure 4.5. In all three cases, the $\delta t$ used was the one provided by the stability condition for the unloaded case, given in subsection 2.2.4 of Chapter 2, which has no dependence on the value of $L$. The width of the impedance surface is $w = 4m$ and the height is $l = 4m$, same to the parallel plate waveguide dimensions. Again, the simulation was performed for 17 periods of time and no instability was observed for all three cases. Therefore, for the proposed time discretization of the inductor the stability condition between the unloaded and loaded case remains the same. Moreover, in Figure 4.5 we show the calculated numerical voltage at the input port for the last 7 periods of time calculated. A good agreement is observed when comparing with the analytical solution. Moreover, one can notice the correct behavior when the model approaches the OC case in Fig 4.5 (d) and SC case in Figure 4.5 (b) respectively. In this way the fully discrete system of equations can be written as,
\[(M_e + \frac{\delta t^2}{2L_w} B_e^L) e_i^{n+1} = M_e e_i^n + \frac{\delta t^2}{L_w} B_e^L \hat{e}_i^n + \delta t(S_e - F_e e_i^{n+1} - \delta t F_e^{ij} e_j^{n+1}) (4.38)\]

\[M_p h_i^{n+\frac{3}{2}} = M_p h_i^{n+\frac{1}{2}} + \delta t(-S_h + F_h^{ij} e_j^{n+1}) e_i^{n+1} (4.39)\]

where \(\hat{e}_i^n = \sum_{k=0}^n e_i^k\).

### 4.1.4 Interior Port

**Space Discretization**

An interior port that uses the resistive voltage source model is used to provide the excitation. The total current is expressed as \(I^{tot} = \frac{1}{R_s} (V^{tot} - V^{inc})\) on \(\Gamma_P\), \(V^{inc}\) is the excitation voltage and \(R_s\) the source resistance. To model this electric current we apply equation (4.40) on \(\Gamma_P\) for element \(K_i\) and (4.42) on \(\Gamma_P\) for element \(K_j\). Because there are no magnetic currents on the port surface, the electric field needs to be tangentially continuous across surface. In other words we need to enforce \(V_{Pi} = V_{Pj}\) by applying equations (4.41), (4.43). In this way, we obtain the following relationships on \(\Gamma_P\),

**Element \(K_i\)**

\[\hat{n}_i \times H_i + \hat{n}_j \times H_j = -\frac{l(\pi_\tau(E_i - E_i^{inc}))}{R_s w} (4.40)\]

\[\hat{n}_i \times E_i + \hat{n}_j \times E_j = 0 (4.41)\]

**Element \(K_j\)**

\[\hat{n}_i \times H_i + \hat{n}_j \times H_j = -\frac{l(\pi_\tau(E_j - E_j^{inc}))}{R_s w} (4.42)\]

\[\hat{n}_i \times E_i + \hat{n}_j \times E_j = 0 (4.43)\]

We get the following residuals:

**Residuals in \(K_i\)**

\[R_{\Gamma_P}^{(1)} = [H]_\gamma + \frac{l(\pi_\tau(E_i - \pi_\tau(E_i^{inc}))}{R_s w} \in H(\text{div}_\tau, \Gamma_P) (4.44)\]

\[R_{\Gamma_P}^{(2)} = [E]_\gamma \in H(\text{div}_\tau, \Gamma_P) (4.45)\]
Residuals in $K_j$

$$R^{(3)}_{\Gamma_P} = [H]_\gamma + \frac{l(\pi_\tau(E_j) - \pi_\tau(E_{j}^{inc})}{R_s w}$$

\(\in H(\text{div}_\tau, \Gamma_P)\) \hspace{1cm} (4.46)

$$R^{(2)}_{\Gamma_P} = [E]_\gamma$$

\(\in H(\text{div}_\tau, \Gamma_P)\) \hspace{1cm} (4.47)

The residuals, $R^{(1),(3)}_{\Gamma_P}$ and $R^{(2)}_{\Gamma_P}$ represent the surface error electric and magnetic currents, respectively. The residual, $R^{(2)}_{\Gamma_P}$ should be tested with a surface $H$ to give rise to the energy density $H \cdot M_{\text{err}}$. The surface magnetic field lies in $H(\text{curl}_\tau, \Gamma)$ and thus, we choose $\pi_\tau(w)$ as testing functions. Moreover, $R^{(1)}_{\Gamma_P}$ and $R^{(3)}_{\Gamma_P}$ should be tested with a surface $E$ to form $E \cdot J_{\text{err}}$. The surface electric field lies in $H(\text{curl}_\tau, \Gamma_P)$ and thus, we choose $\pi_\tau(v)$ as testing functions. The IP formulation can now be written as

$$\left(w_i, R^{(1)}_{K_j}\right)_{K_i} - \left(v_i, R^{(2)}_{K_j}\right)_{K_i} + \left(w_j, R^{(3)}_{K_j}\right)_{K_j} - \left(v_j, R^{(4)}_{K_j}\right)_{K_j}$$

$$+ \frac{1}{2} \langle \pi_\tau(v_i), R^{(1)}_{\Gamma_P}\rangle_{\Gamma_P} + \frac{1}{2} \langle \pi_\tau(v_j), R^{(3)}_{\Gamma_P}\rangle_{\Gamma_P} - \frac{1}{2} \langle \pi_\tau(w_i) + \pi_\tau(w_j), R^{(2)}_{\Gamma_P}\rangle_{\Gamma_P} = 0$$

\(\text{(4.48)}\)

**Time Discretization**

The semi-discrete system within each element $K_i$ is written as

$$\frac{\partial}{\partial t} e_i = S_i h_i - F^{ii}_e h_i - \frac{l}{2R_s w} B^{R_e} e_i - F^{ij}_e h_j + e^{inc}_i \big|_{t=n+\frac{1}{2}}$$

\(\text{(4.49)}\)

$$\frac{\partial}{\partial t} h_i = -S_i e_i + F^{ii}_h e_i + F^{ij}_h e_j \big|_{t=n+1}$$

\(\text{(4.50)}\)

where $(B^{R_e})_{nm} = \frac{1}{2} \int_{\Gamma_P} \pi_\tau(v_{in}) \cdot \pi_\tau(v_{im}) \, ds$, $e^{inc}$ is obtained by integrating $(B^{R_e})_{nm} \pi_\tau(v_{in}) \cdot \pi_\tau(E^{inc}) \, ds$. We use a leap-frog scheme and the average approximation $e_i^{n+\frac{1}{2}} \approx \frac{e^i_n + e^{n+1}_i}{2}$. For an average approximation the stability condition between the loaded and the unloaded case remains unchanged since, the voltage source is identical to the resistor’s case with the addition of the excitation terms. In this way, the fully discrete system of
equations can be written as

\[
(M_e + \frac{\delta t}{2R_e w} B_e^{R_e}) e_i^{n+1} = (M_e - \frac{\delta t}{2R_e w} B_e^{R_e}) e_i^n + \delta t (S_e - F_i^{R_e}) h_i^{n+\frac{1}{2}}
\]

\[-\delta t F_i^{ij} h_j^{n+\frac{1}{2}} + \delta t (e_i^{inc})^{n+\frac{1}{2}} \tag{4.51}\]

\[
M_h h_i^{n+\frac{1}{2}} = M_h h_i^{n+\frac{1}{2}} + \delta t (-S_h + F_h^{ij}) e_i^{n+1} + \delta t F_h^{ij} e_j^{n+1} \tag{4.52}\]

4.2 Numerical Results

In this section, we present the results of the numerical experiments performed to validate and illustrate the capabilities of the method. In the example below we used tetrahedral meshes and first order polynomials and an upwind flux formulation.

4.2.1 Interconnect device with Passive Lumped Elements

In this example, we consider the numerical simulation of an interconnect device whose geometry is given in Figure 4.6. The interconnect consists of 4 metal layers. All the metal layers exists inside a dielectric with \( \epsilon_r = 3.8 \). The whole structure is enclosed in a box whose outer surface is a first order ABC for the domain truncation. Two SMT components are connected in metal layer-1 at positions SMT1 and SMT2 and a third SMT component is connected in metal layer-4 at position SMT3 as shown in Figure 4.6. The SMT components are modeled as an equivalent series RLC network as shown in Figure 4.7. The series RLC network was implemented by cascading in series three impedance surfaces as shown in Figure 4.7 (a) that implement \( R, L \) or \( C \) as it was described in section 4.1. For SMT1, we have \( R_1 = 0.2 \Omega, C_1 = 6.2 pF \) and \( L_1 = 0.06 nH \). For SMT2, \( R_2 = 0.2 \Omega, C_2 = 3.6 pF \) and \( L_2 = 0.06 nH \). For SMT3, \( R_3 = 0.5 \Omega, C_3 = 1000 pF \) and \( L_3 = 0.45 nH \). The device is excited by a Neuman pulse \( N(t) = A_o (\frac{t-t_0}{\tau^2}) e^{-\frac{(t-t_0)}{\tau^2}} \) with \( t_0 = 285 ps, \tau = 75 ps \). The total
number of tetrahedra is 65,676 and the total number of DOF is 1,290,608. The mesh generated in this case is an unstructured locally refined mesh with strong element-size disparities, which leads to a minimum time step of $\delta t_{\text{min}} = 8.6297 \times 10^{-16} \text{s}$ and a maximum time step of $\delta t_{\text{max}} = 1.5804 \times 10^{-13} \text{s}$.

At this point we could note that the main advantages of the proposed method are: (a) the lumped elements are modeled directly using appropriate expressions of the field variables and no auxiliary variables were used to achieve the EM to circuit coupling. In this way the implementation is easy, there is no need to change the implementation of the LTS for the loaded case and no substantial extra cost is added to memory or CPU time (b) the method retains the stability condition of the unloaded case as shown by the numerical examples in section 4.1. This is important since a time step that depends on the values of the lumped elements could penalize the CPU time performance significantly, by leading into very small values for the $\delta t$. The application of the local time-stepping strategy is shown in Table 4.1 and Figure 4.7 (b). The elements are partitioned into 5 classes. Only 0.075% of the elements are the really small elements, that contribute to increase significantly the overall CPU time if a global time step is used. The S-parameters are calculated from the relationships in

Table 4.1: Element Partitioning by classes for the Interconnect

<table>
<thead>
<tr>
<th>#Class</th>
<th>#Elements</th>
<th>$\delta t(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>49</td>
<td>6.904e-16</td>
</tr>
<tr>
<td>1</td>
<td>4687</td>
<td>2.071e-15</td>
</tr>
<tr>
<td>2</td>
<td>38938</td>
<td>6.213e-15</td>
</tr>
<tr>
<td>3</td>
<td>21515</td>
<td>1.864e-14</td>
</tr>
<tr>
<td>4</td>
<td>649</td>
<td>5.592e-14</td>
</tr>
</tbody>
</table>
using voltages as derived in [28]. The voltages were calculated as line integrals of the electric field along the excitation and output ports.

\[ S_{11} = \frac{2V_{1\text{tot}}(\omega) - V_{\text{inc}}(\omega)}{V_{\text{inc}}(\omega)}, \quad S_{21} = \frac{2V_{2\text{tot}}(\omega)}{V_{\text{inc}}(\omega)} \] (4.53)

where \( V_{\text{inc}}(t) \) is the excitation voltage. Moreover, \( V_{1,2}(\omega) \) and \( V_{\text{inc}}(\omega) \) are the Fourier transforms of \( V_{1,2}(t) \) and \( V_{\text{inc}}(t) \) respectively. The S-parameters results are shown in Figures 4.8. A very good agreement is observed between the DGTD data and the measurements for the range of interest 1-6 GHz. Also an excellent agreement is observed between the DGTD data and the frequency domain FEM results. Finally, a

Table 4.2: CPU time gain obtained with the local time-stepping versus the leap-frog scheme

<table>
<thead>
<tr>
<th>Solution time with LTS, ( N_p = 6 )</th>
<th>50 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>No LTS, Standard leap-frog, ( N_p = 6 )</td>
<td>466 hours</td>
</tr>
<tr>
<td>CPU Gain with LTS</td>
<td>8.8</td>
</tr>
</tbody>
</table>

parallelization for shared memory architecture using OpenMP parallelization was used in this example. The simulation was performed on a two quad-core Xeon machine, the number of processors used was \( N_p = 6 \) and the memory used was 597 MB. For this example a standard leap-frog scheme would require a solution time of 466 hours whereas with the local time-stepping the required time is reduced to 50 hours. Therefore, the application the local time-stepping will provide a gain of approximately 8.8 times in the CPU time illustrating its advantages for meshes with strong element-size disparities.
4.3 Discussion

In this chapter, we presented a method to incorporate passive lumped elements such as resistors, capacitors and inductors into the IPDGTD framework. In our methodology the lumped elements are modeled by appropriate expressions of the field variables and without using any auxiliary variables to achieve the EM to circuit coupling. Consequently, with few additions in the implementation we can integrate the loaded case into the LTS strategy and no substantial extra cost is added to memory or CPU time. Moreover, a series of numerical experiments indicate that for the proposed approach, the stability condition of the loaded case is the same as the one of the unloaded case. Hence, the time step does not depend on the values of the lumped elements, which is practically important. If the opposite was true, it could penalize the CPU time performance significantly, by leading into very small values for the $\delta t$. To validate our approach, a 4 metal-layer interconnect device with passive lumped elements was simulated. The application of the local time-stepping strategy provided a gain of approximately 8.78 times in the CPU time. The simulation results show good agreement with both measurement and frequency domain FEM data. Finally, the modeling of active elements (i.e. diodes, transistors, etc.) and/or the coupling with general arbitrary circuits could prove to be an interesting future research direction relative to this chapter.
Figure 4.3: Study of the effect of the resistor in the stability condition for three different values of $R$, with $w = 4\text{m}$ and $l = 4\text{m}$, same to the parallel plate waveguide dimensions. The simulation was performed based on the stability condition of the unloaded case for 17 periods of time and no instability was observed for all three cases. In (b)-(d) we shown the last 7 periods and a good agreement is observed when comparing with the analytical solution. Therefore, for the proposed average approximation the stability condition between the loaded and the unloaded case remains unchanged.
Figure 4.4: Study of the effect of the capacitor in the stability condition for three different values of $C$, with $w = 4m$ and $l = 4m$, same to the parallel plate waveguide dimensions. The simulation was performed based on the stability condition of the unloaded case for 17 periods of time and no instability was observed for all three cases. In (b)-(d) we shown the last 7 periods and good agreement is observed when comparing with the analytical solution. Therefore, for the proposed time discretization the stability condition between the loaded and the unloaded case remains unchanged.
Figure 4.5: Study of the effect of the inductor in the stability condition for three different values of $L$, with $w = 4m$ and $l = 4m$, same to the parallel plate waveguide dimensions. The simulation was performed based on the stability condition of the unloaded case for 17 periods of time and no instability was observed for all three cases. In (b)-(d) we shown the last 7 periods and a good agreement is observed when compared with the analytical solution. Therefore, for the proposed time discretization the stability condition between the loaded and the unloaded case remains unchanged.
Figure 4.6: Geometry of the interconnect (3mm×3mm×0.6mm) and snapshot of the magnitude of the E-field.

Figure 4.7: (a) Modeling of the SMT components as equivalent $R, L, C$ circuit. (b) Partial view of the distribution of the mesh elements after application of the local time stepping algorithm in the interconnect example. In the colorbar the values 0-4 denote class 0-4 respectively.
Figure 4.8: Comparison of magnitude and phase of $S_{11}$ and $S_{12}$ with measurement and our frequency domain FEM code. A very good agreement is observed with both measurement and frequency domain data.
Chapter 5: Parallelization of Interior Penalty Discontinuous Galerkin Time-Domain Method

As already mentioned in the introduction, DG methods are well suited for parallelization. In this chapter, we present our design of a scalable parallel implementation of the IPDGD method to solve the time dependent Maxwell’s equations. Our parallelization is for multi-core CPUs and/or graphics processor units (GPUs) clusters, for distributed memory systems. In the proposed design, we are able to choose between a MPI/CPU or MPI/GPU configuration to launch our simulation. Moreover, in this chapter we emphasize on the MPI/GPU configuration and its comparison against MPI/CPU.

We discuss how to exploit the inherent DGTD parallelism and describe a combined MPI and local time-stepping implementation to merge the benefits of the two strategies. This combination is aimed at increasing efficiency and reducing computational time, especially for multi-scale applications. For the MPI/GPU configuration the CUDA programming model was used. Moreover, for both MPI/CPU and MPI/GPU non-blocking MPI calls that overlap communications across the network were used to increase efficiency. Furthermore, a 10x speedup was observed when using MPI/GPU compared to MPI/CPU (one GPU against one CPU core) for double precision arithmetic. Finally, good scalability with parallelization efficiency of 85% for up to 40
GPUs and 80% for up to 160 CPU cores was achieved on the Ohio Supercomputer Center’s Glenn cluster.

5.1 Introduction and Organization

Practical real-life applications have a high level of complexity and usually result in long simulation times. In order to have efficient numerical simulation times, especially in the time domain, it is desirable to have a numerical method with high parallelism capability that can run on fast hardware. As discussed in earlier chapters, in DG methods the resulting mass matrix is a block diagonal matrix, with the block size equal to the degrees of freedom in the element. Hence the method can lead to a fully explicit time-marching scheme for the solution in time. Moreover, the fully discretized local system of equations (2.31)-(2.32) in Chapter 2 indicate that another feature of the IPDGT methods is that a big part of the computation is local to each element. Additionally, information exchange is required only between neighboring elements regardless of the order of the polynomial approximation and the shape. Thus, DG methods possess what is usually called a locality property which make DG methods very suitable for parallelization.

In this chapter, we present an approach that exploits this locality property to achieve an efficient parallelization. Consequently, DG methods should run on hardware that supports parallelization. At least two possible choices are available in current hardware technology. One is multi-core CPUs and the other is GPUs. For DGTD, current generation multi-core CPUs can offer up to 32 threads (8 quad core sockets) for parallel computation. If more threads are required, multi-threading can be applied, but the operating system will have to switch between threads, and this
could be expensive. Moreover, CPUs offer fast memory access time, but the memory bandwidth is a factor that could limit performance. Another possible candidate for DGTD computation is GPUs. Current generation GPUs, i.e., Tesla 2050, have 14 streaming multi-processors (SMs). Each SM has 32 streaming processors (SPs), or CUDA cores, and can support up to 1,024 threads. Furthermore, in GPUs, memory access time is usually slower than in the CPU, but GPUs also have about an order of magnitude higher memory bandwidth and floating point operation capabilities than their CPU counterparts. Therefore, one can argue that GPUs appear to be a better candidate for DGTD methods.

In the context of DG methods, GPU computing was initially introduced in [21], where a single precision GPU implementation of DGTD was presented. The authors presented a 40x-60x speedup when comparing one GPU versus one CPU core. Furthermore, [15] presented their approach to implementing DGTD in GPU clusters, and the authors reported a 20x speedup in the solution time for a single precision implementation. Also, in [15] the GPU communication was done in a shared memory architecture, since all GPU devices physically reside on the same cluster node and scalability results were reported for up to 8 GPUs. Furthermore, in [16] a multi-rate GPU implementation of DGTD was presented, but there was a restriction to only two classes for time marching. In our approach the number of classes is arbitrary, so no such restriction is applied for the local time stepping. The main contributions of this chapter are summarized in the following:

- We present our approach for an MPI/GPU implementation of DGTD. The proposed approach is applicable in large GPU clusters with distributed-memory architecture. Moreover, we report a 10x speedup compared to CPU clusters (one
GPU against one CPU core) for double precision arithmetic using Quadro FX 5800 cards.

- We combine the local time stepping (LTS) [26] algorithm with MPI/GPU to increase efficiency and reduce the computational time for multi-scale applications.

- We show good scalability and parallelization efficiency of 85% with up to 40 GPUs and 80% with up to 160 CPU cores, on the Glenn Cluster at the Ohio Supercomputer Center.

Finally, the organization of this chapter is as follows: In section 5.2 we discuss the computational layout used to map our IPDGTD to a CUDA framework. Next, in section 5.3 we describe our MPI/GPU implementation for distributed memory architecture. Finally, in section 5.4 we present our performance analysis and give numerical examples to illustrate the potential of the proposed MPI/GPU/LTS strategy.

5.2 CUDA Implementation

In this section we describe the approach we follow on the GPU side of our implementation. Our approach is based on ideas similar to the ones discussed in [21, 15]. The computational layout we follow on the GPU implementation consists mainly of three steps, which are described in the following and summarized in Figure 5.1. In the following, let us denote by $M_i$ the $i^{th}$ partition of an initial large finite element mesh.
5.2.1 FEM Mesh to CUDA Grid Mapping

Firstly, a CUDA-grid is mapped to the finite element mesh of partition $M_i$. According to the CUDA programming model [2], a CUDA grid, $\text{dim}^3 \text{grid}(\text{dim}X, \text{dim}Y, \text{dim}Z)$, can be at most two dimensional ($\text{dim}Z$ must be equal to 1) although declared as $\text{dim}3$. When a CUDA grid with dimensions $\text{dim}X$ and $\text{dim}Y$ is declared, it means that a total of $\text{dim}X \times \text{dim}Y$ CUDA thread blocks will be scheduled to run on the hardware at execution time. Therefore, the dimensions $\text{dim}X$ and $\text{dim}Y$ are chosen such that $\text{dim}X \times \text{dim}Y = N_i$, where $N_i$ is the numbers of elements in partition $M_i$. The requested thread blocks will be assigned by the CUDA model to run in parallel on the available streaming-multiprocessors (SMs) of the GPU device. This concludes the first level of parallelism. By establishing this mapping it is easy to see that we have one thread block for each element, which naturally leads us to the next step. Finally, according to [2] $1 \leq \text{dim}X, \text{dim}Y \leq 65,536$. Hence, for a CUDA grid with
more than 65,536 threads blocks, a two dimensional grid is necessary. However, one should notice also the massive maximum number of 65,536² thread blocks that are allowed to be scheduled to run on the GPU.

5.2.2 Finite Element to CUDA thread block Mapping

Secondly, each thread block is mapped to a finite element. Each thread block is going to be responsible for completely updating one element of the FEM mesh of partition $M_i$. A CUDA thread block can be declared to have three dimensions as $\text{dim3 block}(\text{dimX}, \text{dimY}, \text{dimZ})$. The maximum number of threads per block depends on the compute capability of the GPU device. The Quadro FX 5800 cards we used are of compute capability 1.3 [2]. According to [2], cards of 1.3 capability have a maximum number of threads per block equal to 512. Hence we have that $1 \leq \text{dimX} \times \text{dimY} \times \text{dimZ} \leq 512$. In our case, a one dimensional declaration is sufficient. Therefore, a thread block is chosen as $\text{dim3 block}(1, \text{BlockSize})$ where $\text{BlockSize}$ is equal to the number of DOFs of the element.

5.2.3 DOF to CUDA thread Mapping

After establishing the previous two steps, the third and final step follows naturally. Each thread of a thread block is now responsible for updating one DOF of the finite element. Of course all threads within each thread block are executed in parallel, and this constitutes the second level of parallelism on the GPU side. As mentioned before, $E_{h}^{(i)}(r,t) \approx \sum_{n}^{d_i} e_{in}(t)v_{in}(r)$ and $H_{h}^{(i)}(r,t) \approx \sum_{n}^{d_i} h_{in}(t)w_{in}(r)$ so each thread $t_n$ updates the value of $e_{in}$ and $h_{in}$. 
5.2.4 Description of the leap-frog update kernels

Now that the computational layout is established, we continue our discussion with the description of the kernels used in our implementation. Every thread block performs the matrix-vector multiplications in the update equations (2.31)-(2.32). All the matrix data needed in the update equations are precomputed on the CPU side at pre-processing. Then they are copied to the GPU’s global memory once, before the LTS begins. Memory is also allocated in the GPU’s global memory for the time-stepping vectors. In this way all the computations in the update equations are directly performed on the GPU only. The update at every LTS step is performed by calling one of two functions (LeapFrogE or LeapFrogH) that perform the leap-frog update for the E and H field respectively [26, 8]. Two kernel functions are needed for a LTS update step (E or H), as shown in Figure 5.1. First, a volume kernel \( LE_{\text{Vol}} \) kernel is executed to compute the local contributions. Then, a surface kernel \( LE_{\text{Surf}} \) kernel is launched to compute the neighboring contributions. In both volume and surface kernels, the field data are reused many times in the matrix-vector multiplications. Hence it is beneficial to copy them from global to shared memory, since shared (on chip) memory has much faster access time. Furthermore, the shared memory for the Quadro FX 5800 cards that we use is 16kB for each SM. Since each SM can run at most 8 thread blocks, we have up to 2kB per thread block to store field data which is enough even for high order elements. Thus, both volume and surface kernels will use shared memory for the field data used in the update equations, while the matrix data will be read directly from global memory. Moreover, while the volume kernel is executed, asynchronous copy with \( cudaMemcpyAsync \) is used to copy the excitation data calculated in the CPU to the GPU and overlap copy with computation. This is
achieved by using the CUDA streams functionality as shown Figure 5.1. In this way, the E update is completed. Likewise, two kernels $LH_{Vol\_kernel}$ and $LH_{Surf\_kernel}$ perform the update for the H-field. Furthermore, in the hardware available at OSC (Quadro FX 5800) only one kernel can be executed at a time, so volume and surface kernels are not executed concurrently. However, in more recent GPUs like Tesla 2050, concurrent kernel execution is supported. Features like this can potentially improve even further the performance of CUDA codes. Finally, data are copied from the GPU to the CPU, through the PCIe2 bus, every $m \delta t_N$ of simulation time for post-processing and output to disk, where $m$ is some integer and $N$ is the number of classes.

5.3 MPI/GPU Parallelization

As described in the previous section, the matrix data are stored in the GPU global memory. Consequently, the size of the problem one can solve in a single GPU is limited by the available memory on the device. Hence there is no doubt that only a few real life applications can be solved using a single GPU. The Quadro FX 5800 graphics cards used in this chapter have 4GB (future cards like nVidia’s Tesla 2070 report up to 6GB) of global memory. Therefore, the type of problem one can solve with one GPU is limited. If one wishes to solve problems with large numbers of unknowns in a single GPU, then data must be transferred from the CPU to the GPU and vice versa many times. These transfers through the PCIe bus will eventually become a bottleneck. Therefore, a domain partition strategy for data parallelism on a multi-GPU platform is required to provide solutions to real life applications. Moreover, scalable solution strategies need to be considered. For problems for which one GPU device is enough,
no communication overhead exists and all optimizations can be accounted for in the GPU part of the code. On the other hand, when using multiple GPUs, neighboring partitions need to share data between them to complete their computations. These data exchanges across neighboring partitions need to be addressed properly so that they do not introduce too much latency and restrict scalability. In this chapter we achieve that by using non-blocking MPI calls, which overlap communications over the network and contribute to a scalable implementation.

5.3.1 Combined MPI/GPU/LTS Strategy

In this section we describe our strategy to combine the LTS algorithm with an MPI parallelization. The implementation described below is in some respects a hybrid CPU/GPU implementation. The computationally intensive and also highly parallel parts of the implementation are ported into the GPUs. On the other hand, the less intensive parts, as well as the pre-processing and post-processing, are handled on the CPU side, as shown pictorially in Figure 5.4. Since the most expensive part is actually performing the LTS algorithm, our approach focuses on how to speed up the LTS update using GPUs. Our implementation of the LTS follows the approach in [26]. The LTS algorithm uses two functions LeapFrogE(class $i$, $dt_i$) and LeapFrogH(class $i$, $dt_i$) that perform the time updates in the LTS algorithm. These functions are called in a recursive fashion to update each class. The goal and contribution of our approach is to try to combine the benefits of the LTS approach with the GPU compute capabilities and MPI parallelization. We retain the LTS algorithm unchanged and use multiple processes (CPUs/GPUs) to speed the computations within each class, as shown in Figure 5.2. Our approach is given in the following lines:
• The initial finite element mesh is partitioned using METIS [20] into $M$ partitions, in order to obtain a balanced partition. Next, each partition $M_i$ is then mapped to an MPI process. An simple example is shown in Figure 5.2. Finally, each MPI process can handle a CPU core (MPI/CPU), or a GPU device (MPI/CPU).

• After partitioning, each MPI process will have elements that belong to more than one LTS class. When a LTS update (either LeapFrogE or LeapFrogH) step is performed, all MPI processes having elements that belong to the current class perform the update in a parallel fashion as illustrated in Figure 5.2.

• For every LTS update step, all communications between neighboring partitions are handled with non-blocking MPI calls to improve communication time and scalability.

Figure 5.2: (a) Partitioning and assignment of MPI processes to the partitions. (b)-(c) Proposed MPI/LTS strategy. A number of MPI processes $P_0, P_1, ..., P_N$ work in parallel for every step in the LTS algorithm. Here LE denotes a leap-frog E update and LH denotes a leap-frog H update.
5.3.2 MPI System Design

In this section we continue by describing the coarse-grained level of the parallelization, which is essentially the MPI part of the implementation. For this part, as shown also in Figure 5.3a-5.3b, one has at least two possible design choices:

- **Approach 1**: One choice is that every MPI process operates one GPU device or a CPU core. In this design, the size of the MPI communicator is equal to the number of available GPU devices in the cluster.

- **Approach 2**: The second choice is a hybrid MPI/OpenMP approach. We could use OpenMP threads within each node to operate the GPU devices or CPU cores and MPI only for communication between nodes. In this case, the size of the MPI communicator is equal to the number of nodes, and the number of OpenMP threads per node is equal to the number of GPU devices per node.

One might argue that approach two has a potential advantage over approach one. In the second approach, communication between GPUs residing on the same cluster node is done implicitly in a shared memory space. Conversely, in the first case, communication is done using explicit MPI calls, which potentially could be more time consuming. However, the MVAPICH implementation of MPI at the Ohio Supercomputer Center will use shared memory to communicate between MPI processes that reside on the same cluster node. Therefore there is no overhead for Approach 1 compared to Approach 2. Moreover, the first approach requires only one API compared to two APIs for the second approach. Hence, in this chapter we present results based on the first approach.
5.3.3 Partitioning and Communication Setup

Our MPI implementation can be summarized in the steps shown in Figure 5.4a-5.4b. First, METIS is used to partition the data into \( M \) partitions. Then each partition \( M_i \) is mapped to an MPI process, with each MPI process handling one GPU device for the MPI/GPU case and one CPU core for the MPI/CPU case, as described in Approach 1 in 5.3.2. After the partitioning is done, we proceed with the setup of the communication data structures, since there is obviously a need to exchange data between neighboring partitions. All communications are done at the CPU level. After the field data are updated in the GPU they are transferred to the CPU through the PCIe2. Data are then communicated, with non-blocking MPI calls, only to MPI processes that need them to perform the next step in the LTS algorithm. They are then transferred back to the GPU to be ready for processing at the next LTS time step. We use two functions that set up the communications. First, the function \( \text{ExchangeGhostInfo()} \) identifies the neighbors for each partition and sets up the buffer sizes according to the shared information between neighboring partitions.
These buffers are indexed by MPI rank and LTS class. Next, `SetupSubdomainLinkage()` fills in those buffers. These buffers will be used later on by two communication functions `mpiShareE()` and `mpishareH()` to share the DOFs of field data between neighbors during the LTS update steps. Also, the matrix data in (2.31)-(2.32) are computed on the CPU during pre-processing and copied to the GPU’s global memory once, before the LTS begins. When all communication data structures are set up, we complete the pre-processing and can proceed to the LTS update. Two communication functions `mpiShareE()` and `mpishareH()` are used at each LTS step to communicate data, if necessary, between neighbors as shown in Figure 5.4b. For all communication, non-blocking MPI calls are utilized using MPI `ISend` and MPI `IRecv` to ensure good performance with respect to communication time. This is critical, since a good imple-

![Figure 5.4: (a) Pre-processing step to set up all necessary buffers for communication between subdomains. (b) Flowchart of how one leap-frog E or leap-frog H update of the LTS algorithm shown in Figure 2 is performed in the MPI/GPU implementation.](image)

mentation of the communication part can lead to better scalability of the MPI code.

Finally, each process will write to disk its own local set of data. When the solution
process is finished, in post-processing one can use `.pvtu` files and `vtkMergeCells` from VTK to merge the partitions back together.

5.4 Numerical Examples

In this section we present our results from the performance analysis study, as well a numerical example that shows the potential and capabilities of the proposed implementation to handle real life examples. For all simulations we used the hardware available at the Ohio Supercomputer Center (OSC). The OSC facilities provide GPU-capable nodes on the Glenn Cluster, connected to Quadro Plex S4’s CUDA-enabled graphics devices. Each Quadro Plex S4 contains 4 Quadro FX 5800 GPUs, with 240 cores per GPU and 4GB of global memory per card. The GPU compute nodes in Glenn also contain dual socket quad core 2.5 GHz AMD Opterons and 24 GB RAM; they communicate through a 20Gb/s Infiniband ConnectX host channel adapter (HCA). For the configuration in the OSC Glenn cluster, each compute node has access to two Quadro FX 5800 graphics cards. All implementations were done on a Redhat Linux OS, in C++. For both the CPU and the GPU versions the MVA-PICH1.1 implementation of MPI at OSC was used, and floating point operations were done in double precision arithmetic. Compilation was done with gcc version 4.1.2, with `-O3` optimization. Additionally, BLAS routines from the ATLAS library were used to perform the matrix-vector multiplications in (2.31)-(2.32) for the MPI/CPU code. For the MPI/GPU code, we used CUDA version 2.3.

5.4.1 Performance Analysis

In this section we analyze the performance of our implementation. As an example we use a coated sphere with inner radius $a = 2 \, m$, outer radius $b = 2.25 \, m$, and
$\epsilon_r = 2.0, \mu_r = 1$ shown in Figure 5.5. The sphere is illuminated with a Gaussian pulse with 3dB bandwidth at $f_{3dB} = 300\,MHz$. The mesh under consideration consists of approximately 1.6M elements, which results in 38M unknowns ($p = 1$ elements were used). For this particular example, the maximum time step given by the stability condition was $\delta t_{max} = 2.99 \times 10^{-11} \, s$, the minimum was $\delta t_{min} = 4.6 \times 10^{-12} \, s$, and application of the LTS algorithm resulted in 2 classes. In this study we concern ourselves with a strong scalability analysis. In a strong scalability analysis the number of unknowns remains the same, and the number of processes (and consequently the number of partitions also) is gradually increased. In order to give a more complete picture, we performed and present our scalability analysis results for both an MPI/CPU and an MPI/GPU implementation.

The scalability results for MPI/CPU are shown in Figure 5.6. The average iteration wall-clock time for one complete LTS update is plotted against the number of MPI processes. In the MPI/CPU case, as shown also in Figure 5.6, each MPI process handles one CPU core. In Figure 5.6 the values for the average iteration time for the ideal curve are calculated as $AverageIterationTime(N) = T_1/N$, where $T_1$ is the average iteration time with one MPI process, and $N$ is the number of MPI processes. One can observe that the ideal values are in good agreement with the actual MPI/CPU average iteration time data. Moreover, for larger numbers of MPI partitions, the MPI average iteration time slowly starts to diverge from the ideal curve. To complete the analysis for the MPI/CPU case, we present the parallel efficiency results shown in Figure 5.6. A satisfactory parallelization efficiency of 80% is achieved for up to 160 CPU cores. Next we consider the results for the MPI/GPU case. Each MPI process controls one GPU device. As mentioned before, each compute node has
access to 2 GPUs, so we have two MPI processes per node. Due to the relatively large number of unknowns and the use of double precision, at least 6 GPUs were necessary to run this example. Therefore in Figure 5.7 the values of the average iteration time for the ideal curve are calculated as $\text{Average Iteration Time}(N) = \frac{T_6}{N}$, where $T_6$ is the average iteration time with 6 MPI processes (6 GPUs) and $N$ is the number of MPI processes. We can observe that the ideal values are in good agreement with the actual MPI/GPU average iteration time data. Again, for larger numbers of MPI partitions the MPI average iteration time slowly starts to diverge from the ideal curve. To complete the analysis for the MPI/GPU case, we present the parallel efficiency results shown in Figure 5.7. A satisfactory parallelization efficiency of 80% is achieved for up to 40 GPU devices. Furthermore, in Figure 5.8 we report the resulting speedup in the solution time. A 10x speedup is achieved. Also note that the speedup values reported in Figure 5.8 and throughout this chapter are for the case of 1 CPU core compared against 1 GPU. In summary, the proposed MPI/GPU implementation maintains a 10x speedup with 40 GPUs, which is a quite satisfactory
Figure 5.6: Scalability and parallelization efficiency of MPI/CPU implementation.

performance result. Additionally, it indicates that the proposed MPI/GPU approach can be used to efficiently solve real-life examples, as will be shown in the following section. Finally, according to [2], new generation GPUs like the Tesla 20 series have a 7x improvement in double precision performance compared to the previous Tesla 10 series. Thus with future hardware there could potentially be further improvements in speedup.

5.4.2 3D SRR Metamaterial Cloaking Device

In this section we present our results from a full wave time domain simulation of a metamaterial cloaking device that was originally designed and presented in [31]. The device consists of 5 cylinders as shown in Figure 5.9. Each cylinder is made of splitting ring resonators (SRRs) whose design parameters are documented in [31]. For all the cylinders the SRRs are printed on a RTDuroid 5780 substrate with relative permittivity $\epsilon_r = 2.33$. The design frequency for this example is 8.5 GHz as discussed in
Figure 5.7: Scalability and parallelization efficiency of MPI/GPU implementation.

This example is characterized by a multi-scale geometry with complex features, which makes it challenging for time domain simulations. The volume concealed by the cloak in this case has free space material properties. The generated mesh consists of 6,685,671 elements resulting in approximately 150 million unknowns ($p = 1$ elements were used). A first order ABC was used for the domain truncation. For this example the minimum time step is $\delta t_{\text{min}} = 1.1166 \times 10^{-14} \text{s}$, the maximum time step is $\delta t_{\text{max}} = 5.3834 \times 10^{-13} \text{s}$, and the application of the LTS algorithm in this example results in 4 classes.

A total of 22 compute nodes, each having 8 CPU cores, 2 Quadro FX 5800 GPUs and 24GB of RAM, were used, all part of the Ohio Supercomputer Center (OSC) Glenn cluster. We ran the same simulation using 2 CPUs per node and also using 2 GPUs per node. All simulations were done in double precision arithmetic. The average LTS time-update iteration time with MPI/GPU was approximately 7.085
Figure 5.8: Speed up in iteration time when comparing one CPU against one CPU core.

Table 5.1: Average iteration time of one complete LTS update (4 classes) for the SRR cloaking example

<table>
<thead>
<tr>
<th>CPU cores per node</th>
<th>GPUs/node</th>
</tr>
</thead>
<tbody>
<tr>
<td># Nodes 22</td>
<td>Iter. Time</td>
</tr>
<tr>
<td>1 core</td>
<td>2 cores</td>
</tr>
<tr>
<td>135.5 s</td>
<td>70.30 s</td>
</tr>
<tr>
<td>100%</td>
<td>96%</td>
</tr>
</tbody>
</table>

seconds. In contrast, the average time-step iteration time with MPI/CPU was approximately 70.30 seconds, resulting in a 10x speedup in the solution time as shown in Table 5.1. The simulation was performed for 6000 LTS updates resulting in 12 hours and 117 hours computational times for MPI/GPU and MPI/CPU respectively. The results of the simulation using the MPI/GPU approach are shown in Figure 5.10. The cloaking device was illuminated with a Neuman pulse with 3dB bandwidth 8-11
GHz (Xband); the polarization of the field is along the cylinder axis. In Figure 5.10 we show snapshots of the Neuman pulse as it propagates through the cloaking device. The following comments can be made about the performance of the cloaking device. First, there is a small but noticeable scattered field in the forward region induced by the cloaking, as seen in Figure 5.10 (b)-(f). Moreover, in Figure 5.10 (f) one can clearly observe ripples when the pulse exits the cloaking device. These ripples are caused by time delay due to the long traveling paths through the SRRs at each cylinder, which distorts the pulse shape. This time delay is also justified by the fact that SRRs are quite narrow-band and metamaterials built with SRRs are dispersive. Therefore, this distortion in the pulse shape can be detected and indicates the presence of the cloaking as a scatterer, hence reducing the effectiveness of the cloaking. Thus the cloak plus the concealed volume appear to both scatter waves and distort the transmitted field. Consequently one could come to the conclusion that a true cloaking seems not to be achieved in time domain when the structure is illuminated by a pulse with spectral content in the Xband 8-11 GHz, which is the frequency range that modern radars use.

5.5 Discussion

In this chapter, we have presented our approach on an MPI/GPU and MPI/CPU implementation of the IPDGTD method for the time domain Maxwell’s equations. We used the CUDA programming model for the GPU side of the implementation. Moreover, we combined the local time stepping (LTS) algorithm with our MPI design, to efficiently address the multi-scale nature of most complex applications. By doing
a performance analysis we found that MPI/GPU can provide a 10x speed up compared to its MPI/CPU counterpart, for double precision arithmetic. Furthermore, we demonstrated that the proposed approach provides satisfactory strong scalability for both MPI/CPU and MPI/GPU implementations. An 80% parallelization efficiency was achieved up to 160 CPU cores and an 85% parallelization efficiency was achieved.
Figure 5.10: A pulse with 8GHz-11GHz 3dB bandwidth propagating through the cloaking device. Small but noticeable scattered field in the forward region and ripples, due to time delay, as the pulse exits the cloak, reduce the cloaking effectiveness.

up 40 GPUs, at the Ohio Supercomputer Center, Glenn cluster. Additionally, a full wave simulation of a 3D SRR cloaking device was performed in time domain, to show the ability of the proposed MPI/GPU and MPI/CPU to handle complicated and challenging EM applications.
Chapter 6: Non-conformal Interior Penalty Discontinuous Galerkin Time Domain Method

In this chapter, we extend the work presented in the previous chapters by enriching our IPDGT method with the support for non-conformal meshes. The motivation for a non-conformal IPDGT method, comes from the fact that in cases of very complicated geometries a conformal mesh may be very difficult to obtain. Therefore, the ability to handle non-conformal meshes really comes into picture. In our methodology, at the first step we decompose the computational domain into non-overlapping subdomains. Next, each sub-domain is meshed independently resulting in non-conformal domain interfaces but simultaneously providing great flexibility in the meshing process. The non-conformal triangulations at sub-domain interfaces can be naturally supported within the IPDGT framework. Moreover, a MPI parallelization together with a local time-stepping strategy is applied to significantly increase the efficiency of the method. Additionally, a new balancing is introduced to provide for efficient use of the available computational resources. Finally, some numerical examples demonstrate the capabilities of the proposed approach.
6.1 DGTD Formulation-Support for non-conformal meshes

Up to this point we have presented an IPDGTD method applied on conformal meshes. The meshes were considered to be conformal in the sense that the intersection between two adjacent elements $K_i$ and $K_j$ is a complete face as shown in Figure 6.1b. In the following we discuss the application of our IPDGTD in the case of non-conformal meshes as shown in Figure 6.1c. In our approach, we start by decomposing the computational domain into non-overlapping subdomains. In this work, the interfaces between subdomains are allowed to be geometrically non-conformal but they also have to be planar as shown in Figure 6.1a. Non planar interfaces are left as future work. Each subdomain is meshed independently resulting in non-conformal triangulations at domain interfaces. In this way, the mesh generation becomes less intensive not only because the conformity requirements at interfaces are dropped, but also because meshing can be done in a parallel fashion. As already mentioned in Chapter 2, in DG methods the tangential continuity of the electric and magnetic fields between elements, is enforced in the weak sense by means of numerical fluxes. These numerical fluxes are enforced at the intersection of the element boundaries ($\partial K_i \cap \partial K_j$). Because of this weak treatment of the tangential continuity, it is apparent that DG can naturally support non-conformal meshes. Thus, for non-conformal meshes, the IPDGTD formulation presented in Chapter 2 section 2.2 remains identical and is still completely valid. The main difference is in the computation of the coupling matrices which we describe shortly below. Hence, we will not repeat it in this section and we will start our discussion from the fully discretized local system of equations (2.31)-(2.32) which are repeated below in (6.1)-(6.2) for convenience.
\[ M_e e_i^{n+1} = (M_e + e\delta t P_{e e}^{n}) e_i^{n} + \delta t (S_e - F_{e e}^{n}) h_i^{n+\frac{1}{2}} - \delta t F_{e e}^{ij} h_j^{n+\frac{1}{2}} + e\delta t P_{e e}^{ij} e_j^{n} \]

(6.1)

Local terms

Coupling terms

\[ M_\mu h_i^{n+\frac{3}{2}} = (M_\mu + f\delta t P_{h h}^{n}) h_i^{n+\frac{3}{2}} + \delta t (-S_h + F_{h h}^{n}) e_i^{n+1} + \delta t F_{h h}^{ij} e_j^{n+1} + f\delta t P_{h h}^{ij} h_j^{n+\frac{3}{2}} \]

(6.2)

Local terms

Coupling terms

When non-conformal meshes are concerned, the elements that need to be updated according to (6.1)-(6.2) can be split into the following two cases:

- An element that couples conformally to all its neighbors. In this case each tetrahedron will have at most four neighbors, and the coupling matrices are calculated in a straightforward manner. In our case a universal matrix approach [34] is used to increase efficiency.

- An element that couples non-conformally through one or more faces to some of its neighbors and conformally to the rest as shown in Figure 6.2a for Tet2. In this case, the coupling matrices \( F_{e(h)}^{ij} \) and \( P_{e(h)}^{ij} \) are calculated on the intersection \( face_i \cap face_j \) of the two overlapping faces \( face_i \) and \( face_j \). Hence, \( (F_{e(h)}^{ij})_{nm} = \frac{1}{2} \int_{face_i \cap face_j} \pi_T(v_{in}) \cdot \gamma_T(w_{jm}) \, ds \), \( (P_{e(h)}^{ij})_{nm} = \int_{face_i \cap face_j} \pi_T(v_{in}) \cdot \pi_T(v_{jm}) \, ds \).

A simple illustration example is shown in Figure 6.2a. Let us focus on Tet2, since a similar logic applies to the others tetrahedrons as well. Tet2 will couple non-conformally to Tet1, Tet4 and conformally to Tet3, Tet5 as shown in Figure 6.2a. The matrices \( P^{23}(F^{25}) \), \( P^{23}(P^{25}) \) are calculated straightforwardly. The matrices \( F^{21}(F^{24}) \), \( P^{21}(P^{24}) \) for the non-conformal coupling to Tet1 and Tet4 are computed as described in the following section.
Figure 6.1: (a) Generic illustration of the domain partitioning into non-overlapping subdomains. The subdomains must have planar interfaces but they can also be geometrically non-conformal. (b) The dashed red line shows the intersection where the coupling is enforced. Example of conformal case coupling. (c) The dashed red line shows the intersection where the coupling is enforced. Example of non-conformal case coupling.

Non-conformal Interface Matrix Calculation

The following steps are applied to accurately calculate the coupling matrices for the non-conformal case. A pictorial illustration is also given in Figure 6.2b.
• First the intersection between the two non-conformal overlapping faces $face_i$ and $face_j$ is computed. Since, in this work the interfaces between sub-domains are planar the intersection is in general a planar polygon. The GPC library [1] is used to compute the intersection polygon.

• Next, the intersection polygon is triangulated into $N$ triangles.

• For each triangle $t_i$, Gauss Quadrature is used to perform the integration. Finally, all the triangle integrations are summed to get the final result of the the integration over the polygonal intersection. ($\int_{face_i \cap face_j} dx^2 = \sum_{i=1}^{N} \int_{t_i} dx^2$).

Figure 6.2: (a) Example of non-conformal planar interface. (b) Calculation of interface matrix on the non-conformal interface.
Stability Analysis and Local Time-Stepping (LTS)

The stability analysis and local time-stepping for the non-conformal case follow the approaches described in earlier chapters. In section 2.2.4 of Chapter 2, we have already discussed that the stability condition presented there can be used for both conformal and non-conformal meshes. Therefore, for more details the reader is pointed to 2.2.4 of Chapter 2 and/or Appendix A. Moreover, the local time-stepping strategy is same for both conformal and non-conformal cases and has been described in section 2.2.4 of Chapter 2.

6.2 MPI Parallelization

In this section, we will discuss some of the issues that we faced in the parallelization of the non-conformal IPDGT and how we address them. Initially, let us point out, that the MPI/LTS design of Chapter 5 is the same for both conformal and non-conformal cases and therefore will not be repeated here. The major difference, is that in the conformal case, METIS [20] was used to partition the conformal mesh into subdomains. On the other hand, in the non-conformal case the subdomains are created by geometrically partitioning the computational domain into subdomains and then meshing each subdomain independently. This difference in the way of partitioning, as it will be explained in the next paragraph, will create some significant balancing issues for the non-conformal case. Conversely, these balancing issues, were not significant in the conformal case because of METIS [20] has the ability to create relatively well balanced partitions. In the following we describe our balancing methodology.
6.2.1 Balancing Strategy

As already mentioned in previous sections, the first step in our IPDGTD strategy for non-conformal meshes is to decompose the computational domain into non-overlapping subdomains. Moreover, one should keep in mind, that in our scenario this initial decomposition is mainly aimed to relax the mesh generation process for geometrically complex structures. Therefore, the subdomains are formed in such a way that each one of them would be as easy as possible to mesh. However, this decomposition does do not guarantee that the subdomains will be also well balanced when distributed for computation to a cluster. In fact we have observed that the opposite can be true. Therefore, a balancing strategy is needed to efficiently distribute the problem to a cluster. To start our discussion, assume we are given an initial decomposition of a domain into $M_{unbal}$ subdomains that are unbalanced. Our goal is to generate a new decomposition of the same original domain into $M_{bal}$ subdomains that are better balanced. Let us denote by $SD_{i}^{unbal}$, an unbalanced domain and by $SD_{i}^{bal}$, a balanced subdomain. The general principles of our methodology are described at Algorithm 1. A depiction of the three steps of Algorithm 1 for a simple parallel plate

---

**Algorithm 1**: Balancing Strategy

**Step1**: Compute the amount of floating point operations (FLO) $FLO_{SD_{i}}$ that each $SD_{i}^{unbal}$ will need in order to perform a complete LTS update step. Then, find the minimum of those $FLO_{SD_{i}}$ and define what we choose to call a work unit (WU) as $WU = \alpha \min(FLO_{SD_{i}})$, $0 < \alpha < 1$.

**Step2**: Split each $SD_{i}^{unbal}$ into $n_{i}^{unbal}$ smaller partition using METIS [20]. For each $SD_{i}^{unbal}$, the number $n_{i}^{unbal}$ is computed by $n_{i}^{unbal} = \lceil FLO_{SD_{i}} / WU \rceil$.

**Step3**: Regroup the available work units into $M_{bal}$ balanced subdomains $SD_{i}^{bal}$ and distribute them for computation.
waveguide (PPWG) example is given in Figure 6.3. The original five subdomains that are found to be unbalanced are regrouped into five new balanced subdomains, as it will be shown later on in Figure 6.4. In the following we give more details about the three steps of Algorithm 1. For simplicity, but without loss of generality we will continue to use the parallel plate waveguide (PPWG) example to navigate throughout our discussion. The application of the balancing strategy to more complex examples will be shown in section 6.3.2.

**Analysis-Partitioning into Work Units (WUs)**

In this section, we provide more details about Steps 1 and 2 of Algorithm 1. The detailed description for Step 1 is given in Algorithm 2. Once Step 1 is completed, then Step 2 of Algorithm 1 is straightforward application of the METIS [20] library. At this point, let us also clarify that Algorithm 1, and subsequently Algorithm 2 are going to be executed only once during the pre-processing stage. Continuing our discussion, we apply Algorithm 2 to the PPWG illustration example. This examples
Algorithm 2 : Step 1

(a) : Run the unbalanced case for only one complete LTS update step to generate the distribution of the elements into the LTS classes.

(b) : Compute the amount of floating point operations (FLO) $FLO_{SD_i}$ that each subdomain/MPI-process will need to perform in order to complete one LTS update step. The $FLO_{SD_i}$ for each subdomain are computed by the following procedure:

1. Let NeigNum be the numbers of neighbors for each element $K_i$ of the subdomain. If we break down (6.1)-(6.2), we need $F_{Mx} = 2 \times \gamma \times NeigNum$ ($\gamma = 2$ for upwind and $\gamma = 1$ for central) surface matrix-vector multiplications for the coupling terms. Furthermore, we need 4 volume matrix-vector multiplications to compute the local terms and 2 volume matrix-vector multiplication for the inverse mass matrix multiplications. Hence, in total $V_{Mx} = 6$ volume matrix-vector multiplications.

2. Define by $d_{vol_i}$ the volume DOFs of element $K_i$. Moreover, let $d_{face_i,j}$ and $d_{face_j}$ ($j \in Neigh(i)$) be the DOFs from the $i$ and $j$ side respectively of face$_{ij}$ of element $K_i$. Then, the FLO for a volume matrix-vector multiply are $FLO_{volume} = 2d_{vol_i}(d_{vol_i} - 1)$. Furthermore, the FLO for a face matrix vector-multiply are $FLO_{face} = 2d_{face_i}(d_{face_j} - 1)$. Hence, we get $FLO_{elem} = V_{Mx} \times FLO_{volume} + F_{Mx} \times FLO_{face}$.

3. Moreover, due to the LTS strategy, for each element $K_i$ (6.1)-(6.2) will be evaluated $3^{N_{class} - 1 - k}$, $k = 0, 1, ..., N_{class} - 1$. Here, $N_{class}$ is the number of LTS classes, and $k$ is the index of the class that element $K_i$ belongs to. Hence, we obtain the the final FLO for each element $K_i$ as $FLO_{elem} \leftarrow FLO_{elem} \times 3^{N_{class} - 1 - k}$.

4. Finally, we obtain the FLO for the subdomain by adding all the element FLO to get $FLO_{SD_i} = \sum_{elem} FLO_{elem}$.

(c) : Set $WU = \alpha \min(FLO_{SD_i})$, $0 < \alpha < 1$. 
has 4 LTS classes with $\delta t_{\text{min}} = 8.63 \cdot 10^{-13}$ s and $\delta t_{\text{max}} = 2.33 \cdot 10^{-11}$ s. The distribution of the $FLO_{SD_i}$ shown in Figure 6.4. The work unit for this particular example was set to be $WU = 1.16e07$ FLO. With this value for the work unit the distribution of the splits $n_{i}^{unbal}$ was found to be \{10, 10, 3, 2, 20\}. As expected, the number of splits $n_{i}^{unbal}$ is analogous to $FLO_{SD_i}$. Furthermore, Figure 6.4 clearly indicates that the original decomposition is unbalanced in terms of FLO. As a consequence, the MPI-processes with high values of FLO will take longer and dominate the LTS update time while all other MPI-processes are sitting idle. The following subsection discusses how we can regroup the work units into a more balanced decomposition.

**Synthesis-Regrouping**

Continuing on the parallel plate waveguide (PPWG) example, we will describe the steps of our regrouping strategy which is Step 3 of Algorithm 1. The basic idea is that we can create a graph $G(v,e)$ to describe the WUs and their connectivity and use METIS [20] to split the graph into $M_{bal}$ balanced partitions. The steps of the regrouping strategy are shown in Algorithm 3. After the application of Algorithm 3 to our PPWG example we will obtain the FLO distribution shown in Figure 6.4. One can clearly observe that the new regrouped subdomains of Figure 6.3 are much better balanced compared to the original subdomains.

**6.3 Numerical examples**

In this section, we present some numerical examples to validate the proposed methodology and show its potential for the simulation of engineering applications.
Algorithm 3: Step 3

(a) : After the split into \(WU\) is finished, map a \(WU\) to a vertex in a graph \(G(v, e)\). Moreover, represent the connection between neighboring \(WU\) with an edge in the graph.

(b) : Call METIS \[20\] again to split the graph \(G(v, e)\) into \(M_{bal}\) balanced partitions.

(c) : Assign all of the \(WU\) of partition \(M_i\) to a new subdomain \(SD_{i}^{bal}\). Next, map a MPI process to handle one of the \(SD_{i}^{bal}\) following the MPI/LTS design of Chapter 5.

(d) : Calculate the FLO for the new \(SD_{i}^{bal}\) subdomain as described at Algorithm 2 and verify that the new \(SD_{i}^{bal}\) are better balanced.

---

Figure 6.4: (a) FLO distribution for original subdomains. One can clearly notice the original subdomains are not well balanced for computation. (b) FLO distribution for new subdomains. The new subdomains are better balanced for computation.

6.3.1 Simple Via Example

In this example, a simple microstrip line coupling through a via between two layers is studied. The complete geometry of the problem is shown in Figure 6.5. The model is decomposed into 8 subdomains and each subdomain is meshed independently. The generated meshes have a total of 96,031 elements and the polynomial order was set to
$p = 2$. The minimum and maximum time step were found to be $\delta t_{\text{min}} = 1.13 \cdot 10^{-15}$ s and $\delta t_{\text{max}} = 1.85 \cdot 10^{-13}$ s, resulting in 5 classes in the LTS algorithm. Following our MPI/LTS design the problem was distributed to 8 compute nodes with 4 cores and 8GB per node at the Ohio SuperComputer Center (OSC) Glenn cluster. The input port is excited with a Gaussian pulse with a $3dB$ frequency $f_{3dB} = 10$ GHz. A first-order Silver-Muller ABC was used for the domain truncation. The voltages are recorded at the input and output ports and the $S-$parameters are computed in the frequency domain by means of Fourier transform $S_{ij} = F\left\{ \frac{V_{\text{ref}}(t)}{V_{\text{inc}}(t)} \right\}$. Then, the $S-$parameters are compared with the ones from the commercial software HFSS as shown in Figure 6.5. A good comparison is observed when compared with the HFSS result validating our approach.

**6.3.2 IBM (IC) Model**

**Problem Description**

In this section, we show the application of the developed IPDGTID to the simulation of an IC package provided by IBM [22]. The structure shown in Figure 6.6, consists of eight metalization layers embedded into a dielectric substrate or $\epsilon_r = 4.4$. These eight-layers are: ground/mounting pads (SURFACE), signal (FC3), ground (FC2), signal (FC1), signal (BC1), power (BC2), signal (BC3), and ground/mounting pads (BASE). A schematic view of the stack-up is shown in Figure 6.6. In our example, we will focus our attention to the analysis of the part of the IC package that is enclosed in the red box in Figure 6.6 [32]. The dimensions of this reduced model are 10.5 mm $\times$ 16.5 mm. More geometrical details about each layer are given in Figure 6.7. The majority of the signal traces propagate on the layer FC3. The signal is
Figure 6.5: (a) Geometry of the problem. The computational domain is decomposed into 8 subdomains. (b) Comparison of magnitude of $S_{11}$ parameter. (c) Comparison of phase of $S_{11}$ parameter. (d) Comparison of magnitude of $S_{12}$ parameter. (e) Comparison of phase of $S_{12}$ parameter.
connecting to the trace lines of FC3 through vias, then propagates through layer FC3 and finally routed by vias directly to the BASE side of the structure.

As it can be seen from Figure 6.7 the geometrical description of the model is of very high complexity. Due to this high complexity, we were unable to obtain a conformal mesh for the whole structure. Consequently, a divide and conquer non-conformal approach is necessary for this example in order to be able to obtain a mesh and perform a simulation. As discussed previously, the first step consists of partitioning the original domain into non-overlapping subdomains. In our case, we reuse the partition that was originally presented in [32] and the author would like to thank the authors of [32] for providing the partition. The whole domain is decomposed in totally 149 non-overlapping subdomains with planar interfaces. The subdomains are shown layer by layer on Figure 6.7 and are allowed to be geometrically non-conformal. Next, each subdomain is meshed completely independent providing the final mesh to be used for computation.

Figure 6.6: (a) IC package provided by IBM. Here, we focus our attention on a signal integrity analysis for the part enclosed in the red box (courtesy of [32]). (b) Schematic illustration of the eight layers in IBM model structure.
Figure 6.7: (a)-(h) Top view of the metalization of the 8 layers that the IBM model consists of. (ground/mounting pads (SURFACE), signal (FC3), ground (FC2), signal (FC1), signal (BC1), power (BC2), signal (BC3), and ground/mounting pads (BASE)). Each layer is decomposed into a number of subdomains resulting in a total of 149 subdomains.

6.3.3 Numerical Results

Here, we present our numerical results for the model of the IBM IC package. One of the trace lines is excited and then the voltage at the output will be monitored. More specifically, we are going to excite trace line DATA06 (input) and compute the voltage at the end of the line DATA06 (output). To give a better description of the problem, the physical location of the input and output ports are shown in Figure 6.9.
At the input, the signal is a step source with rise time of 80\,ps like the one shown in Figure 6.9. This input signal, is the same as the one used in the measurement data so a direct comparison can be made between measured and simulated data. For this problem, the final mesh consists of approximately $4.9 \times 10^6$ tetrahedral elements and $167.5 \times 10^6$ DOFs. For the domains including the trace lines $p = 2$ basis where used and $p = 1$ basis where used elsewhere. The minimum and maximum time step were calculated to be $\delta t_{\text{min}} = 5.31 \times 10^{-17}\,s$ and $\delta t_{\text{max}} = 1.32 \times 10^{-13}\,s$, resulting in 8 classes in the LTS algorithm. The distribution of the elements to the LTS classes is given in Table 6.2.

**Application of the Balancing Strategy**

In this section, we discuss the application of the balancing strategy of section 6.2.1 to the IBM model example of Figure 6.6 - 6.7. To evaluate the performance of our balancing strategy we have performed 3 simulations. In the first simulation the original/unbalanced 149 subdomains ($SD_{i}^{\text{unbal}}$) of the problem are distributed to 50 nodes with 8 cores and 24GB per node at the Ohio SuperComputer Center (OSC) Glenn cluster. In this case 2 CPU cores were used per subdomain. In the second simulation, we apply our balancing strategy and the new balanced subdomains ($SD_{i}^{\text{bal}}$) are distributed to 50 nodes with 8 cores and 24GB per node on the Glenn cluster of OSC. Again 2 CPU cores were used per subdomain so a direct comparison can be made with the unbalanced case of simulation number one. Finally, in the third simulation the same new balanced subdomains are distributed to 38 nodes with 12 cores and 48GB per node on the Oakley cluster of OSC. For the Oakley cluster simulation 3 CPU cores were used per subdomain.
To indicate how unbalanced a distribution of subdomains is, we compute the ratio of the maximum FLO over the average FLO defined as $R_{unbal} = \frac{FLO_{max}}{FLO_{avg}}$. Smaller values of $R_{unbal}$ indicate better balancing. The distribution of the $FLO_{SD_i}$ for the first simulation is shown in Figure 6.8a. One can clearly observe that the original subdomains are highly unbalanced for computation, which is also indicated by computing $R_{unbal} = 9.0$. Likewise, the distribution of the $FLO_{SD_i}$ for the second and third simulation is shown in Figure 6.8b. From the graph, it is clear that the new subdomains are better balanced. The value of $R_{bal}$ in this case was found to be $R_{unbal} = 2.0$ indicating a much better balancing amongst the subdomains/MPI-processes. Finally, the benefits of this improved balancing are clearly shown in Table 6.1 where we compare the average iteration time before and after the balancing. As it can be seen the balancing strategy offers an additional 2.2x gain in the solution time for the Glenn cluster set up, whereas a 5.0x speed up is obtain for the Oakley cluster set up.

![Graph](image-url)

Figure 6.8: (a) Distribution of the $FLO_{SD_i}$ for the original/unbalanced 149 subdomains. (b) Distribution of the $FLO_{SD_i}$ for the new balanced 149 subdomains.
### Table 6.1: Computational statistics for the IBM model.

<table>
<thead>
<tr>
<th>Method</th>
<th>Nodes</th>
<th>Iter. Time</th>
<th>Cluster</th>
<th>Balancing Time</th>
<th>LTS Time</th>
<th>CPU Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Bal.</td>
<td>50</td>
<td>360 (sec)</td>
<td>Glenn</td>
<td></td>
<td>16 days</td>
<td>1.0</td>
</tr>
<tr>
<td>Balan.</td>
<td>50</td>
<td>150 (sec)</td>
<td>Glenn</td>
<td>19 hours</td>
<td>6.8 days</td>
<td>2.2</td>
</tr>
<tr>
<td>Balan.</td>
<td>38</td>
<td>47 (sec)</td>
<td>Oakley</td>
<td>16 hours</td>
<td>2.5 days</td>
<td>5.0</td>
</tr>
</tbody>
</table>

### Table 6.2: Distribution of elements into the LTS classes for the IBM model.

<table>
<thead>
<tr>
<th>#Class</th>
<th>#Elements</th>
<th>$\delta t$ (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>164</td>
<td>5.31e-17</td>
</tr>
<tr>
<td>1</td>
<td>1,016</td>
<td>1.59e-16</td>
</tr>
<tr>
<td>2</td>
<td>3,167</td>
<td>4.78e-16</td>
</tr>
<tr>
<td>3</td>
<td>112,654</td>
<td>1.43e-15</td>
</tr>
<tr>
<td>4</td>
<td>3,587,431</td>
<td>4.30e-15</td>
</tr>
<tr>
<td>5</td>
<td>1,132,131</td>
<td>1.29e-14</td>
</tr>
<tr>
<td>6</td>
<td>78,317</td>
<td>3.87e-14</td>
</tr>
<tr>
<td>7</td>
<td>40</td>
<td>1.16e-13</td>
</tr>
</tbody>
</table>

### Computation and Comparison of Time-domain Voltages

The voltages at the input and the output are obtained by performing the line integral of the electric field. The computed voltage at the input and at the output of trace line DATA06 are shown in Figure 6.9. A relatively good agreement is observed with the measurement data. A small error is observed in the delay time. The measured delay time was $119.3 \text{ ps}$ whereas the computed delay time was $134.3 \text{ ps}$. Moreover, the difference in magnitude between measured and computed voltage is explained by the fact that both conductor and dielectric losses are not considered in the simulation but they are present in the measurement. Also, as expected, the results are identical between unbalanced and balanced cases. In conclusion, the obtained results show
Figure 6.9: (a) Simulation set up. The input of line DAT06 is excited with a step source with 80 ps rise time and the voltage at the output of DATA06 is computed. (b) Snapshot of the distribution of electric field magnitude at domain 41 where the input port is located. (c) Comparison between computed and measured voltages. A fairly good agreement with the measurement data is observed. (d) Snapshot of the distribution of electric field magnitude at the signal layer FC3.
the potential of the non-conformal IPDGTD to simulated practical and complicated applications. Finally, the total solution time for the simulation time of 0.45\textit{ps} of Figure 6.9 was approximately 16 days on the Glenn cluster for the unbalanced case and for the balanced case (including balancing time): (a) 7.5 days on the Glenn cluster; (b) 3.15 days on the Oakley cluster. In the future a MPI/GPU version of the same approach would potentially further decrease the solution time.

6.4 Discussion

In this chapter, we presented how we can enrich the capabilities of our IPDGTD method by adding the ability to handle non-conformal discretizations. The support for non-conformal meshes can be practically useful since there exist applications where a conformal mesh generation for the whole structure can be very challenging. Moreover, in some applications each component may be designed individually before mounted to the whole structure. In such a scenario non-conformal meshes are again useful since we only need to re-mesh the component rather than the whole structure. In our approach, a divide and conquer strategy is applied. Initially, the original domain is decomposed into subdomains. Next, each subdomain is mesh independently offering great flexibility. Moreover, to further increase efficient we have successfully parallelized the proposed IPDGTD. Our parallelization is a combined MPI and local time stepping strategy whose details are found in Chapter 5. Furthermore, to further address the unbalanced nature of the original geometrical cut a general balancing strategy has been proposed and implemented. Finally, the presented numerical examples validate our approach and show the potential of the proposed methodology in the simulation and analysis of interesting engineering applications.
Appendix A: Appendix 1

A.1 Stability Analysis by an Energy Technique

In this section, we attempt to perform a study on the stability of the leap-frog discretized system (2.31)-(2.32), of our IPDGD without taking into account local time-stepping (LTS) strategy. The stability with LTS is a more complicated task and will be studied in the future. Our study is based on an energy technique similar to the ones in [9, 10, 11, 26], to find a stability condition for both upwind and central flux. Herein, $K_i$ is an element of $\mathcal{T}_h$ and $a_{ij} = \partial K_i \cap \partial K_j \in \mathcal{F}_h^T, j \in \text{Neigh}(i)$, is the intersection area of the overlapping faces between elements $K_i$ and $K_j$ of $\mathcal{T}_h$.

A.1.1 Energy Conservation

We use an approach similar to [11, 26], to study the energy conservation properties of the discretized system (2.31)-(2.32) for both central and upwind flux. From (2.31)-(2.32) we have:

$$
\int_{K_i} \mathbf{v}_i \cdot \epsilon_i \cdot \frac{\mathbf{E}_i^{n+1} - \mathbf{E}_i^n}{\delta t} = \int_{K_i} \mathbf{v}_i \cdot (\nabla \times \mathbf{H}_i^{n+1/2})
- \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} \mathbf{v}_i \cdot (\mathbf{n}_i \times \mathbf{H}_i^{n+1/2} + \mathbf{\hat{n}}_j \times \mathbf{H}_j^{n+1/2}) + \sum_{j \in \text{Neigh}(i)} e_{ij} \int_{a_{ij}} \mathbf{v}_i \cdot (\mathbf{\hat{n}}_i \times \mathbf{\hat{n}}_i \times (\mathbf{E}_i^n - \mathbf{E}_j^n))
$$

(A.1)
\[ \int_{K_i} w_i \mu_i \cdot \left( \frac{H_i^{n+3/2} - H_i^{n+1/2}}{\delta t} \right) = -\int_{K_i} w_i \cdot (\nabla \times E_i^{n+1}) \]
\[ + \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} w_i \cdot (n_i \times E_i^{n+1} + \hat{n}_j \times E_j^{n+1}) \]
\[ + \sum_{j \in \text{Neigh}(i)} f_{ij} \int_{a_{ij}} w_i \cdot (\hat{n}_i \times \hat{n}_i \times (H_j^{n+1/2} - H_j^{n+1/2})) \]

where \( f_{ij}(j) = \frac{\kappa}{1/2(\sqrt{\epsilon_i/\mu_i} + \sqrt{\epsilon_j/\mu_j})} \) and \( e_{ij}(j) = \frac{\kappa}{1/2(\sqrt{\mu_i/\epsilon_i} + \sqrt{\mu_j/\epsilon_j})} \). Also, \( \kappa = 0 \) for central flux and \( \kappa = 1/2 \) for the upwind flux. Let \( v_i(j) \rightarrow E_i(j) \) and \( w_i(j) \rightarrow H_i(j) \).

Define the discrete analog of the local and global energy as,

\[ E_i^n = \frac{1}{2} \epsilon_i \int_{K_i} E_i^n \cdot E_i^n + \frac{1}{2} \mu_i \int_{K_i} H_i^{n-1/2} \cdot H_i^{n+1/2}, \quad E_{\text{global}} = \sum_i E_i^n \]

respectively. We have that:

\[ 2(E_i^{n+1} - E_i^n) = \int_{K_i} (E_i^{n+1} + E_i^n) \epsilon_i \cdot (E_i^{n+1} - E_i^n) + \int_{K_i} H_i^{n+1/2} \mu_i \cdot (H_i^{n+1/2} - H_i^{n-1/2}) \]
\[ = \delta t \int_{K_i} (E_i^{n+1} + E_i^n) \cdot \nabla \times H_i^{n+1/2} \]
\[ - \frac{\delta t}{2} \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} (E_i^{n+1} + E_i^n) \cdot (n_i \times H_i^{n+1/2} + \hat{n}_j \times H_j^{n+1/2}) \]
\[ + \sum_{j \in \text{Neigh}(i)} e_{ij} \delta t \int_{a_{ij}} (E_i^{n+1} + E_i^n) \cdot (\hat{n}_i \times \hat{n}_i \times (E_i^n - E_j^n)) \]
\[ - \delta t \int_{K_i} H_i^{n+1/2} \cdot \nabla \times (E_i^{n+1} + E_i^n) \]
\[ + \frac{\delta t}{2} \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} H_i^{n+1/2} \cdot (n_i \times (E_i^{n+1} + E_i^n) + \hat{n}_j \times (E_j^{n+1} + E_j^n)) \]
\[ + \sum_{j \in \text{Neigh}(i)} f_{ij} \delta t \int_{a_{ij}} H_i^{n+1/2} \cdot (\hat{n}_i \times \hat{n}_i \times (H_i^{n+1/2} + H_i^{n-1/2})) \]
\[ + \sum_{j \in \text{Neigh}(i)} f_{ij} \delta t \int_{a_{ij}} H_i^{n+1/2} \cdot (\hat{n}_i \times \hat{n}_i \times \left( - (H_j^{n+1/2} + H_j^{n-1/2}) \right)) \]

(A.3)

Using the vector identity,

\[ (E_i^{n+1} + E_i^n) \cdot (\nabla \times H_i^{n+1/2}) = -\nabla \cdot ((E_i^{n+1} + E_i^n) \times H_i^{n+1/2}) + H_i^{n+1/2} \cdot \nabla \times (E_i^{n+1} + E_i^n) \]
and the divergence theorem one can arrive at:

\[
2(\mathcal{E}_i^{n+1} - \mathcal{E}_i^n) = \frac{-\delta t}{2} \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} (E_i^{n+1} + E_i^n) \cdot (\hat{n}_j \times H_j^{n+1/2})
\]

\[
+ \sum_{j \in \text{Neigh}(i)} e_{ij} \delta t \int_{a_{ij}} (E_i^{n+1} + E_i^n) \cdot (\hat{n}_i \times \hat{n}_i \times (E_i^n - E_j^n))
\]

\[
+ \frac{\delta t}{2} \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} H_i^{n+1/2} \cdot (n_j \times (E_j^{n+1} + E_j^n))
\]

\[
+ \sum_{j \in \text{Neigh}(i)} f_{ij} \delta t \int_{a_{ij}} H_i^{n+1/2} \cdot \left(\hat{n}_i \times \hat{n}_i \times \left(H_i^{n+1/2} + H_i^{-n-1/2}\right)\right)
\]

\[
+ \sum_{j \in \text{Neigh}(i)} f_{ij} \delta t \int_{a_{ij}} H_i^{n+1/2} \cdot \left(\hat{n}_i \times \hat{n}_i \times \left(-(H_i^{n+1/2} + H_i^{-n-1/2})\right)\right)
\]

\[
(A.5)
\]

Then, summing up for all elements, we get

\[
2 \sum_i \mathcal{E}_i^{n+1} - \mathcal{E}_i^n = \sum_{a_{ij}} -\frac{\delta t}{2} \int_{a_{ij}} (E_i^{n+1} + E_i^n) \cdot (\hat{n}_j \times H_j^{n+1/2})
\]

\[
+ \sum_{a_{ij}} \frac{\delta t}{2} \int_{a_{ij}} H_j^{n+1/2} \cdot (\hat{n}_i \times (E_i^{n+1} + E_i^n))
\]

\[
+ \sum_{a_{ij}} -\frac{\delta t}{2} \int_{a_{ij}} (E_j^{n+1} + E_j^n) \cdot (\hat{n}_i \times H_i^{n+1/2})
\]

\[
+ \sum_{a_{ij}} \frac{\delta t}{2} \int_{a_{ij}} H_i^{n+1/2} \cdot (\hat{n}_j \times (E_j^{n+1} + E_j^n))
\]

\[
+ \sum_{a_{ij}} e_{ij} \delta t \int_{a_{ij}} (E_i^{n+1} + E_i^n) \cdot (\hat{n}_i \times \hat{n}_i \times (E_i^n - E_j^n))
\]

\[
+ \sum_{a_{ij}} f_{ij} \delta t \int_{a_{ij}} (H_i^{n+1/2} \cdot (\hat{n}_i \times \hat{n}_i \times \left(H_i^{n+1/2} + H_i^{-n-1/2}\right))
\]

\[
+ \sum_{a_{ij}} f_{ij} \delta t \int_{a_{ij}} H_i^{n+1/2} \cdot \left(\hat{n}_i \times \hat{n}_i \times \left(-(H_i^{n+1/2} + H_i^{-n-1/2})\right)\right)
\]

\[
+ \sum_{a_{ij}} f_{ij} \delta t \int_{a_{ij}} (H_j^{n+1/2} \cdot (\hat{n}_j \times \hat{n}_i \times \left(H_j^{n+1/2} + H_j^{-n-1/2}\right))
\]

\[
+ \sum_{a_{ij}} f_{ij} \delta t \int_{a_{ij}} H_j^{n+1/2} \cdot \left(\hat{n}_j \times \hat{n}_j \times \left(-(H_j^{n+1/2} + H_j^{-n-1/2})\right)\right)
\]

104
The first four terms will cancel each other. Hence, after cancellations and groupings we have:

\[ 2 \sum_i E_i^{n+1} - E_i^n = \sum_{a_{ij}} e_{ij} \delta t \int_{a_{ij}} (E_i^{n+1} - E_j^{n+1}) \cdot (\hat{n}_i \times \hat{n}_i \times (E_i^n - E_j^n)) \]

\[ + \sum_{a_{ij}} e_{ij} \delta t \int_{a_{ij}} (E_i^n - E_j^n) \cdot (\hat{n}_i \times \hat{n}_i \times (E_i^n - E_j^n)) \]

\[ + \sum_{a_{ij}} f_{ij} \delta t \int_{a_{ij}} (H_i^{n+1/2} - H_j^{n+1/2}) \cdot (\hat{n}_i \times \hat{n}_i \times (H_i^{n+1/2} - H_j^{n+1/2})) \]

\[ + \sum_{a_{ij}} f_{ij} \delta t \int_{a_{ij}} (H_i^{n+1/2} - H_j^{n+1/2}) \cdot (\hat{n}_i \times \hat{n}_i \times (H_i^{n-1/2} - H_j^{n-1/2})) \]

\[ = - \sum_{a_{ij}} e_{ij} \delta t \int_{a_{ij}} (\hat{n}_i \times (E_i^{n+1} - E_j^{n+1})) \cdot (\hat{n}_i \times (E_i^n - E_j^n)) \]

\[ - \sum_{a_{ij}} e_{ij} \delta t \int_{a_{ij}} (\hat{n}_i \times (E_i^n - E_j^n)) \cdot (\hat{n}_i \times (E_i^n - E_j^n)) \]

\[ - \sum_{a_{ij}} f_{ij} \delta t \int_{a_{ij}} (\hat{n}_i \times (H_i^{n+1/2} - H_j^{n+1/2})) \cdot (\hat{n}_i \times (H_i^{n+1/2} - H_j^{n+1/2})) \]

\[ - \sum_{a_{ij}} f_{ij} \delta t \int_{a_{ij}} (\hat{n}_i \times (H_i^{n+1/2} - H_j^{n+1/2})) \cdot (\hat{n}_i \times (H_i^{n-1/2} - H_j^{n-1/2})) \]

\[ = - \sum_{a_{ij}} e_{ij} \delta t \int_{a_{ij}} [E]^{n+1}_\gamma \cdot [E]^{n}_\gamma + [E]^{n}_\gamma \cdot [E]^{n}_\gamma \]

\[ - \sum_{a_{ij}} f_{ij} \delta t \int_{a_{ij}} [H]^{n+1/2}_\gamma \cdot [H]^{n+1/2}_\gamma + [H]^{n+1/2}_\gamma \cdot [H]^{n-1/2}_\gamma \]

Thus, we obtain the final result to be:

\[ E_{\text{global}}^{n+1} - E_{\text{global}}^n = \sum_i E_i^{n+1} - E_i^n \]

\[ = - \sum_{a_{ij}} e_{ij} \delta t \int_{a_{ij}} [E]^{n+1}_\gamma \cdot [E]^{n}_\gamma + [E]^{n}_\gamma \cdot [E]^{n}_\gamma \]

\[ - \sum_{a_{ij}} f_{ij} \delta t \int_{a_{ij}} [H]^{n+1/2}_\gamma \cdot [H]^{n+1/2}_\gamma + [H]^{n+1/2}_\gamma \cdot [H]^{n-1/2}_\gamma \]

(A.7)
Central Flux

From equation (A.7) it is clearly seen that for a central flux \( f_{ij} = e_{ij} = 0 \) the global energy \( \mathcal{E}^n_{\text{global}} \) as defined previously is conserved.

Upwind Flux

On the other hand for an upwind flux \( f_{ij} > 0, e_{ij} > 0 \) we will show that there is energy loss with respect to an appropriately defined energy. We follow an analysis similar to [26]. From equation (A.7) we have that:

\[
\mathcal{E}^{n+1}_{\text{global}} - \mathcal{E}^n_{\text{global}} = - \sum_{a_{ij}} \left( \frac{e_{ij} \delta t}{2} \int_{a_{ij}} [E]_\gamma^n \cdot [E]_\gamma^n + \frac{f_{ij} \delta t}{2} \int_{a_{ij}} [H]_{\gamma^n}^{1/2} \cdot [H]_{\gamma^n}^{1/2} \right)
\]

\[
+ \sum_{a_{ij}} \left( \frac{e_{ij} \delta t}{2} \int_{a_{ij}} [E]_\gamma^{n+1} \cdot [E]_\gamma^n + \frac{f_{ij} \delta t}{2} \int_{a_{ij}} [H]_{\gamma^n}^{1/2} \cdot [H]_{\gamma^n}^{1/2} \right)
\]

\[
\leq - \sum_{a_{ij}} \left( \frac{e_{ij} \delta t}{2} \int_{a_{ij}} [E]_\gamma^n \cdot [E]_\gamma^n + \frac{f_{ij} \delta t}{2} \int_{a_{ij}} [H]_{\gamma^n}^{1/2} \cdot [H]_{\gamma^n}^{1/2} \right)
\]

\[
+ \sum_{a_{ij}} \left( \frac{e_{ij} \delta t}{2} \left| \int_{a_{ij}} [E]_\gamma^{n+1} \cdot [E]_\gamma^n \right| + \frac{f_{ij} \delta t}{2} \left| \int_{a_{ij}} [H]_{\gamma^n}^{1/2} \cdot [H]_{\gamma^n}^{1/2} \right| \right)
\]

\[
\leq - \sum_{a_{ij}} \left( \frac{e_{ij} \delta t}{2} \int_{a_{ij}} [E]_\gamma^n \cdot [E]_\gamma^n + \frac{f_{ij} \delta t}{2} \int_{a_{ij}} [H]_{\gamma^n}^{1/2} \cdot [H]_{\gamma^n}^{1/2} \right)
\]

(A.8)

Continuing, by means of the Cauchy-Schwartz inequality and the estimate \( ab \leq \frac{1}{2} (a^2 + b^2) \) we arrive at:

\[
\mathcal{E}^{n+1}_{\text{global}} - \mathcal{E}^n_{\text{global}} \leq - \sum_{a_{ij}} \left( \frac{e_{ij} \delta t}{4} \left| [E]_\gamma^n \right|_{a_{ij}}^2 + \frac{f_{ij} \delta t}{4} \left| [H]_{\gamma^n}^{1/2} \right|_{a_{ij}}^2 \right)
\]

\[
+ \sum_{a_{ij}} \left( \frac{e_{ij} \delta t}{4} \left| [E]_\gamma^{n+1} \right|_{a_{ij}}^2 + \frac{f_{ij} \delta t}{4} \left| [H]_{\gamma^n}^{1/2} \right|_{a_{ij}}^2 \right)
\]

(A.9)
Thus finally by defining a better adapted energy \[26\],
\[
\tilde{E}^n_{\text{global}} = E^n_{\text{global}} - \sum_{a_{ij}} \frac{e_{ij} \delta t}{4} \|[E]_\gamma^n\|_{a_{ij}}^2 + \sum_{a_{ij}} \frac{f_{ij} \delta t}{4} \|[H]^{n-1/2}\|_{a_{ij}}^2
\]
we finally arrive at \(\tilde{E}^{n+1}_{\text{global}} - \tilde{E}^n_{\text{global}} \leq 0\). This adapted energy could be thought as the corrected energy to account for the loss of the upwind flux similar to the loss of the ABC in \[11\]. Thus, from (A.9) it is clear that for a upwind flux \((f_{ij} > 0, e_{ij} > 0)\) the global energy \(\tilde{E}^n_{\text{global}}\) as defined previously is decaying at every time step.

### A.1.2 Definite Positivity

In this section we will find the condition on \(\delta t\) under which \(\tilde{E}^n\) and \(E^n\) are positive definite quadratic forms.

**Remark 1** For upwind flux, since
\[
\tilde{E}^n_{\text{global}} = E^n_{\text{global}} - \sum_{a_{ij}} \frac{e_{ij} \delta t}{4} \|[E]_\gamma^n\|_{a_{ij}}^2 + \sum_{a_{ij}} \frac{f_{ij} \delta t}{4} \|[H]^{n-1/2}\|_{a_{ij}}^2
\]
it is sufficient to study \(\hat{E}^n = E^n_{\text{global}} - \sum_{a_{ij}} \frac{e_{ij} \delta t}{4} \|[E]_\gamma^n\|_{a_{ij}}^2\). If \(\hat{E}^n\) is a positive definite quadratic form then \(\tilde{E}^n\) is also, since the term + \(\sum_{a_{ij}} \frac{f_{ij} \delta t}{4} \|[H]^{n+1/2}\|_{a_{ij}}^2\) is always positive definite regardless of \(\delta t\).

We start with the discrete analog of the local energy in every element \(K_i\) defined as:
\[
E^n_i = \frac{1}{2} \epsilon_i \int_{K_i} E^n_i \cdot E^n_i + \frac{1}{2} \mu_i \int_{K_i} H^{n-1/2}_i \cdot H^{n+1/2}_i
\]
\[
= \frac{1}{2} \epsilon_i \int_{K_i} E^n_i \cdot E^n_i + \frac{1}{2} \mu_i \int_{K_i} H^{n-1/2}_i \cdot H^{n-1/2}_i - \frac{\delta t}{2} \int_{K_i} H^{n-1/2}_i \cdot \nabla \times E^n_i
\]
\[
+ \frac{\delta t}{4} \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} H^{n-1/2}_i \cdot \hat{n}_i \times (E^n_i - E^n_j) \quad \text{(A.10)}
\]
\[
+ \frac{\delta t}{4} \sum_{j \in \text{Neigh}(i)} f_{ij} \int_{a_{ij}} H^{n-1/2}_i \cdot \hat{n}_i \times \hat{n}_i \times (H^{n-1/2}_i - H^{n-1/2}_j)
\]
where \( f_{ij} = \frac{a}{(1/2)(\sqrt{\mu_i} + \sqrt{\mu_j})} \) for upwind flux and \( f_{ij} = 0 \) for central flux. Next, we use

\[
\sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} H_i^{n-1/2} \cdot (\hat{n} \times E_i^n) = \int_{K_i} \nabla \cdot (E_i^n \times H_i^{n-1/2})
\]

\[
= \int_{K_i} H_i^{n-1/2} \cdot \nabla \times E_i^n - E_i^n \cdot \nabla \times H_i^{n-1/2}
\]

to arrive at

\[
\mathcal{E}_i^n = \frac{1}{2} \varepsilon_i \int_{K_i} E_i^n \cdot E_i^n + \frac{1}{2} \mu_i \int_{K_i} H_i^{n-1/2} \cdot H_i^{n-1/2} - \frac{\delta t}{4} \int_{K_i} H_i^{n-1/2} \cdot (\hat{n} \times E_i^n)
\]

\[+ \frac{\delta t}{4} \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} H_i^{n-1/2} \cdot (\hat{n} \times E_j^n) - \frac{\delta t}{4} \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} H_i^{n-1/2} \cdot \hat{n} \times \hat{n} \times (H_i^{n-1/2} - H_j^{n-1/2}) \]

\[= \frac{1}{2} \varepsilon_i ||E_i^n||_{K_i}^2 + \frac{1}{2} \mu_i ||H_i^{n-1/2}||_{K_i}^2 - \frac{\delta t}{4} \chi_i^n \]

where

\[
\chi_i^n = \int_{K_i} H_i^{n-1/2} \cdot \nabla \times E_i^n + E_i^n \cdot \nabla \times H_i^{n-1/2}
\]

\[+ \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} H_i^{n-1/2} \cdot (\hat{n} \times E_j^n) - \sum_{j \in \text{Neigh}(i)} f_{ij} \int_{a_{ij}} H_i^{n-1/2} \cdot \hat{n} \times \hat{n} \times (H_i^{n-1/2} - H_j^{n-1/2}) \]

(A.12)

Here, \(||X_i||_{K_i}\) and \(||X_i||_{a_{ij}}\) denote the \(L^2\)-norm of the vector field \(X_i\) over volume \(K_i\) and interface \(a_{ij} = \partial K_i \cap \partial K_j\) respectively. Let us note that, \(\mathcal{E}_i^n = \frac{1}{2} \varepsilon_i ||E_i^n||_{K_i}^2 + \frac{1}{2} \mu_i ||H_i^{n-1/2}||_{K_i}^2 - \frac{\delta t}{4} \chi_i^n \geq \frac{1}{2} \varepsilon_i ||E_i^n||_{K_i}^2 + \frac{1}{2} \mu_i ||H_i^{n-1/2}||_{K_i}^2 - \frac{\delta t}{4} ||\chi_i^n||.\) By means of the Cauchy-Schwartz inequality,

\[
||\langle x, y \rangle || = ||x|| ||y|| \leq \left( \int_D |x|^2 \right)^{1/2} \left( \int_D |y|^2 \right)^{1/2} = ||x||_D ||y||_D \]

(A.13)

108
we can derive the following lower bounds:

\[ |\mathcal{X}_i^n| \leq \left| \int_{K_i} \mathbf{H}_i^{n-1/2} \cdot \nabla \times \mathbf{E}_i^n \right| + \left| \int_{K_i} \mathbf{E}_i^n \cdot \nabla \times \mathbf{H}_i^{n-1/2} \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{E}_j^n) \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} f_{ij} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{H}_j^{n-1/2} - \mathbf{H}_j^{n-1/2}) \right| \\
\leq \left| \int_{K_i} \mathbf{H}_i^{n-1/2} \cdot \nabla \times \mathbf{E}_i^n \right| + \left| \int_{K_i} \mathbf{E}_i^n \cdot \nabla \times \mathbf{H}_i^{n-1/2} \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{E}_j^n) \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} f_{ij} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{H}_j^{n-1/2} - \mathbf{H}_j^{n-1/2}) \right| \\
\leq \left| \int_{K_i} \mathbf{H}_i^{n-1/2} \cdot \nabla \times \mathbf{E}_i^n \right| + \left| \int_{K_i} \mathbf{E}_i^n \cdot \nabla \times \mathbf{H}_i^{n-1/2} \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{E}_j^n) \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} f_{ij} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{E}_j^n) \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} f_{ij} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{H}_j^{n-1/2} - \mathbf{H}_j^{n-1/2}) \right| \\
\leq \left| \int_{K_i} \mathbf{H}_i^{n-1/2} \cdot \nabla \times \mathbf{E}_i^n \right| + \left| \int_{K_i} \mathbf{E}_i^n \cdot \nabla \times \mathbf{H}_i^{n-1/2} \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{E}_j^n) \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} f_{ij} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{E}_j^n) \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} f_{ij} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{H}_j^{n-1/2} - \mathbf{H}_j^{n-1/2}) \right|
\leq \left| \int_{K_i} \mathbf{H}_i^{n-1/2} \cdot \nabla \times \mathbf{E}_i^n \right| + \left| \int_{K_i} \mathbf{E}_i^n \cdot \nabla \times \mathbf{H}_i^{n-1/2} \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{E}_j^n) \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} f_{ij} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{E}_j^n) \right| \\
+ \left| \sum_{j \in \text{Neigh}(i)} f_{ij} \int_{a_{ij}} \mathbf{H}_i^{n-1/2} \cdot (\mathbf{\hat{n}}_i \times \mathbf{H}_j^{n-1/2} - \mathbf{H}_j^{n-1/2}) \right| \\
= \left| \int_{a_{ij}} \mathbf{H}_i \cdot (\mathbf{\hat{n}}_i \times \mathbf{E}_j) \right| \leq \frac{1}{\sqrt{\epsilon_i \mu_i}} \int_{a_{ij}} |\sqrt{\mu_i} \mathbf{H}_i| |\sqrt{\epsilon_i} \mathbf{E}_j| \\
\leq \frac{1}{\sqrt{\epsilon_i \mu_i}} \left( \int_{a_{ij}} |\sqrt{\mu_i} \mathbf{H}_i|^2 \right)^{1/2} \left( \int_{a_{ij}} |\sqrt{\epsilon_i} \mathbf{E}_j|^2 \right)^{1/2} \\
\leq \frac{1}{2\sqrt{\epsilon_i \mu_i}} \left( \int_{a_{ij}} |\sqrt{\mu_i} \mathbf{H}_i|^2 \right) + \int_{a_{ij}} |\sqrt{\epsilon_i} \mathbf{E}_j|^2 \\
= \frac{1}{2} \sqrt{\frac{\mu_i}{\epsilon_i}} |\mathbf{H}_i|^2_{a_{ij}} + \frac{1}{2} \sqrt{\frac{\epsilon_i}{\mu_i}} |\mathbf{E}_j|^2_{a_{ij}} 
\]
Using A.16 we have that:

\[
|X_i^n| \leq \|\nabla \times \mathbf{E}_i\|_{K_i} \|H_i\|_{K_i} + \|\mathbf{E}_i\|_{K_i} \|\nabla \times H_i\|_{K_i} + \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \sqrt{\frac{\mu_i}{\epsilon_i}} \|H_i\|_{a_{ij}}^2 + \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \frac{\beta_{ij} s_{ij}}{V_i} \|H_i\|_{a_{ij}}^2
\]

\[
+ \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \sqrt{\frac{\epsilon_i}{\mu_i}} \|E_{ij}\|_{a_{ij}}^2 + \sum_{j \in \text{Neigh}(i)} f_{ij} \|H_i\|_{a_{ij}}^2 + \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \frac{\beta_{ij} s_{ij}}{V_i} \|H_j\|_{a_{ij}}^2
\]

\[
\leq \|\nabla \times \mathbf{E}_i\|_{K_i} \|H_i\|_{K_i} + \|\mathbf{E}_i\|_{K_i} \|\nabla \times H_i\|_{K_i} + \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \sqrt{\frac{\mu_i}{\epsilon_i}} \|H_i\|_{a_{ij}}^2
\]

\[
+ \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \sqrt{\frac{\epsilon_i}{\mu_i}} \|E_{ij}\|_{a_{ij}}^2 + \sum_{j \in \text{Neigh}(i)} f_{ij} \left( \|H_i\|_{a_{ij}}^2 + \frac{1}{2} \left( \|H_i\|_{a_{ij}}^2 + \|H_j\|_{a_{ij}}^2 \right) \right)
\]

\[
= \|\nabla \times \mathbf{E}_i\|_{K_i} \|H_i\|_{K_i} + \|\mathbf{E}_i\|_{K_i} \|\nabla \times H_i\|_{K_i} + \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \sqrt{\frac{\mu_i}{\epsilon_i}} \|H_i\|_{a_{ij}}^2
\]

\[
+ \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \sqrt{\frac{\epsilon_i}{\mu_i}} \|E_{ij}\|_{a_{ij}}^2 + \frac{3}{2} \sum_{j \in \text{Neigh}(i)} f_{ij} \|H_i\|_{a_{ij}}^2 + \frac{1}{2} \sum_{j \in \text{Neigh}(i)} f_{ij} \|H_j\|_{a_{ij}}^2
\]

(A.17)

Next, we will make use of the following definition (Definition 2 form H. Fahs [10]).

**Definition 1** We assume tensors \( \epsilon_i \) and \( \mu_i \) are piecewise constant, i.e. \( \epsilon_i = \bar{\epsilon}_i \) and \( \mu_i = \bar{\mu}_i \). Let \( c_i = 1/\sqrt{\epsilon_i \mu_i} \) be the propagation speed in the finite element \( K_i \). We also assume that there exist dimensionless constants \( \alpha_i \) and \( \beta_{ij} \) \((j \in \text{Neigh}(i))\) such that:

\[
\|\nabla \times X\|_{K_i} \leq \frac{\alpha_i P_i}{V_i}, \quad \|X\|_{a_{ij}}^2 \leq \frac{\beta_{ij} s_{ij}}{V_i} \|X\|_{K_i}^2 \quad \forall X \in [P^k(K)]^3
\]

(A.18)

where \( V_i \) is the volume of \( K_i \), \( s_{ij} \) is the area of \( a_{ij} \) and \( P_i = \sum_{j \in \text{Neigh}(i)} s_{ij} \). Using

**Definition 1** in A.17 one can arrive at the following:

\[
|X_i^n| \leq \frac{2\alpha_i c_i P_i}{V_i} \sqrt{\mu_i} \|H_i\|_{K_i} \sqrt{\epsilon_i} \|E_i\|_{K_i} + \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \sqrt{\frac{\mu_i}{\epsilon_i}} \frac{\beta_{ij} s_{ij}}{V_i} \|H_i\|_{K_i}^2
\]

\[
+ \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \sqrt{\frac{\epsilon_i}{\mu_i}} \frac{\beta_{ij} s_{ij}}{V_j} \|E_{ij}\|_{K_j}^2 + \frac{3}{2} \sum_{j \in \text{Neigh}(i)} f_{ij} \frac{\beta_{ij} s_{ij}}{V_i} \|H_i\|_{K_i}^2
\]

\[
+ \frac{1}{2} \sum_{j \in \text{Neigh}(i)} f_{ij} \frac{\beta_{ij} s_{ij}}{V_j} \|H_j\|_{K_j}^2
\]

(A.19)
Again by setting $a = \sqrt{H_i}||H_i||_{K_i}$, $b = \sqrt{\epsilon_i}||E_i||_{K_i}$ and using the inequality $ab \leq (a^2 + b^2)/2$ for the first lower bound term to arrive at:

$$
|A_i^n| \leq \sum_{a_{ij}} \frac{\alpha_i c_i s_{ij}}{V_i} (\mu_i ||H_i||_{K_i}^2 + \epsilon_i ||E_i||_{K_i}^2) + \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \sqrt{H_i} \frac{\beta_{ij} s_{ij}}{\epsilon_i} ||H_i||_{K_i}^2 + \frac{1}{2} \sum_{j \in \text{Neigh}(i)} \sqrt{\epsilon_i} \frac{\beta_{ij} s_{ij}}{\mu_i} ||E_j||_{K_j}^2 + \frac{3}{2} \sum_{j \in \text{Neigh}(i)} f_{ij} \frac{\beta_{ij} s_{ij}}{V_j} ||H_i||_{K_i}^2 
$$

(A.20)

Using $P_i = \sum_{j \in \text{Neigh}(i)} s_{ij}$, ($s_{ij}$ is the area of $a_{ij}$), we have:

$$
A_i^n = \frac{1}{2} \epsilon_i ||E_i||_{K_i}^2 + \frac{1}{2} \mu_i ||H_i||_{K_i}^2 = \sum_{j \in \text{Neigh}(i)} s_{ij} \left( \frac{1}{2P_i} \left( \frac{\alpha_i c_i \delta t}{V_i} \right) (\epsilon_i ||E_i||_{K_i}^2 + \mu_i ||H_i||_{K_i}^2) \right)
$$

(A.21)

which leads to

$$
\mathcal{E}_i^n \geq A_i^n - \delta t \frac{1}{4} |A_i^n| \geq \sum_{j \in \text{Neigh}(i)} s_{ij} \left( \frac{1}{2P_i} \left( \frac{\alpha_i c_i \delta t}{V_i} \right) (\epsilon_i ||E_i||_{K_i}^2 + \mu_i ||H_i||_{K_i}^2) \right)
$$

$$
- \frac{\delta t}{8} \sum_{j \in \text{Neigh}(i)} \sqrt{\frac{\epsilon_i}{\mu_i} \frac{\beta_{ij} s_{ij}}{V_i} ||H_i||_{K_i}^2} - \frac{\delta t}{8} \sum_{j \in \text{Neigh}(i)} \sqrt{\frac{\epsilon_i}{\mu_i} \frac{\beta_{ij} s_{ij}}{V_j} ||E_j||_{K_j}^2} - \frac{3\delta t}{8} \sum_{j \in \text{Neigh}(i)} f_{ij} \frac{\beta_{ij} s_{ij}}{V_j} ||H_i||_{K_i}^2 - \frac{\delta t}{8} \sum_{j \in \text{Neigh}(i)} f_{ij} \frac{\beta_{ij} s_{ij}}{V_j} ||H_j||_{K_j}^2
$$

(A.22)

This completes our lower bound for $\mathcal{E}_i^n$.

**Central flux ($f_{ij} = e_{ij} = 0$)**

For central flux, using (A.22) we get $\mathcal{E}_{global} = \sum_i \mathcal{E}_i^n \geq \sum_{a_{ij}} s_{ij} W_{ij}$, where

$$
W_{ij} = \left( \frac{1}{2P_i} - \frac{\alpha_i c_i \delta t}{4V_i} \right) \epsilon_i - \frac{\delta t}{8} \left( \frac{\epsilon_i \beta_{ij}}{\mu_i V_i} \right) ||E_i||_{K_i}^2
$$

$$
+ \left( \frac{1}{2P_i} - \frac{\alpha_i c_i \delta t}{4V_i} \right) \mu_i - \frac{\delta t}{8} \left( \frac{\mu_i \beta_{ij}}{\epsilon_i V_i} \right) ||H_i||_{K_i}^2
$$

$$
+ \left( \frac{1}{2P_j} - \frac{\alpha_j c_j \delta t}{4V_j} \right) \epsilon_j - \frac{\delta t}{8} \left( \frac{\epsilon_j \beta_{ij}}{\mu_j V_j} \right) ||E_j||_{K_j}^2
$$

$$
+ \left( \frac{1}{2P_j} - \frac{\alpha_j c_j \delta t}{4V_j} \right) \mu_j - \frac{\delta t}{8} \left( \frac{\mu_j \beta_{ij}}{\epsilon_j V_j} \right) ||H_j||_{K_j}^2
$$

(A.23)
Therefore, for central flux to achieve positive definiteness we need:

\[
\begin{align*}
||E_i||_{K_i}^2 \text{ and } ||H_i||_{K_i}^2 \text{ terms} & \quad ||E_j||_{K_j}^2 \text{ and } ||H_j||_{K_j}^2 \text{ terms} \\
\frac{1}{2P_i} - \frac{\alpha_i c_i \delta t_1}{4V_i} - \frac{\delta t_1 c_i \beta_{ij}}{8V_i} > 0 & \quad (A.24) \\
\frac{1}{2P_j} - \frac{\alpha_j c_j \delta t_2}{4V_j} - \frac{\delta t_2 c_i \beta_{ji}}{8V_j} > 0 & \quad (A.25) \\
\delta t_1 [2\alpha_i c_i + \beta_{ij} c_i] < \frac{4V_i}{P_i} & \quad (A.26) \\
\delta t_2 [2\alpha_j c_j + \beta_{ji} c_j] < \frac{4V_j}{P_j} & \quad (A.27)
\end{align*}
\]

Therefore, to ensure stability for element \( K_i \) we should choose
\[ \delta t_i = \min_{j \in \text{Neigh}(i)} \left( \delta t_1, \delta t_2 \right). \]

Furthermore, the choice \( \delta t = \min(\delta t_i) \) obviously would suffice for every element \( K_i \) of the mesh to be stable. Moreover, as shown in \[10\],
\[ \alpha_i, \beta_{ij} \text{ are given by:} \]
\[ \alpha_i = \sqrt{V_i^2 ||M^{-1/2} S_1 M^{-1/2}||/P_i^2}, \quad \beta_{ij} = V_i ||M^{-1/2} S_2 M^{-1/2}||/P_i \] (A.28)

where, \((M)_{nm} = \int_{K_i} w_{in} \cdot w_{im}, (S_1)_{nm} = \int_{K_i} \nabla \times w_{in} \cdot \nabla \times w_{im}, (S_2)_{nm} = \int_{f_i \cap f_j} w_{in} \cdot (\hat{n}_i \times w_{jm}) \). Finally, the norm \( ||M^{-1/2} S_k M^{-1/2}||, k = 1, 2 \) is computed as, \( max(\lambda_n) = ||M^{-1/2} S_k M^{-1/2}|| \) where \( \lambda_n \) are the eigenvalues of \( M^{-1/2} S_k M^{-1/2} \).

**Upwind flux** \((f_{ij} > 0, e_{ij} > 0)\)

For upwind flux, since
\[
\tilde{\mathcal{E}}_{\text{global}}^n = \mathcal{E}_{\text{global}}^n - \sum_{a_{ij}} \frac{e_{ij} \delta t}{4} ||[E]_\gamma^n||_{a_{ij}}^2 + \sum_{a_{ij}} \frac{f_{ij} \delta t}{4} ||[H]_\gamma^{n+1/2}||_{a_{ij}}^2
\]

it is sufficient to study \( \tilde{\mathcal{E}}_{\text{global}}^n = \mathcal{E}_{\text{global}}^n - \sum_{a_{ij}} \frac{e_{ij} \delta t}{4} ||[E]_\gamma^n||_{a_{ij}}^2 \). If \( \tilde{\mathcal{E}}_{\text{global}}^n \) positive definite quadratic form then \( \tilde{\mathcal{E}}_{\text{global}}^n \) is also, since the term \( + \sum_{a_{ij}} \frac{f_{ij} \delta t}{4} ||[H]_\gamma^{n+1/2}||_{a_{ij}}^2 \) is always positive definite regardless of \( \delta t \). We have derived a lower bound for \( \mathcal{E}_{\text{global}}^n \) and we
proceed to derive an upper bound for $\sum_{a_{ij}} \frac{e_{ij} \delta t}{4} ||[E]_\gamma^n||_{a_{ij}}^2$. We have that,

$$
\sum_{a_{ij}} \frac{e_{ij} \delta t}{4} ||[E]_\gamma^n||_{a_{ij}}^2 = \sum_{a_{ij}} \frac{e_{ij} \delta t}{4} \int_{a_{ij}} (\hat{n}_i \times E_i - \hat{n}_i \times E_j) \cdot (\hat{n}_i \times E_i - \hat{n}_i \times E_j)
$$

$$
= \sum_{a_{ij}} \frac{e_{ij} \delta t}{4} \int_{a_{ij}} |\hat{n}_i \times E_i|^2 + |\hat{n}_i \times E_j|^2
$$

$$
- \sum_{a_{ij}} \frac{2e_{ij} \delta t}{4} \int_{a_{ij}} (\hat{n}_i \times E_i \cdot \hat{n}_i \times E_j)
$$

$$
\leq \sum_{a_{ij}} \frac{e_{ij} \delta t}{4} \left( ||E_i||_{a_{ij}}^2 + ||E_j||_{a_{ij}}^2 \right)
$$

$$
+ \sum_{a_{ij}} \frac{2e_{ij} \delta t}{4} \int_{a_{ij}} (\hat{n}_i \times E_i \cdot \hat{n}_i \times E_j)
$$

$$
\leq \sum_{a_{ij}} \frac{e_{ij} \delta t}{4} \left( ||E_i||_{a_{ij}}^2 + ||E_j||_{a_{ij}}^2 \right)
$$

$$
+ \sum_{a_{ij}} \frac{2e_{ij} \delta t}{4} \left| \int_{a_{ij}} (\hat{n}_i \times E_i \cdot \hat{n}_i \times E_j) \right|
$$

$$
\leq \sum_{a_{ij}} \left( \frac{e_{ij} \delta t}{4} \left( ||E_i||_{a_{ij}}^2 + ||E_j||_{a_{ij}}^2 \right) + \frac{e_{ij} \delta t}{4} \left( ||E_i||_{a_{ij}}^2 + ||E_j||_{a_{ij}}^2 \right) \right)
$$

$$
\leq \sum_{a_{ij}} \left( \frac{e_{ij} \delta t}{8} \left( ||E_i||_{a_{ij}}^2 + ||E_j||_{a_{ij}}^2 \right) \right)
$$

$$
\leq \sum_{a_{ij}} \left( 4e_{ij} \frac{\beta_{ij}s_{ij}}{V_i} \frac{\delta t}{8} ||E_i||_{V_i}^2 + 4e_{ij} \frac{\beta_{ij}s_{ij}}{V_j} \frac{\delta t}{8} ||E_j||_{V_j}^2 \right)
$$

(A.29)

Therefore, we can obtain,

$$
\hat{\mathcal{E}}_{\text{global}}^n = \mathcal{E}_{\text{global}}^n - \sum_{a_{ij}} \frac{e_{ij} \delta t}{4} ||[E]_\gamma^n||_{a_{ij}}^2 \geq \sum_{a_{ij}} s_{ij} W_{ij} - \sum_{a_{ij}} \frac{e_{ij} \delta t}{4} ||[E]_\gamma^n||_{a_{ij}}^2
$$

$$
\geq \sum_{a_{ij}} s_{ij} W_{ij} - \sum_{a_{ij}} \left( 4e_{ij} \frac{\beta_{ij}s_{ij}}{V_i} \frac{\delta t}{8} ||E_i||_{V_i}^2 + 4e_{ij} \frac{\beta_{ij}s_{ij}}{V_j} \frac{\delta t}{8} ||E_j||_{V_j}^2 \right)
$$

(A.30)

$$
= \sum_{a_{ij}} s_{ij} \tilde{W}_{ij}
$$
where,
\[
\tilde{W}_{ij} = \left( \frac{1}{2P_i} - \frac{\alpha_i c_i \delta t}{4V_i} \right) \epsilon_i - \delta t \left( \frac{\epsilon_i \beta_{ij}}{\mu_i V_i} + \frac{4 e_{ij} \beta_{ij}}{\epsilon_i 8 V_i} \right) ||E_i||^2_{K_i}
\]
\[
+ \left( \frac{1}{2P_i} - \frac{\alpha_i c_i \delta t}{4V_i} \right) \mu_i - \delta t \left( \frac{\mu_i \beta_{ij}}{\epsilon_i V_i} + \frac{4 f_{ij} \beta_{ij}}{\epsilon_i 8 V_i} \right) ||H_i||^2_{K_i}
\]
\[
+ \left( \frac{1}{2P_j} - \frac{\alpha_j c_j \delta t}{4V_j} \right) \epsilon_j - \delta t \left( \frac{\epsilon_j \beta_{ji}}{\mu_j V_j} + \frac{4 e_{ij} \beta_{ji}}{\epsilon_j 8 V_j} \right) ||E_j||^2_{K_j}
\]
\[
+ \left( \frac{1}{2P_j} - \frac{\alpha_j c_j \delta t}{4V_j} \right) \mu_j - \delta t \left( \frac{\mu_j \beta_{ji}}{\epsilon_j V_j} + \frac{4 f_{ij} \beta_{ji}}{\epsilon_j 8 V_j} \right) ||H_j||^2_{K_j}
\]

(A.31)

Then, in order to have a positive definite quadratic form we require the following:

\[
||E_i||^2_{K_i} \text{ term}
\]
\[
\frac{1}{2P_i} - \delta t_1 \left( \frac{\alpha_i c_i}{4V_i} + \frac{c_i \beta_{ij}}{V_i} + \frac{4 e_{ij} \beta_{ij}}{\epsilon_i 8 V_i} \right) > 0 \quad (A.32)
\]
\[
\delta t_1 \left[ 2 \alpha_i c_i + \beta_{ij} c_i + 4 \beta_{ij} \frac{e_{ij}}{\epsilon_i} \right] < \frac{4V_i}{P_i} \quad (A.33)
\]

\[
||E_j||^2_{K_j} \text{ term}
\]
\[
\frac{1}{2P_j} - \delta t_2 \left( \frac{\alpha_j c_j}{4V_j} + \frac{c_j \beta_{ji}}{V_j} + \frac{4 e_{ij} \beta_{ji}}{\epsilon_j 8 V_j} \right) > 0 \quad (A.34)
\]
\[
\delta t_2 \left[ 2 \alpha_j c_j + \beta_{ji} c_j + 4 \beta_{ji} \frac{e_{ij}}{\epsilon_j} \right] < \frac{4V_j}{P_j} \quad (A.35)
\]

\[
||H_i||^2_{K_i} \text{ term}
\]
\[
\frac{1}{2P_i} - \delta t_3 \left( \frac{\alpha_i c_i}{4V_i} + \frac{c_i \beta_{ij}}{8V_i} + \frac{4 f_{ij} \beta_{ij}}{\mu_i 8 V_i} \right) > 0 \quad (A.36)
\]
\[
\delta t_3 \left[ 2 \alpha_i c_i + \beta_{ij} c_i + 4 \beta_{ij} \frac{f_{ij}}{\mu_i} \right] < \frac{4V_i}{P_i} \quad (A.37)
\]

\[
||H_j||^2_{K_j} \text{ term}
\]
\[
\frac{1}{2P_i} - \delta t_4 \left( \frac{\alpha_j c_j}{4V_j} + \frac{c_j \beta_{ji}}{8V_j} + \frac{4 f_{ij} \beta_{ji}}{\mu_j 8 V_j} \right) > 0 \quad (A.38)
\]
\[
\delta t_4 \left[ 2 \alpha_j c_j + \beta_{ji} c_j + 4 \beta_{ji} \frac{f_{ij}}{\mu_j} \right] < \frac{4V_j}{P_j} \quad (A.39)
\]
To ensure stability for element $K_i$ we should choose $\delta t_i = \min_{j \in \text{Neigh}(i)} (\delta t_1, \delta t_2, \delta t_3, \delta t_4)$. Furthermore, the choice $\delta t = \min(\delta t_i)$ obviously would suffice for every element $K_i$ of the mesh to be stable. Moreover, as shown in [10], $\alpha_i, \beta_{ij}$ are given by:

$$
\alpha_i = \sqrt{V_i^2 ||M^{-1/2}S_1M^{-1/2}||/P_i^2}, \quad \beta_{ij} = V_i ||M^{-1/2}S_2M^{-1/2}||/P_i \tag{A.40}
$$

where, $(M)_{nm} = \int_{K_i} w_{in} \cdot w_{im}$, $(S_1)_{nm} = \int_{K_i} \nabla \times w_{in} \cdot \nabla \times w_{im}$, $(S_2)_{nm} = \int_{f_i \cap f_j} w_{in} \cdot (\hat{n}_i \times w_{jm})$. Finally, the norm $||M^{-1/2}S_kM^{-1/2}||$, $k = 1, 2$ is computed as, $max(|\lambda_n|) = ||M^{-1/2}S_kM^{-1/2}||$ where $\lambda_n$ are the eigenvalues of $M^{-1/2}S_kM^{-1/2}$. 
Bibliography


[22] B. Krauter, M. Beattie, D. Widiger, Hao-Ming Huang, Jinwoo Choi, and Yong Zhan. Parallelized full package signal integrity analysis using spatially distributed


