Parallel Field-Oriented Computation for Electromagnetic Transient Simulation of Power System Components

by

Peng Liu

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Energy Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

The finite element method (FEM) has been commonly employed to solve the partial differential equations in different physical domains including structural analysis, heat transfer, and electromagnetics due to its field-oriented nature, high accuracy, and capability to consider complex geometries and material properties. As essential components in power system, traditional models of transformer and transmission line cannot sufficiently represent the often omitted but important physical phenomena like the ionized field, magnetic saturation, eddy currents, and hysteresis. Therefore, finite element models are given increasing consideration in electromagnetic transient (EMT) simulation to provide more accurate results and a comprehensive view of the physical details, which can assist engineers to make better decisions when designing and testing equipment.

However, when seeking accuracy and detailed field information, the computational cost of the finite element method substantially increases compared with conventional numerical models. The spatial discretization generates innumerable interconnected nodes and elements, which subsequently are assembled into a large system of equations to be solved with matrix solvers. The computational efficiency of the finite element solver, especially for transient studies and nonlinear problems that require repetitive matrix factorization, remains an intractable challenge. The prevalent trend of parallel processing using high-performance computing resources including multi-core CPUs, many-core GPUs, and field-programmable gate arrays (FPGAs) provides a possible solution to resolve the computation efficiency issue, yet the finite element solution procedure needs to be improved and adapted to parallelism and data dependency to efficiently utilize the parallel hardware resources.

In this thesis, parallelism is explored at both the node-level and element-level to make the finite element computation suitable for massively parallel processing. First, the nodal domain decomposition scheme is proposed to solve the finite element problem with node-level parallelism. Each finite element node and its neighbors make up a sub-domain and each sub-domain can be mapped to one computational unit and solved independently. The

assembling phase and matrix of the finite element method are avoided and the algorithms are perfected for single instruction multiple data (SIMD) stream hardware such as GPUs. The sub-domain solver works for both linear and nonlinear problems, and the mixed boundary conditions are incorporated in the sub-domain solver to accelerate the convergence. By fully using the massively parallel computing resources, the computational efficiency can be greatly improved compared with commercial software while maintaining high accuracy. The node-level parallelism is also extended for the finite difference method when computing the ionized field around high-voltage direct-current (HVDC) transmission line, and the Poisson's equation and the current continuity equation are solved alternatively on GPU to speed up the computation.

Second, the transmission line decoupling technique is employed to solve the nonlinear finite element problem at the element-level parallelism. Each element is decoupled from the interconnected network using the transmission line so the nonlinearity of the decoupled element can be solved independently on each computational core in parallel, which is suitable for SIMD hardware. Instead of repetitively factorizing the Jacobian matrix, a constant admittance matrix is required and matrix factorization is required once for all. Real-Time implementation is carried out on FPGA to explore the hardware concurrency and data pipelining for a 2D nonlinear finite element transformer model.

In addition, to interface the finite element model with the electrical network, an indirect field-circuit coupling scheme is proposed to extract and exchange the coupling coefficients at each time-step. Multi-physics finite element simulation considering thermal effects is also discussed in this thesis.

Preface

The dissertation is an original intellectual product of the author Peng Liu and all the work presented was conducted in the Real-Time Experimental Laboratory (RTX-Lab) at the University of Alberta. Peng Liu was the lead investigator, responsible for all major areas of concept formation, data collection, and analysis, as well as manuscript composition. Venkata Dinavahi was the supervisory author and was involved throughout the work in the idea formation and manuscript improvement. Ning Lin contributed to the development of the modular multilevel converter (MMC) in Chapter 6.

Chapter 3 includes some contents from the following paper:

- Peng Liu and Venkata Dinavahi, "Matrix-free nodal domain decomposition with relaxation for massively parallel finite element computation of EM apparatus", *IEEE Trans. on Magnetics*, vol. 54, no. 9, Jul 2018.

Chapter 4 contains results published in the following papers:

- Peng Liu and Venkata Dinavahi, "Real-time finite-element simulation of electromagnetic transients of transformer on FPGA", *IEEE Trans. on Power Delivery*, vol. 33, no. 4, pp. 1991-2001, Mar 2018.

- Peng Liu, Jiacong Li, and Venkata Dinavahi, "Matrix-free nonlinear finite element solver using transmission-line modeling on GPU", *IEEE Trans. on Magnetics*, vol. 55, no. 7, Jul 2019.

A version of Chapter 5 has been published in the following paper:

- Peng Liu and Venkata Dinavahi, "Finite-difference relaxation for parallel computation of ionized field of HVDC lines", *IEEE Trans. on Power Delivery*, vol. 33, no. 1, pp. 119–129, Jan 2017.

Chapter 6 is based on the contents of the following published paper:

- Peng Liu, Ning Lin, and Venkata Dinavahi, "Integrated thermo-electromagnetic field-transient massively parallel simulation of converter transformer interaction with MMC in multi-terminal DC grid", *IEEE Trans. on Electromagnetic Compatibility*, accepted for publication, IEEE early access, May 2019.

For my wife and parents.

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Venkata Dinavahi for all the support and academic guidance during my study at the University of Alberta. Dr. Dinavahi has always been enthusiastic, supportive and patient, and I'm deeply influenced and motivated by his passion for conducting innovative research.

Besides, I want to express my thanks to all the graduate students including Jiacong Li, Qingjie Xu, and Ning Lin in RTX-Lab who are very talented and knowledgeable in their research areas. Communicating with them has been enjoyable and expanded my knowledge. And to my pastor Dr. Nie and my friends in Edmonton, who are warm-hearted and spiritually supportive.

My love for my wife Sophie Gao and my parents.

To God be the glory!

Table of Contents

List of Tables

List of Figures

List of Acronyms

AC	Alternating Current
BVP	Boundary Value Problem
CAE	Computer-Aided Engineering
CG	Conjugate Gradient
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DC	Direct Current
DD	Domain Decomposition
EMT	Electromagnetic Transient
FDM	Finite Difference Method
FE	Finite Element
FEM	Finite Element Method
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Unit
GPGPU	General-Purpose Graphics Processing Unit
HLS	High-Level Synthesis
HPC	High-Performance Computing
HVDC	High-voltage Direct Current
LU	Lower-Upper
LUT	Look-Up Table
MEC	Magnetic Equivalent Circuit
MIMD	Multiple Instruction Multiple Data
NDD	Nodal Domain Decomposition
NF	Newton-Raphson
PDE	Partial Differential Equation
SDS	Sub-domain Solver
SIMD	Single Instruction Multiple Data
SISD	Single Instruction Single Data
SM	Streaming Multi-processors
TLM	Transmission Line Modeling
VHDL	Very-high-speed-integrated-circuit Hardware Description Language

1 Introduction

Advances in computer-aided engineering (CAE) have made simulation tools prevalent for solving a broad range of engineering problems in the power system. Electromagnetic transient (EMT) simulation plays an important role for the design and test of equipment under different operating conditions, and nowadays, the representation of different electrical and magnetic components tends to be device-leveled, highly-accurate, and including more physical details, thus causing the finite element method more and more widely employed.

This chapter will introduce the background and motivation conducting the research topic of parallel finite element computation in electrical engineering, including the prevalent high-performance computing trend, existing state-of-the-art technique, a brief review of commercial software, and the main challenges of the topic. The summarized contributions and the thesis outline are provided at the end of this chapter.

1.1 Finite Element Method for EMT Simulation

Lumped-element models of different components have been widely developed and used in an EMT program [1], however, for some magnetic components such as transformers, motors, and generators with the material nonlinearity, eddy currents, saturation, and field distributions to consider, traditional lumped-element models are incapable of providing detailed and accurate information. Under this circumstance, the field-oriented Maxwell's equations can be used to fully represent the material properties and physical details, and provide the most accurate results compared with other simplified or equivalent model. Among all the numerical methods to solve Maxwell's equation, the finite element method is the most prevalent due to its flexibility to mesh complex geometries [2]. Since the number of finite element nodes to solve can reach thousands or even millions, the solution

efficiency usually depends on the sparse matrix solver, which is computationally expensive and regarded as a notable problem especially for nonlinear and transient analysis that involves repeated matrix factorization. Essentially, it is a trade-off between modeling accuracy and computational efficiency when choosing different models.

The transformer models for EMT studies include the admittance matrix representation, star-circuit representation, and the magnetic equivalent circuit (MEC) representation [3], [4], and these representations essentially are solving the Ampere's law in Maxwell's equation based on different simplifications. For example, the lumped inductances are usually assumed constant while in fact, the values are time-varying and determined by the winding currents and saturation conditions; In the MEC representation, the magnetic flux through each branch is assumed constant, which is not true. On the contrary, a finite element transformer model solves the magnetic potentials in Ampere's law directly and relies on fewer assumptions. Therefore, it is usually acknowledged that the finite element model is the most accurate and comprehensive to represent a transformer, and the finite element model is irreplaceable in applications like fault analysis. However, the bottleneck that limits its application in EMT is the computational efficiency because the domain discretization will generate numerous nodes and elements to solve. The design and test cycles in real application can be reduced by improving the computational efficiency of finite element modeling, especially when the transient simulation needs to run repetitively under different parameters and setups.

Another example of the importance of the finite element method in power system is the calculation of the ionized field around the high-voltage transmission lines [5]. Existing line models such as the traveling wave model and serial π models can barely describe the corona caused by high-voltage discharging, and the coupled Poisson's equation and the current continuity equation governing the ionized field have to solved iteratively using field-oriented numerical methods. Thus, due to the iterative computation of the coupled partial differential equations, computational efficiency also remains a concern for the ionized field computation [5].

Although the finite element method is capable of modeling physical phenomena accurately and comprehensively, the computational efficiency problem remains a challenge in many physical areas such as structural analysis [6], heat transfer [7], [8], and electromagnetics [2], and improving the computational efficiency will shorten the design cycle in different applications. The urgency of fast computation is especially high for EMT studies because 1) Newton-Raphson iteration is required to handle the nonlinear magnetic material property, and 2) transient studies usually include thousands to millions of time-steps to solve.

The primary objective of this thesis is to resolve the computational efficiency problem of the finite element model of the power transformer and transmission line utilizing parallel computing. To achieve this goal, a glance of the finite element method is necessary. The

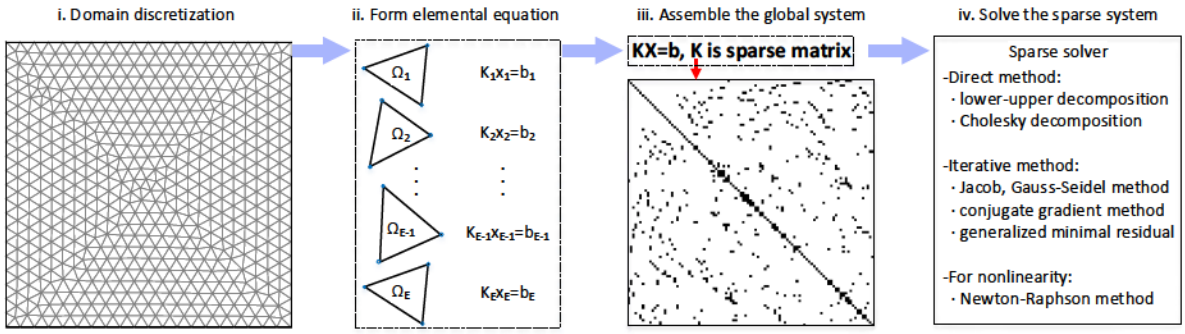


Figure 1.1: Procedure of traditional finite element method.

finite element analysis for different problems has hitherto shared the general procedure as follows [9]:

- Meshing: discretize the problem domain to many small interconnected nodes and elements (e.g. triangular or quadrangular for 2D, tetrahedral for 3D).
- Solving: for each element, interpolate and obtain the elemental equation using different strategies (variational principle, Galerkin, Ritz, etc.)
- Solving: assemble all the elemental equations to a global linear system of equations that involves a large sparse matrix.
- Solving: apply boundary conditions and solve the linear system using the sparse solver technique, applying Newton-Raphson iteration for nonlinear problems.
- Post-processing: obtain the results, plot, and display.

For the three phases (or five steps) described above, this thesis will focus on exploring parallel algorithms to speed up the finite element computing only in the solving phase. In other words, the algorithm used to generate mesh will not be involved and the mesh information is assumed given. The problem will be solved with the proposed parallel finite element solver with mesh information and other necessary input parameters, and the solution is then processed for plotting and displaying using tools like Matlab.

1.2 State of the Art: Parallel Finite Element Computation

1.2.1 High-Performance Computing

It is natural to mention the prevalent high-performance computing (HPC) trend when it comes to the computational efficiency issue. In the past decade, the computing power of a workstation can be hardly enhanced by solely increasing the working frequency; instead, more and more computing cores are integrated to increase the computing capability of

Table 1.1: Data-sheet of Nvidia Tesla V100 GPU [12].

	Tesla V100 PCIe	Tesla V100 SXM2	Tesla V100S PCIe
GPU Architecture	NVIDIA Volta		
NVIDIA Tensor Cores	640		
NVIDIA CUDA® Cores	5,120		
Double-Precision Performance	7 TFLOPS	7.8 TFLOPS	8.2 TFLOPS
Single-Precision Performance	14 TFLOPS	15.7 TFLOPS	16.4 TFLOPS
Tensor Performance	112 TFLOPS	125 TFLOPS	130 TFLOPS
GPU Memory	32GB /16GB HBM2		32GB HBM2
Memory Bandwidth	900GB/sec		1134 GB/sec
ECC	Yes		
Interconnect Bandwidth	32 GB/sec	300 GB/sec	32 GB/sec
System Interface	PCIe Gen3	NVIDIA NVLink	PCIe Gen3
Form Factor	PCIe Full Height/Length	SXM2	PCIe Full Height/Length
Max Power Comsumption	250 W	300 W	250 W
Thermal Solution	Passive		
Compute APIs	CUDA, DirectCompute, OpenCL™, OpenACC		

different architectures such as the general-purpose graphics processing unit (GPGPU) and computer clusters.

For example, the summit supercomputer has 4608 workstations and each workstation is equipped with dual IBM Power9 CPUs and 6 Nvidia Tesla GPUs [10]. Table 1.1 shows the data-sheet of one Nvidia Tesla V100 with 5120 Cuda cores and 32GB HBM2 memory. It can be inferred that the summit supercomputer has over 110 thousands of CPU cores and 140 millions of Cuda cores available for high-performance computing.

Besides, field-programmable gate arrays (FPGA) are also integrating more and more hardware resources to satisfy the growing needs for high-performance scientific computing; for example, the Xilinx VCU118 developing board with xcvu9p FPGA has more than 1.18 million look-up tables, 2.36 million flip-flops and 6840 DSP slices for massively parallel processing [11]. Also, the FPGA can be configured for data pipelining by fully unroll a computational instance.

However, it is important to mention that the parallel computing resources only provide the necessary hardware and don't sufficiently lead to the acceleration of the finite element computation. According to Amdahl's law [13], the speedup of a program is limited by the fraction of the program that can be parallelized, implying that the parallelism of the algorithm is the prerequisite to achieve good computational efficiency via parallel computing

resources. It seems quite direct that a sequential algorithm cannot benefit from whatever amount of parallel resources, and suitable parallel algorithms for the finite element method is required to fully utilize the parallel computing resources.

Nowadays, the finite element computation is being deployed in high-performance computing environments, and it remains a challenge to execute finite element computation with massive parallelism on these computing resources having single instruction multiple data (SIMD) paradigm. Among the most commonly used parallel strategies are the parallel sparse matrix solvers (direct solver and iterative solver) and the domain-decomposition based algorithms [2]. A brief introduction of the existing parallel strategies is presented below.

1.2.2 Parallel Sparse Matrix Solver

As shown in Fig. 1.1, assembly of the elemental equations will generate a system of equations $KX = b$ to be solved. Since each finite element node is directly connected only with its neighboring nodes, implying most entries in each row of the matrix are zeros, the stiffness matrix K is sparse. Therefore, the computational efficiency of the finite element solution usually depends on the sparse solver employed. For nonlinear problems, an iterative scheme such as Newton-Raphson is required to solve the nonlinearity, and for each iteration, a linear system $J\Delta x = r$ with the Jacobian matrix J still needs to be solved for the vector increment Δx [9].

There are two categories of methods to solve a sparse system: the direct methods [14] and iterative methods [15]. Direct methods including the Gaussian elimination, frontal solver, lower-upper (LU) factorization, and Cholesky decomposition seek to solve the system analytically and obtain an accurate solution after a finite number of mathematical operations [14]. All direct methods are associated with matrix factorization combined with forwarding (for lower triangular matrix) and backward substitution (upper triangular matrix). Although most of the operations of the matrix factorization are sequential, some parallelism exists in different phases and can be utilized for parallel processing [14].

Among the most popular parallel sparse solvers are the SuperLU [16], PARallel DIrect sparse SOlver (PARDISO) [17], and MUltifrontal Massively Parallel sparse direct Solver (MUMPS) [18], [19]. All these solvers are LU based and explore parallelism during the matrix factorization to make use of the multi-core CPUs of a modern computer. Issues will arise when the problem size becomes very large such as the accumulation of the roundoff errors caused by floating-point arithmetic and the memory requirement to store a large number of intermediate results.

Iterative methods, on the other hand, explore the solution iteratively and seek for approximate solution satisfying the user-defined error instead of an accurate solution. All the iterative methods start with an initial guess of the solution, and then the error between the updated solution and the final solution gradually diminishes through iterations. The

solution converges and the error can be controlled by users. Within each iteration, the roundoff error will not accumulate and impair the final converged solution. The iterative solver is memory-friendly since the required memory increases linearly with the problem size.

The most commonly used iterative methods include the Jacobi and Gauss-Seidel method for diagonally dominant matrix, conjugate gradient (CG) method for symmetric and positive-definite matrix and the generalized minimal residual (GMRES) for others [15]. The number of iterations required for convergence heavily relies on the initial guess and preconditioning is usually required to accelerate the iterative process. The main mathematical operations during each iteration are matrix-vector product Ax and vector inner product $x_1 x_2^T$, which is very suitable for massively parallel processing. The conjugate gradient solver has been implemented on GPU to explore parallelism for the matrix-vector multiplication and speedup the computation [20]–[22].

Recently, the new class of assembly-free or matrix-free solution technique has also been proposed to fully utilize the parallel computing resources to solve the finite element problem without assembling all the elemental equations to a global matrix [23]–[25]. Besides, the matrix-free fashion is also applied to a so-called element-by-element scheme and the matrix-vector product is done at each finite element without explicitly exploring sparse matrix [26], [27]. The new method follows a divide-and-conquer strategy and is suitable for massively parallel processing.

As for the nonlinearity of magnetic materials, the Newton-Raphson iterative scheme is commonly used. It usually starts with an initial solution and then repeatedly solves the Jacobian matrix to obtain the increment until convergence. The Jacobian matrix is also sparse, and it keeps changing and needs assembly at each iteration, the parallel sparse solvers aforementioned are also required for nonlinear problems.

1.2.3 Domain Decomposition Method

Another class of parallel strategy to solve a finite element problem is the domain decomposition technique [28]. The number of floating-point operations required to factorize a matrix will increase dramatically when the matrix size increases, for example, the complexity of Gauss elimination and Cholesky decomposition is $O(n^3)$. Therefore, for large-scale finite element problems, it may be more efficient to partition the problem to some smaller sub-domains to solve instead of solving the large system directly, provided the fact that the smaller sub-domains can be solved independently using parallel processing.

On the contrary to the merits of smaller sub-domains that can be solved in parallel, the drawback is that iterations are required for sub-domains to exchange information. Essentially, the domain decomposition scheme is also an iterative method, which shares the common features of iterative methods such as initial guess of solutions, approximate results, and convergence issue.

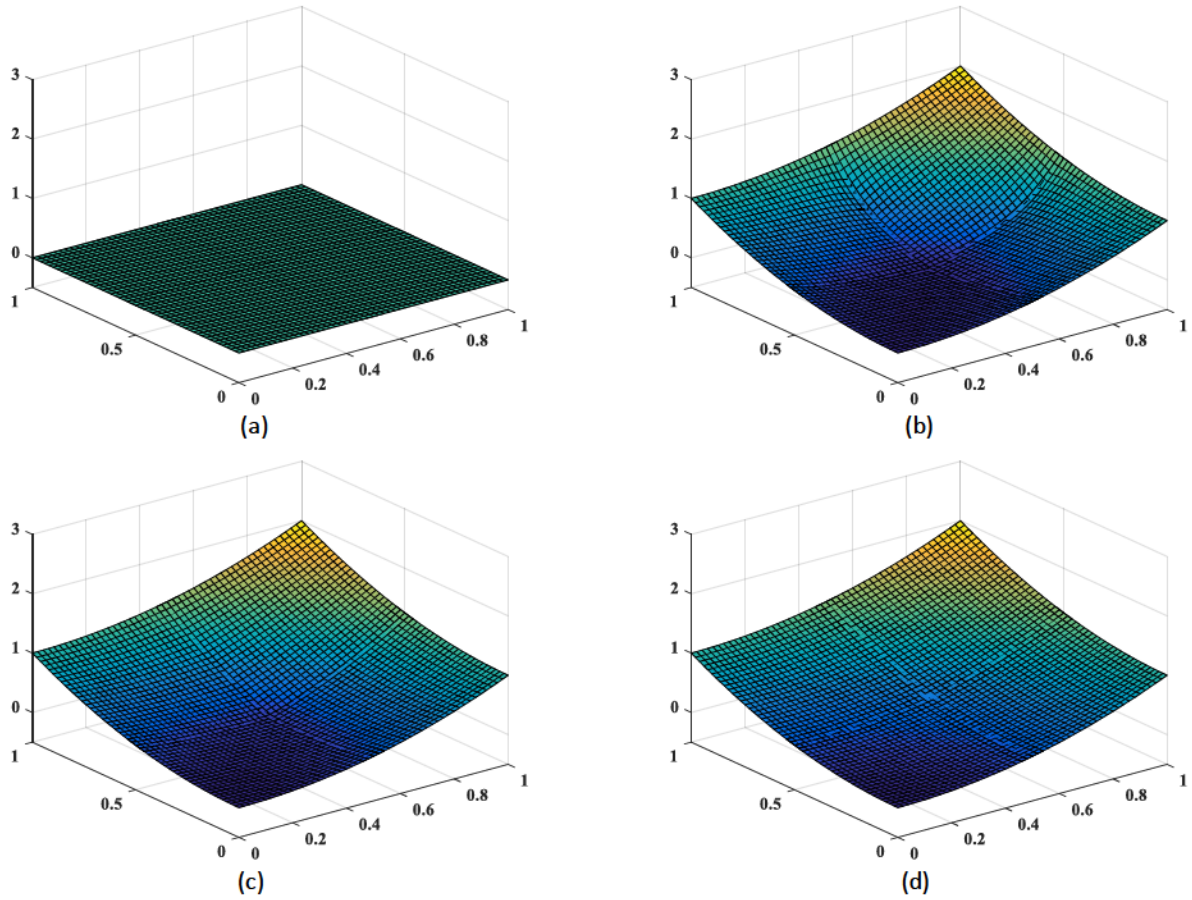


Figure 1.2: Domain decomposition through iterations: (a) initial value, (b) 2 iterations, (c) 5 iterations, and (d) 10 iterations.

Figure 1.2 shows a simple 2D boundary value problems solved using the domain decomposition method. The problem is divided into 4 sub-domains with some overlaps, and the solution progressively converges to the analytical solution, which is $u(x, y) = x^2 + y^2$ in this case. At each iteration, the 4 sub-domains can be solved with multiple computational cores in parallel, and then the sub-domains exchange boundary conditions in the overlapped zone for the next iteration. Although iterations are required, matrix factorization of sub-domains happens only once for all in the beginning, and the main operations are the forward and backward substitution for each iteration. It is worth to mention that the domain decomposition method works regardless of the linearity or nonlinearity of each sub-domain.

Since there are different sub-domain partition strategies, the domain decomposition method can be divided into overlapping (Schwarz) scheme such as shown in Fig. 1.2 and non-overlapping (or disjoint) scheme with sub-domains sharing only boundaries [28]. Based on the iterative method applied, i.e., to solve the sub-domains in parallel or sequentially, there are multiplicative and additive solving schemes. Also, there are different kinds

of implementation for the information exchange between sub-domains, for example, the Dirichlet type, the Neumann type, and the Robin type [28].

Although iterations are required for domain decomposition method, the reduced problem size and parallelized computation for each sub-domain have made it possible for high-performance computing using multiple CPUs. It is also worth to mention that the matrix factorization for each sub-domain is required only once, and most calculations through iterations are mere forward and backward substitution.

1.2.4 Glance of Commercial Software

The most commonly used commercial software for finite element modeling of transformer, inductor, and other electrical machines include Comsol Multiphysics™ AC/DC module [29] and Ansys Maxwell™ [30]. As finite element software, both Comsol™ and Ansys™ are capable of modeling 2D and 3D geometries, generating the user-controlled mesh, providing built-in and user-defined linear and nonlinear materials, deploying state-of-the-art sparse solvers, and plentiful of features for post-processing. Also, multi-physics and multi-domain simulation are included to consider the interaction of the electromagnetic field and the thermal field.

Ansys™ is more industry-oriented and provides convenience from the point of view of an engineer. Since in reality transformers are fully coupled with the external electrical circuit and it is inevitable to consider the interaction of finite element model with electromagnetic transients of power electronics or power system. Ansys Maxwell™ can perform transient field-circuit co-simulation when connected with Ansys Simplorer™, which provides system-level transient simulation with lots of built-in libraries for detailed device-level components. However, Ansys Maxwell™ reveals few details of its solving procedures such as the mesh information and provides little flexibility for the solver setup, making it difficult to reproduce its results using different parallel solvers.

On the contrary, Comsol™ is more academic-friendly since the whole solving procedure is transparent. All the intermediate data including the mesh information and the assembled matrix can be accessed and exported. The different built-in sparse solvers including CG, PARDISO, and MUMPS mentioned before can be selected and configured in detail. Besides, the LiveLink™ with Matlab and the equation-based modeling feature provide much convenience to solve different boundary value problems. One shortcoming regarding the electromagnetic transient is that the library to build external electrical circuits includes only basic AC/DC sources and RLC components, which is not capable of complex circuit simulations with different components like transistor and transmission line.

Both Comsol™ and Ansys™ have released the high-performance computing (HPC) license and parallel algorithms such as the domain decomposition method to parallelize its solver on multiple CPUs to accelerate the computation. Recently, since the GPU is

integrating more and more resources, Ansys™ is developing its GPU solver to further speedup the solution phase.

Despite the resourceful hardware, the future trend of high-performance scientific computing, especially the burdensome finite element analysis, still calls for suitable parallel algorithms to fully explore the computing power of HPC resources.

1.3 Motivation and Challenges

Massively parallel computing resources like GPU follow the single instruction multiple data (SIMD) paradigm and are most suitable for vectorized computation. The existing parallel sparse solvers of commercial software aforementioned explore the parallelism at different phases of matrix factorization and benefit from multiple CPUs. However, most operations of the matrix factorization, including the forward and backward substitution, are sequential in nature, the computation can be hardly vectorized to fully utilize SIMD hardware. Solving a nonlinear finite element problem in a massively parallel fashion remains a huge challenge and requires much investigation.

The prevalent massively parallel computational resources featured SIMD provide great potentials to accelerate the finite element computation, especially for transient simulation of nonlinear problems in the time domain. The commercial finite element solvers, usually accommodated for CPUs, have not involved vectorized computation on SIMD hardware. Therefore, exploring suitable parallel algorithms for SIMD hardware, with expected features like vectorization and little communication between cores, would be very crucial for future parallel finite element computation in high-performance computing environments.

The main challenges confronted include the following aspects:

- The previous finite element method, whether parallelized or not, follows the stereotype that the elemental equation must be assembled to a global system to solve. If a global matrix is assembled by following the traditional finite element procedures, the sparse solvers are essential, and it is very difficult to fully utilize the SIMD hardware. Gathering all the information from each element and node into a global system will somehow impair its potential for decentralized or parallel computation. Some out-of-box thinking is required to alter the traditional solution procedure to explore suitable parallel solving technique on SIMD hardware.

- To solve the nonlinearity, the most popular method has been the Newton-Raphson iterative method. For each iteration, the Jacobian matrix keeps changing and assembling and factorizing matrices have to be repeated, which causes the heavy computation burden for nonlinear problems. The nonlinearity requires to be considered and accommodated for parallel processing.

- The solution procedure should also be memory-friendly and avoid numerous intermediate parameters that increase immensely with problem size, especially regarding the matrix-related operations;
- For power transformers, the input is the current density and the output is the magnetic potential according to Ampere's law. Thus the field circuit coupling issue should be resolved to include complex electrical circuits connected to the primary and secondary windings.

1.4 Contributions of the Thesis

The following contributions are made when exploring suitable algorithms for massively parallel finite element computation on SIMD hardware:

- A new finite element solution procedure named nodal domain decomposition method that avoided assembling to any matrices was proposed to explore parallelism at the finite element node level. The computation is entirely decentralized at each node and the computation of each node shares the same pattern and can be accomplished with one single program (sub-domain solver), thus easily vectorized for the SIMD pattern. The proposed method works for both linear and nonlinear problems, and the convergence rate was improved over 4 times by exchanging information between sub-domains with mixed boundary conditions. The execution time of the proposed method on GPU is over 30 times faster than commercial software.
- The transmission-line decoupling technique was employed for nonlinear finite element solutions to explore parallelism at the finite element level. Each element is decoupled from the interconnected system using a virtual transmission line so the nonlinearity of the decoupled element can be solved independently in parallel. A constant admittance matrix is required in the solution without repeatedly factorizing the Jacobian matrix. The adapted admittance matrix scheme was proposed to greatly reduce the number of required iterations by adding a couple of matrices to factorize. The transmission-line decoupling technique was implemented on FPGA to achieve real-time finite element computation for the first time: the execution time drops from several milliseconds to 84 μs for a nonlinear finite element transformer model with 196 unknowns.
- The nodal level parallelism was also extended for the finite difference method used to solve the ionized field around high-voltage direct-current transmission line. The Poisson's equation and current continuity equation were iteratively solved with the parallel finite difference method with differentiated grid sizes on GPU, increasing the computational efficiency of more than 30 times.

Considering the application of a fast finite element model in electromagnetic transient studies, the following contributions are made:

- An indirect field-circuit coupling technique was proposed to extract the self and mutual inductances from the winding zones of the finite element transformer at each time-step, and these parameters can be used by the external electrical circuits connected to the windings.
- The Joule effect of the winding current was also considered in a multi-physics case study. The electrical circuits, magnetic field, and thermal field were solved independently using coupling coefficients.
- The parasitic capacitance of the transformer model was extracted from a static electric field to consider the transients under high-frequency conditions.

1.5 Thesis Outline

This thesis includes 7 chapters:

- **Chapter 1: Introduction** - The background, motivation and objectives are briefly summarized.
- **Chapter 2: Finite Element Method and Parallel Computing** - In this chapter, the finite element method and the current high-performance computing trend is introduced, and a core question that motivates this work is proposed: how the finite element method can benefit from these massively parallel computing resources?
- **Chapter 3: Node-Wise Parallelism: Nodal Domain Decomposition** - This chapter proposes a novel finite element computing procedure that explores parallelism at each finite element node, and the computation of each finite element node can be mapped to each computational core for both linear and nonlinear problems to utilize high-performance computing resources such as GPU. The exchange of mixed boundary conditions between sub-domains is explored to accelerate the convergence.
- **Chapter 4: Element-Wise Parallelism: Transmission-Line Decoupling** - A computing procedure that explores parallelism at each triangular element level is explored. The computation of each triangular element is decoupled from the network and thus achieves massive parallelism. Besides, for nonlinear problems, the admittance matrix remains unchanged and matrix factorization is required once for all.
- **Chapter 5: Parallel Finite-Difference Computation** -The node-wise parallelism is applied for the finite difference method for the computation of the ionized field

around HVDC transmission lines. The coupled partial differential equations governing the ionized field are alternatively solved on GPU with differentiated finite-difference grids.

- **Chapter 6: Application for Electromagnetic Transient Studies** - In this chapter, the application of the parallel algorithms is discussed with electromagnetic transient studies, including the field-circuit coupling and multi-physics finite element simulation considering thermal effects.

- **Chapter 7: Conclusions** - This chapter presents the conclusions of the thesis and discusses future work.

The chapters are organized as follows:

Chapter 1 and Chapter 2 present the necessary background for EMT simulation, finite element method, and parallel computing.

Then, motivated by how to divide the finite element problem into numerous small pieces for massively parallel processing, Chapter 3 and Chapter 4 describe the proposed node-wise and element-wise parallel algorithms implemented on GPU and FPGA, respectively.

Chapter 5 can be regarded as an extension of the node-wise parallelism for the finite different method, which is utilized to solve the burdensome ionized field problem via parallel computing.

Chapter 6 describes an efficient multi-physics simulation model with the interfacing of modular multi-level converter and finite element models and proves how the runtime of such an integrated model can be reduced from weeks to hours.

At last, Chapter 7 summarizes the thesis and gives the conclusion.

2

Finite Element Method and Parallel Computing

2.1 Introduction

The finite element method is one of the most commonly used numerical methods to solve boundary value problems. To elaborate on how the finite element computation can be performed in parallel, an introduction of the finite element method solving partial differential equations and some background regarding parallel processing will be presented in this chapter.

2.2 Boundary Value Problem

Figure 1.2 in Chapter 1 shows a simple example of a boundary value problem (BVP), which is solved with the domain decomposition (DD) scheme. Figure 2.1 shows the same example without the DD method. In both cases, the three parts that define a boundary value problem are:

a) Problem Domain: The problem domain shows the interested domain in which the physical problem has a definition. The problem domain is usually defined by boundaries that can make up a closed-loop. In this example, the problem domain is $x \in [0, 1]$, $y \in [0, 1]$.

b) Governing Equation: The equation that determines the physical phenomenon within the problem domain. The equation is usually ordinary partial equation (ODE) for a 1D problem and partial differential equation (PDE) for 2D and 3D problems. The governing equation can also be an integral equation. In the above example, the governing equation is

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 4. \tag{2.1}$$

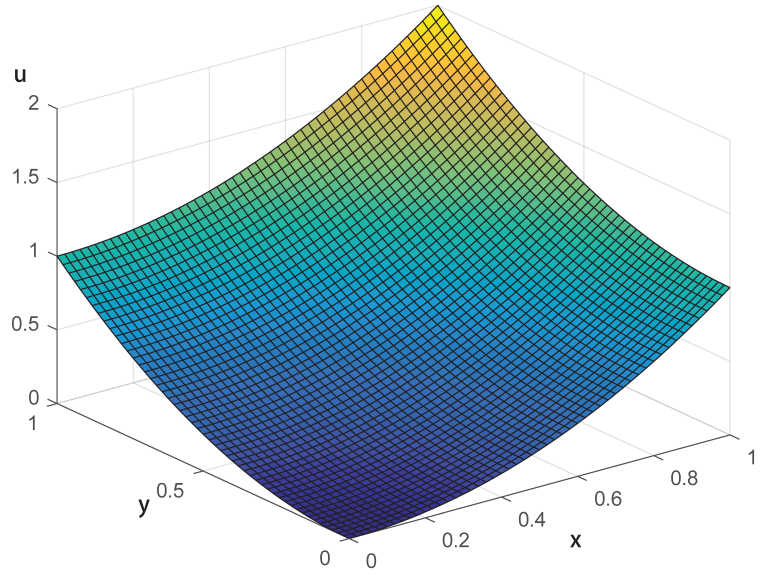


Figure 2.1: A simple example of 2D boundary value problem.

c) Boundary Conditions: The problem domain and the governing equation can not guarantee the uniqueness of the solution. Different kinds of boundary conditions can be applied for different physical constraints or assumptions, and all boundary conditions can be categorized into three kinds: the Dirichlet type, the Neumann type, and the mixed type, which will be introduced later. The boundary condition of the example is prescribed as: $u(x,0) = x^2$, $u(x,1) = x^2 + 1$, $u(0,y) = y^2$, $u(1,y) = y^2 + 1$.

It is worth to mention the difference between the initial value problem and the boundary value problem. For the initial value problem (IVP), the solution is provided at a time point $t = 0$, and the time marching scheme is required to calculate each time step. All the boundary conditions in BVP is spatial and usually do not include the time term. For a PDE with the time-varying term, it can be both BVP and IVP.

With all the three parts specified, the unique solution can be solved with analytical or numerical methods. For the above example, the boundary conditions are intentionally selected so the solution has an analytical expression: $u(x,y) = x^2 + y^2$. However, in the real world, considering the irregular geometries in the problem domain, different materials, and boundary conditions, most boundary value problems do not have an analytical solution. In this case, numerical methods are required to obtain an approximate solution over the problem domain.

The commonly used numerical methods to solve a BVP include the finite difference method (FDM), finite element method (FEM), boundary element method (BEM), finite volume method (FVM), etc. Since the Ampere's law and Poisson's equation are both elliptical problems, the FDM and FEM are the dominant methods used by engineers [31].

The finite difference method converts the partial differential equation to algebraic equa-

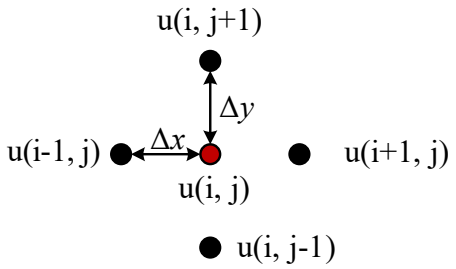


Figure 2.2: Rectangular grid for finite difference method.

tions by replacing the differential equation with difference equation using the Euler method or central difference scheme. For example, on the grid shown in Fig. 2.2, the second order partial differential term in (2.1) can be rewritten as [32]:

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(i-1,j) + u(i+1,j) - 2u(i,j)}{(\Delta x)^2}, \frac{\partial^2 u}{\partial y^2} = \frac{u(i,j-1) + u(i,j+1) - 2u(i,j)}{(\Delta y)^2}. \quad (2.2)$$

More details about the finite difference method will be described in Chapter 5.

For the finite element method, the strategy to convert the PDE to algebraic equations is different.

2.3 Galerkin's Finite Element Method

2.3.1 Ampere's Law for Eddy Current Problem

The Ampere's law with Maxwell's addition, which states that the magnetic field in space is generated by both electric current and displacement current, is the working principle of a transformer. A finite element transformer model, which provides more accurate and comprehensive information for simulation by considering the factors from a design perspective such as geometries, winding parameters, and material nonlinearity, is governed by the following partial differential equation [32]:

$$\nabla \times H = \nabla \times (\upsilon \nabla \times A) = J - \sigma \frac{\partial A}{\partial t} + \frac{\partial D}{\partial t}, \quad (2.3)$$

where A is the magnetic potential and the relation with magnetic flux density B satisfies $B = \nabla \times A$, υ is the nonlinear magnetic reluctivity, σ is the conductivity, and the right-hand terms are the impressed current, the eddy current, and the displacement current, respectively. In 2D cases, magnetic flux density B has B_x and B_y components while magnetic potential A only has A_z component.

Note that the displacement current term $\partial D/\partial t$ is usually not important so it can be ignored in low frequency applications. Since the curl of curl of A satisfies the following identity:

$$\nabla \times (\nabla \times A) = \nabla(\nabla \cdot A) - \nabla^2 A. \tag{2.4}$$

Applying the gauge $\nabla \cdot A = 0$, (2.3) can be rewritten as

$$\nabla \cdot (\upsilon \nabla A) = -J + \sigma \frac{\partial A}{\partial t}, \tag{2.5}$$

which is usually known as the eddy current problem and governs the magnetic field in electrical devices having windings such as transformer and generator.

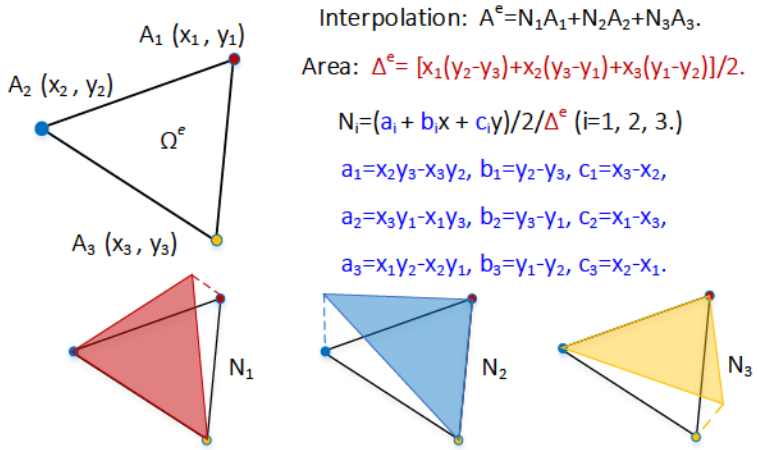


Figure 2.3: Triangular element and the interpolation (weight) function.

2.3.2 Domain Discretization and Interpolation

Given the problem domain, material properties υ and σ, external excitation J, and appropriate boundary conditions, the solution of magnetic potential A is unique and can be solved. However, it is impossible to solve the problem analytically and provide an exact expression for $A(x, y, t)$, which is why the finite element method is required to solve such a boundary value problem numerically.

Instead of obtaining the analytical expression A, the FEM discretizes the problem into numerous interconnected nodes and elements (e.g. Fig. 1.1) and seeks for a solution at each node or edge. Then the magnetic potential A at any location (x, y) can be obtained by interpolation.

For a 2D geometry, the triangular mesh is most commonly utilized because it is very flexible to represent different shapes using triangles of different sizes. In practice, tetrahedral elements and higher-order interpolation can be applied, and in this thesis, a 2D case with the triangular element and linear interpolation is used as an example to elaborate the parallel algorithms. Figure 2.3 shows a triangular element and the linear interpolation function. There are three unknowns at the vertexes and the inner distribution is presented using interpolation:

$$A^e = N_1 A_1 + N_2 A_2 + N_3 A_3. \tag{2.6}$$

The elemental equation, describing the relation of vertex unknowns A_1, A_2, and A_3, should be derived based on (2.5), which is the essential part of the finite element method.

2.3.3 Weighted Residual Method

Substitute the linear interpolation in Fig. 2.3 to (2.5), for each element Ω with index e, let r_e be the residual of (2.5):

$$r_e = \nabla \cdot (v^e \nabla A^e) + J^e - \sigma^e \frac{\partial A^e}{\partial t}. \tag{2.7}$$

Thus the following two statements are equivalent:

- 1. (2.5) is satisfied in the whole problem domain.
- 2. r_e is always 0 for each element Ω^e.

Since r_e is a distribution over each triangular element, to set r_e to 0, it will be more convenient to convert r_e to a single value via the integral over each element. For example, the above statements are also equivalent to the following statement:

- 3. The integral of r_e^2 over each triangular element Ω^e is always 0, namely,

$$\iint_{\Omega^e} r_e^2 dxdy = 0. \ (e = 1, 2, 3...) \tag{2.8}$$

There are many ways to manipulate the integral to make such an equivalent statement, of which the most popular one is the weighted residual method. Let w_e be any function defined on element Ω^e, the following statement is also an equivalence:

- 4. For any weight function w_e, the following integral is always 0 for each element Ω^e:

$$\iint_{\Omega^e} w_e \cdot r_e dxdy = 0. \ (e = 1, 2, 3...) \tag{2.9}$$

It is important to mention that the weight function w_e is arbitrary. Setting the weight function to the shape functions N_1, N_2, and N_3 in Fig. 2.3, respectively:

$$\iint_{\Omega^e} N_i \cdot r_e dxdy = 0, \ (i = 1, 2, 3; \ e = 1, 2, 3...) \tag{2.10}$$

3 equations with unknown A_1, A_2, and A_3 can be obtained for each element. The N_i, r_e, and A^e involved can be found in Fig. 2.3 and (2.7). By using the shape functions as the weight function, the above scheme is called the Galerkin's FEM [33].

The coefficients can be obtained by calculating the integrals, and the 3×3 elemental equations for Ω^e can be written as :

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} + \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} \begin{bmatrix} \frac{\partial A_1}{\partial t} \\ \frac{\partial A_2}{\partial t} \\ \frac{\partial A_3}{\partial t} \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} + \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}, \tag{2.11}$$

where A and F are 3×1 vectors represent the unknown magnetic potentials and external source at the vertexes of triangle, respectively; P is associated with the boundary conditions; M and T are 3×3 matrices whose entries are determined by the material properties and shape functions N_i. Within a specific element e they can be written as follows:

$$M_{ij}^e = -\upsilon^e \iint_{\Omega^e} \nabla N_i \cdot \nabla N_j dxdy, \tag{2.12}$$

$$T_{ij}^e = \sigma^e \iint_{\Omega^e} N_i \cdot N_j dxdy, \tag{2.13}$$

$$F_i^e = \iint_{\Omega^e} N_i J dxdy. \tag{2.14}$$

2.3.4 Assemble and Solve

The local 3×3 system of equations can't be solved directly, and the traditional way is to assemble all the elemental equations to a global system of equations. The assembling procedure essentially is a summation of local coefficients to the entries of the global matrix. For example, let G be the global symmetric matrix, if the three nodes indexed 1, 2, and 3 locally in element Ω^e are indexed I, J, and K globally, the following operations would be performed during the assembling phase:

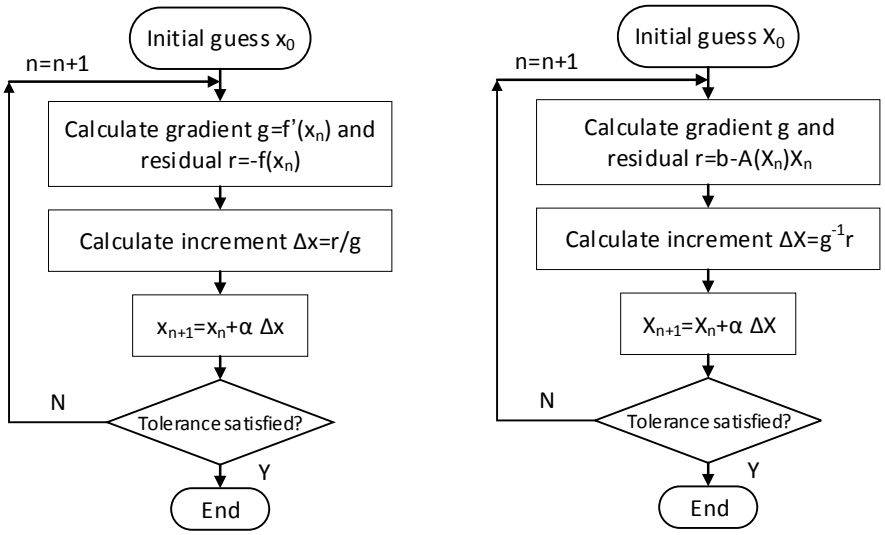
$$\begin{aligned} &G[I][I] = G[I][I] + M_{11}^e,\ G[J][J] = G[J][J] + M_{22}^e,\ G[K][K] = G[K][K] + M_{33}^e, \\ &G[I][J] = G[I][J] + M_{12}^e,\ G[J][K] = G[J][K] + M_{23}^e,\ G[K][I] = G[K][I] + M_{31}^e. \end{aligned} \tag{2.15}$$

Note that the entry $G[X][Y]$ of a global matrix is only non-zero when node X and node Y are connected directly and share one edge, implying the global matrix is sparse because each node is only related to its neighboring nodes.

The assembled global matrix is singular and needs constraints to have a unique solution. The boundary conditions will be applied here and then the system can be solved using a sparse matrix solver.

2.3.5 Newton-Raphson Method

The reluctivity υ^e in (2.12) usually depends on the unknown magnetic potential A for materials with nonlinear B-H curve such as iron, thus a system of nonlinear equations need to be solved after assembling.



(a) NR to solve f(x)=0. (a) NR to solve A(X) X=b.

Figure 2.4: Newton-Raphson scheme to solve nonlinear problem.

The Newton-Raphson scheme is commonly used to solve the nonlinearity. For a simple nonlinear equation $f(x)$ with only one unknown, the solution can be obtained by iterating the procedure in Fig. 2.4(a).

For a nonlinear system of equations, the procedure shown in Fig. 2.4(b) remains almost the same except the operations for matrix and vector. Note that X is a vector of unknowns $[x_1, x_2, x_3, ..., x_m]$; the gradient g, also known as the Jacobian matrix, is defined as

$$g = \begin{bmatrix} \frac{\partial A_1}{\partial x_1} & \cdots & \frac{\partial A_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial A_m}{\partial x_1} & \cdots & \frac{\partial A_m}{\partial x_n} \end{bmatrix}. \tag{2.16}$$

The Jacobian matrix is associated with X_n and changes at each iteration, implying the sparse matrix solver has to be repeatedly called during the Newton-Raphson iterations. Besides, the parameter α in Fig. 2.4 is the relaxation factor that can be adjusted to fix the divergence issue or to accelerate the convergence.

2.4 Parallel Computing

Traditional computer software is usually designed for serial computing, and the instructions are executed one by one sequentially on one central processing unit. Recently, computing hardware is integrating more and more processing units (PU) such as CPU cores and GPU CUDA cores, and it has motivated researchers to divide the problem into independent parts so that each processor can execute its part simultaneously.

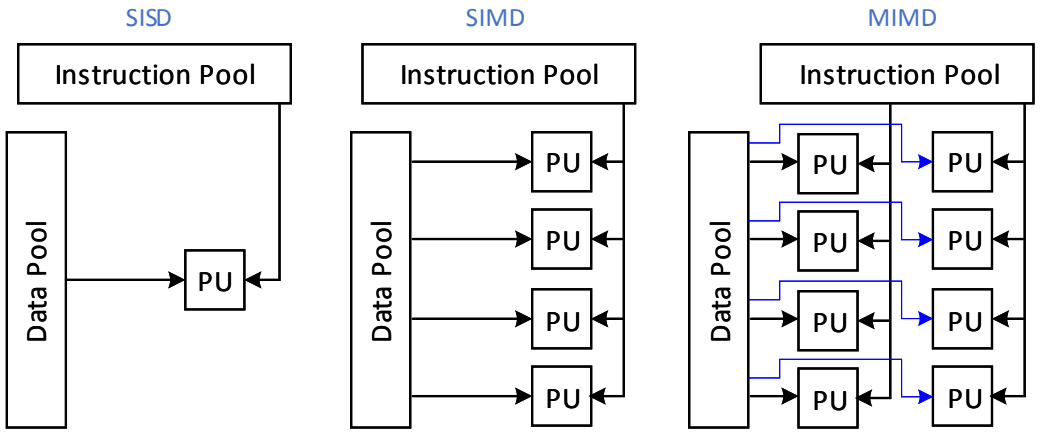


Figure 2.5: Flynn's taxonomy for parallel computing.

A program is composed of instructions and data, Flynn's taxonomy [34] is usually used to classify different patterns of parallel computing based on how instructions and data are fed to each processing unit. Figure 2.5 shows the commonly used classifications and some examples are provided below [35]:

- Single instruction single data (SISD) stream: single-core CPU.
- Single instruction multiple data (SIMD) stream: array processors, GPU.
- Multiple instruction single data (MISD) stream: mostly nonsense.
- Multiple instruction multiple data (MIMD) stream: Multiple-core CPU, computer clusters.

Also, based on the memory that each processing unit can access, parallel processing can be classified into shared-memory programming and distributed-memory programming. Although the processing units usually execute assigned tasks independently, the tasks need to exchange data or synchronize at some point. It is necessary to introduce the parallel software used to schedule the hardware usage and to control the program flow.

Multi-core central processing units (CPU) and many-core graphics processing units (GPU) are the essential components in the prevalent high-performance computing environments. And field-programmable gate arrays (FPGA) with a massive amount of look-up tables (LUT) and flip-flops (FF) are irreplaceable for hardware acceleration in areas like real-time computing, and more complex and large scale computing is being deployed on FPGA. To make use of the computing power of these resources, both parallel algorithms and supporting programming environments are required.

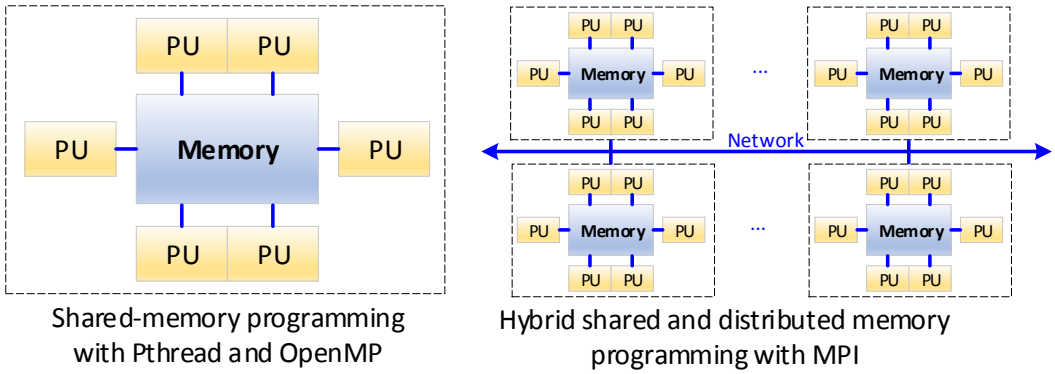


Figure 2.6: Shared and distributed memory structure for pthreads, OpenMP, and MPI.

2.4.1 Parallel Software for CPUs

Modern personal computers and workstations usually have tens of CPU cores to process tasks concurrently. POSIX Threads (also referred to as pthreads) and Open Multi-Processing (OpenMP) are the most commonly used application programming interfaces to perform parallel processing on one single computer node with multi-core CPU. Since all the CPU cores share the global memory on the motherboard, pthreads and OpenMP are shared-memory programming interfaces. They have such common features as supporting different programming language on different platforms, and also have some differences [35]:

Pthreads is very low-level and gives users extremely fine-grained control of each thread, and the thread function, thread communication, and synchronization can be implemented by users with much flexibility. On the contrary, OpenMP is high-level and easier to use because most low-level operations are transparent to users. One example is to unroll and parallelize the for-loop by simply adding some directives.

For computer clusters having numerous computer nodes that do not share memories, distributed memory programming is involved (Fig. 2.6). In this case, the message passing interface (MPI) can be used to implement data communication like broadcast, gather, reduction, etc. Although computer clusters can also integrate thousands of CPU cores, the data communication between distributed memory may cause serious overhead.

2.4.2 Parallel Software for GPUs

The Tesla V100 GPU released by Nvidia integrates 80 steaming multi-processors (SM), and each SM has 64 FP32 cores, 64 INT cores, and 32 FP64 cores, implying a total of 5120 cores for single-precision and integer operations and 2560 cores for double-precision operations simultaneously. Besides, Tesla V100 provides 16GB high bandwidth memory and NVLink^{TM} to achieve high-speed signaling interconnect between multiple GPUs.

Fig. 2.7 illustrates the inner view of a streaming multi-processors. Each SM has 128KB

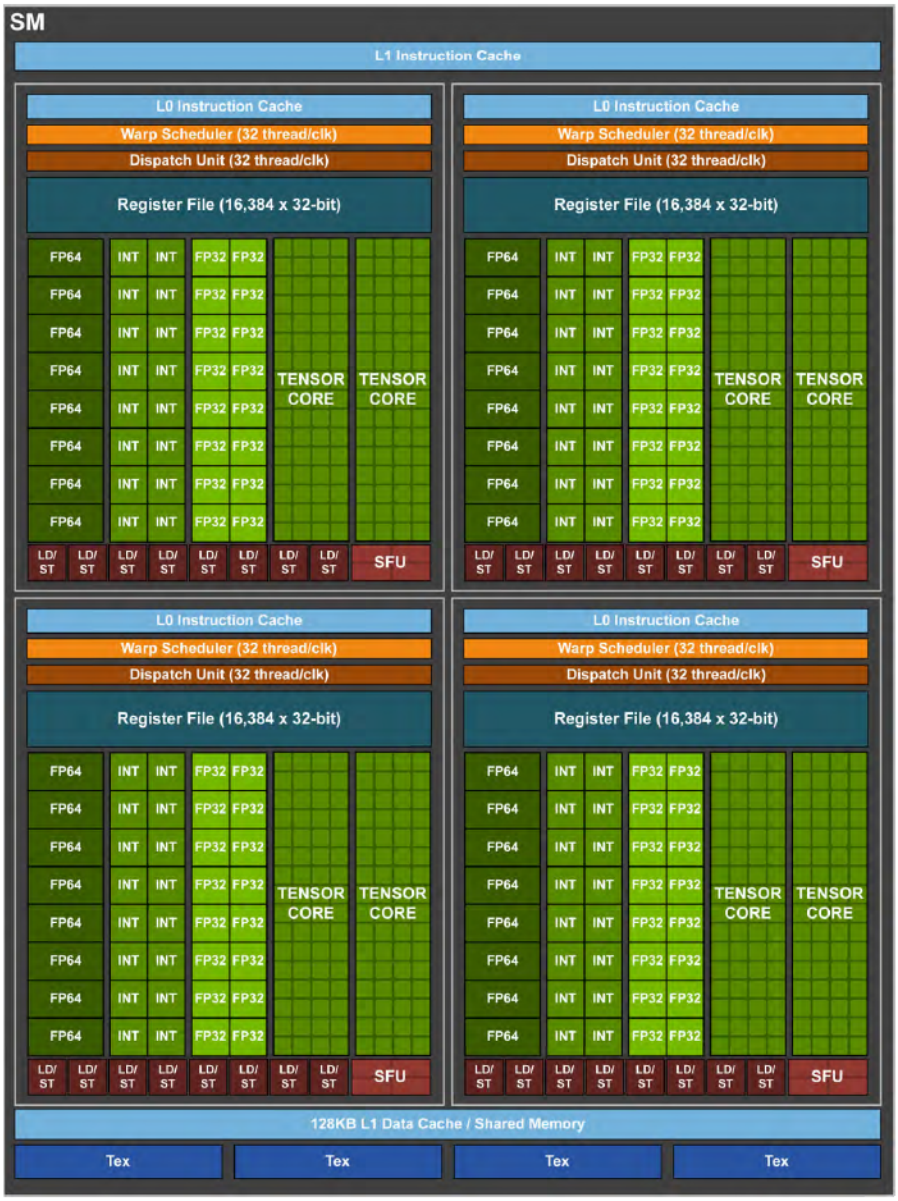


Figure 2.7: Details within one streaming multi-processors of Tesla V100.

shared memory for the 64 cores, which is much faster than accessing the global device memory. The compute unified device architecture (CUDA) is an application programming interface created by Nvidia and allows developers and engineers to use the massive GPU cores for general-purpose computing.

Note that GPU is a typical single instruction multiple data (SIMD) paradigm, and in CUDA codes, the single instruction is defined by the kernel. Once a kernel is launched, the same function will be applied to multiple input data sets, with each data set executed on different GPU cores. One simple example is the vector summation with adding operation for different input data, which is most suitable on GPU. Thus, to fully explore the comput-

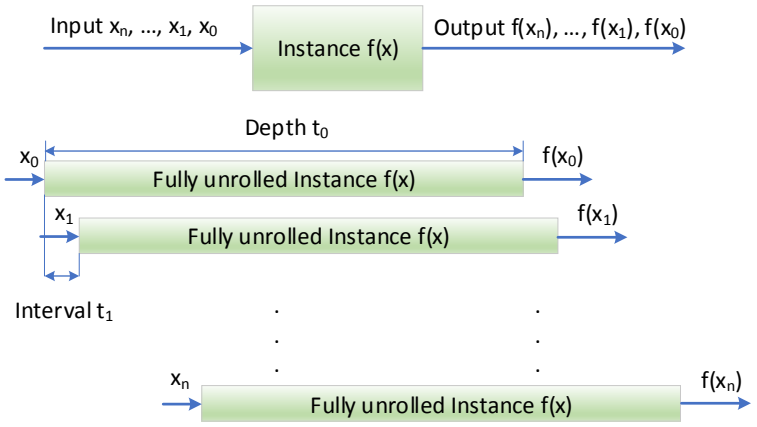


Figure 2.8: An example of data pipelining to achieve parallelism.

ing power of GPU, it is necessary to find a suitable algorithm that has a similar patter with the SIMD stream.

2.4.3 Parallel Computing for FPGAs

Recently, FPGA prototyping boards are also integrating more and more resources to provide high-performance computing solutions for engineering problems via hardware acceleration. FPGA is featured by hardware concurrency, and different hardware blocks can be configured to run different tasks concurrently. The very-high-speed-integrated-circuit hardware description language (VHDL) is used to implement different algorithms with the data-path and controller fashion. In addition to the hardware concurrency, data pipelining is another feature to achieve parallelism on FPGA, as shown in Fig. 2.8. Once an instant, usually composed of different phases, is entirely unrolled and implemented by hardware, the input data can be continuously fed without having to wait for the previous data processing to be finished.

Due to the concurrent nature of VHDL, which is quite different from other programming languages, it is usually a challenge to implement a very complicated large-scale algorithm using VHDL. In this case, the high-level synthesis (HLS) tool is very useful that it can convert the function in C language into an intellectual property (IP) core with different configurations such as unroll and pipelining using directives.

Generally, a C function will be compiled to an IP block with inputs, outputs, and some handshake signals for interfaces between blocks. The matrix-vector multiplication model is given as an example to elaborate on how to optimize the execution time using directives.

If calculating $b_i = \sum_1^N A_{ij}x_j$ is called a task, the computation of the matrix-vector multiplication is composed of many independent tasks. For the CPU codes, the summation loop is sequential in nature and the next task can not be started until the current task is ended. Whereas in FPGA implementation, the summation can be unrolled completely and

be compiled as an instance on hardware, and the multiplications $A_{ij}x_j$ can be executed in parallel on massive multiplying units. Although the subsequent summation of the products is the critical path and takes the major latency, the next task needs not to wait because the multiplying units are ready for the next computation. Thus, different tasks can be executed on the unrolled instance in a pipelined fashion.

The computation time of a pipelined IP block can be calculated as

$$t = t_{depth} + N \times t_{interval}, \tag{2.17}$$

where t_{depth} is the execution time of a single task, $t_{interval}$ is the time interval after which the next task begins, and N is the number of tasks.

The t_{depth} is usually determined by the algorithm and $t_{interval}$ can be optimized by array partitioning. In this example, the b vector can be partitioned by rows, and the Y matrix can be partitioned by its second dimension.

The pseudo-code for this matrix-vector multiplication block in Vivado HLS™ usually has the following form:

```
Function solve(b[N], x[N])
A[N][N]=[....];
#pragma array partition b dim 1
#pragma array partition Y dim 2
loop1 i in 1 to N
#pragma pipeline
  loop2 j in 1 to N
  #pragma unroll
  b[i]+=A[i][j]b[j];
  end loop2
end loop1
```

2.4.4 Hybrid Parallel Computing

Recently, hybrid parallel computing across different platforms is utilized to explore the merits of different computing architectures. For example, hybrid parallel CPU-GPU computing [36] is very suitable for problems including SIMD tasks and MIMD tasks: GPU can efficiently handle the vectorized computation while CPU can handle the communication between modules and the serial tasks.

Compared with CPU and GPU, FPGA has much smaller and deterministic latency, thus suitable for real-time hardware-in-loop application. However, when handling very large-scale computation, the lack of configurable hardware resources is the drawback. Nowa-

days, hybrid platforms such as the Intel Xeon-FPGA Hybrid Chip and CPU-FPGA [37] are also being developed for high performance computing.

2.5 Summary

In this chapter, the boundary value problem and the Galerkin finite element method are introduced, and the traditional procedures for finite element computation are presented. Besides, modern high-performance computing hardware and software are briefly introduced for CPU, GPU, and FPGA, respectively. The contents in this chapter are important background for the entire thesis.

3

Node-Wise Parallelism: Nodal Domain Decomposition

3.1 Introduction

This chapter proposes a novel finite element procedure that explores parallelism at each finite element node to accommodate the SIMD hardware and the scheme is applied to both linear and nonlinear problems. The finite element method (FEM) has been dominantly utilized for the modeling and design of electromagnetic apparatus such as rotating machines and transformers due to its accuracy and flexibility for complex geometries. The notorious computational burden, stemming from repeatedly factorizing the updated Jacobian matrix, demands exploration of efficient nonlinear FE solvers for conveniently designing and testing electromagnetic apparatus.

The prevalent trend in high-performance computing (HPC) resources is to increase the number of cores massively either as clusters of multi-core central processing units (CPUs) or many-core graphics processing units (GPUs); for example, the recently released NVIDIA® Tesla V100 accelerator card is equipped with 5120 Cuda cores and 16GB of HBM2 memory [12], and it has motivated engineers to explore suitable parallel algorithms to make full use of its computing power. In the past, GPU-accelerated conjugate gradient solver [20], [38]–[40] and domain decomposition (DD) method on CPUs [41], [42] have been implemented to improve the efficiency of the electromagnetic field computation. However, massive parallelism could be hardly achieved since the *assemble and then solve* procedure in traditional FE technique is essentially a centralized way of thinking. On the other hand, to fully explore the computational power of GPUs, a decentralized thinking pattern resulting in massive parallelism would be greatly promising. For example, the element-wise FE technique was proposed for matrix-vector multiplication with inde-

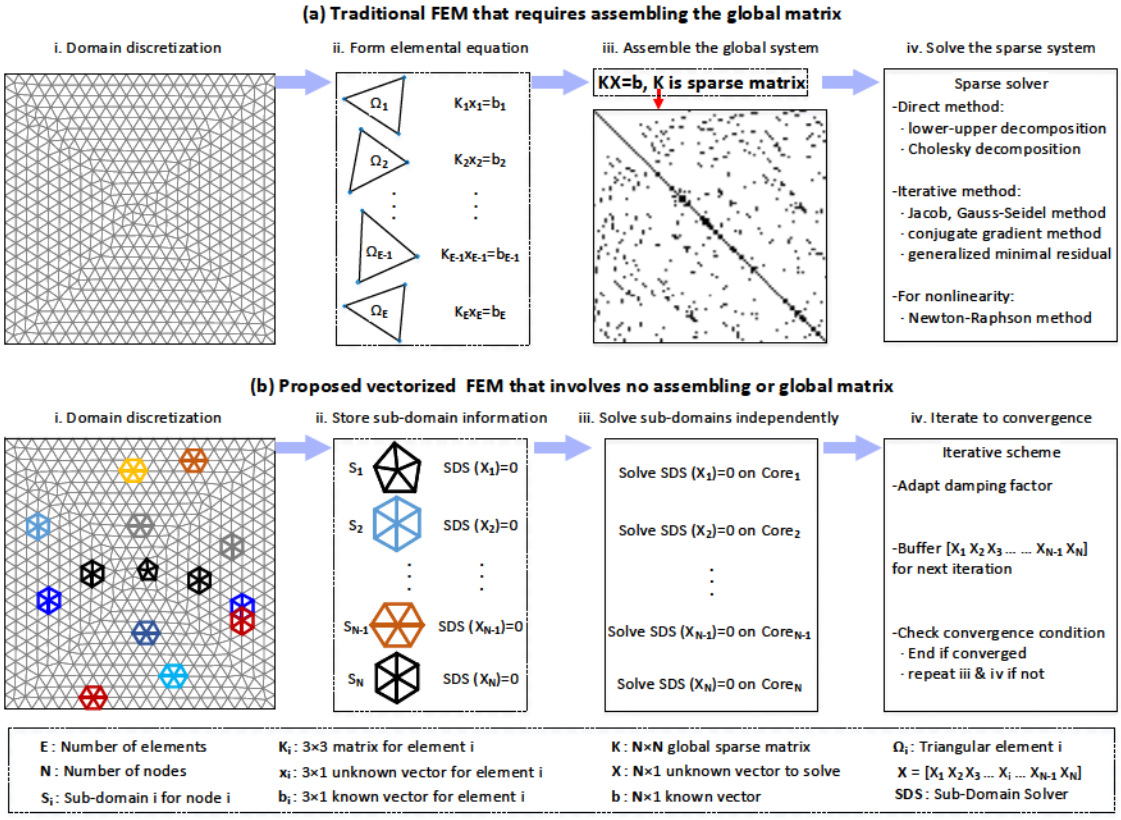


Figure 3.1: Illustration of traditional FEM and the proposed nodal domain decomposition (NDD) based FEM.

pendent computation at the triangular element level [27], [43], and was then developed to solve a linear FE problem on GPU architecture [23], [24]. Besides, the Jacobi-based GPU solver has also been explored for the computation of Poisson's equation [44]. Although deficient to handle nonlinear FE problems, massive parallelism and a decent speedup are achieved on GPUs due to the divide-and-conquer strategy. Massively parallel simulations have also already been investigated for large-scale power system dynamic and transient simulation applications [45], [46].

In this chapter, the nodal domain decomposition (NDD) with relaxation is proposed to explore the massive parallelism in FE computation of both linear and nonlinear problems. This novel idea is inspired by the following question regarding the extreme partition of a domain: *What if a sub-domain contains only one unknown node?*

As illustrated in Fig. 3.1, the proposed NDD has the following features compared with the traditional nonlinear FE solver based on the Newton-Raphson (NR) algorithm:

1. There exists only one unknown in each sub-domain and thus no matrices are necessary, i.e., the NDD is matrix-free and memory-friendly.

2. Instead of applying the NR algorithm to a global system, which is sequential in na-

ture, the problem is solved utilizing a neat relaxation scheme by iteratively updating each node (sub-domain) in a massively parallel manner.

3. Each sub-domain is solved independently following the same pattern and the computation is concise since there is only one unknown. Therefore, the NDD shows perfect modularity for Kernel programming on GPU architectures.

4. A specific data structure is required for the single instruction multiple data (SIMD) programming on GPU.

The matrix-free feature stems from the fact that the more number of sub-domains, the fewer the number of unknowns required to be solved for each sub-domain. It is also worth mentioning that for a linear problem, both the NDD scheme and the N-scheme in [10] using weighted summation of elemental contributions are equivalent to the Jacobi iterative scheme. However, the proposed NDD scheme can be easily extended for nonlinear problems; in this sense, the NDD scheme better reveals the domain decomposition advantage regardless of the linearity or nonlinearity of the problem. Also, the massively parallel computing resource (either large-scale CPU or GPU clusters) that matches the FE problem size is a prerequisite to explore the full potential of the NDD scheme, especially for large scale FE problems.

This chapter is organized as follows: Section 3.2 briefly introduces the FE equations of general EM apparatus with the nonlinear B-H curve. Then Section 3.3 presents the methodology of the NDD and the supporting data structure. Section 3.4 briefly introduces the mixed boundary conditions exchange between sub-domains to accelerate the convergence. In Section 3.5, case studies employing the NDD scheme are implemented on both CPUs and GPUs, and the results are compared with the commercial FE package Comsol Multiphysics™ concerning the accuracy and speedup. Finally, Section 3.6 gives the summary.

3.2 Finite Element Equations of EM Apparatus

Governed by the Ampere's law, a 2-D magnetostatic problem can be described by the following equation [47]:

$$\nabla \cdot (\upsilon \nabla A) = -J, \tag{3.1}$$

where A is the z-component of the magnetic vector potential to be solved, υ is the material reluctivity, and J is the z-component of the impressed current density.

The problem can be solved by the well-known Galerkin FEM [48], and the general steps are discretizing the domain, forming elemental equations, assembling, applying boundary conditions and solving. The product of the weight function and the residual is integrated over each triangular element (Fig. 3.2) and the following elemental equations can be obtained by forcing the integral to be zero:

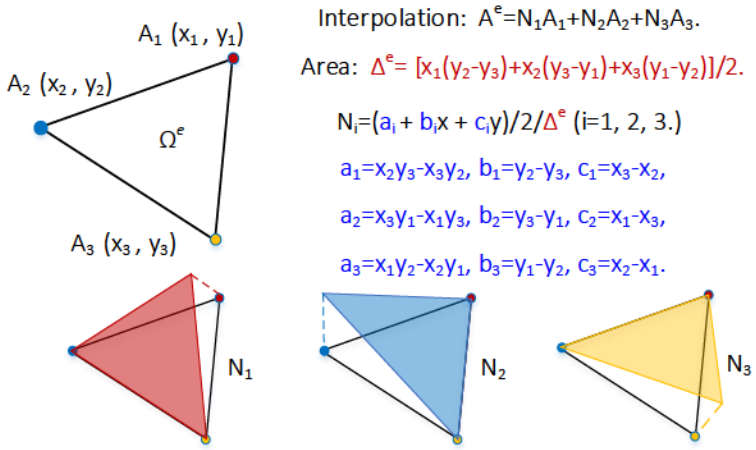


Figure 3.2: Triangular element and the interpolation (weight) function for the Galerkin FEM.

$$\frac{\upsilon^e}{4\Delta^e}\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}\begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} = \frac{J^e\Delta^e}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \tag{3.2}$$

where $k_{11} = b_1b_1 + c_1c_1$, $k_{12} = k_{21} = b_1b_2 + c_1c_2$,
$k_{22} = b_2b_2 + c_2c_2$, $k_{23} = k_{32} = b_2b_3 + c_2c_3$,
$k_{33} = b_3b_3 + c_3c_3$, $k_{31} = k_{13} = b_1b_3 + c_1c_3$.

For ferromagnetic triangular elements, the reluctivity υ^e depends on the strength of the unknown magnetic vector potential, implying that the problem is nonlinear. For the nonlinear solution, the calculation of Jacobian matrix for (3.2) is required using the NR scheme. To obtain $\frac{\partial\upsilon^e}{\partial A_i}$, the following chain rule in partial differentiation is usually utilized [9]:

$$\frac{\partial\upsilon^e}{\partial A_i} = \frac{\partial\upsilon^e}{\partial B^2}\frac{\partial B^2}{\partial A_i}(i = 1, 2, 3). \tag{3.3}$$

The $\partial\upsilon^e/\partial B^2$ can be obtained from the nonlinear B-H representation while $\partial B^2/\partial A_i$ is derived using the definition of the magnetic flux density: $B = \nabla \times A$.

For the traditional nonlinear FE solver, all the elemental equations are assembled to form a global sparse system and then solved with some iterative schemes such as the NR technique. Therefore, large matrices and efficient sparse solvers are always inevitable. In the proposed NDD scheme, the nonlinear system is solved in a decentralized manner.

3.3 Nodal Domain Decomposition with Relaxation

The traditional Schwartz domain decomposition [49] divides the whole domain into several sub-domains and each sub-domain can be solved independently. The information exchange between sub-domains is implemented by subtly manipulating the boundary con-

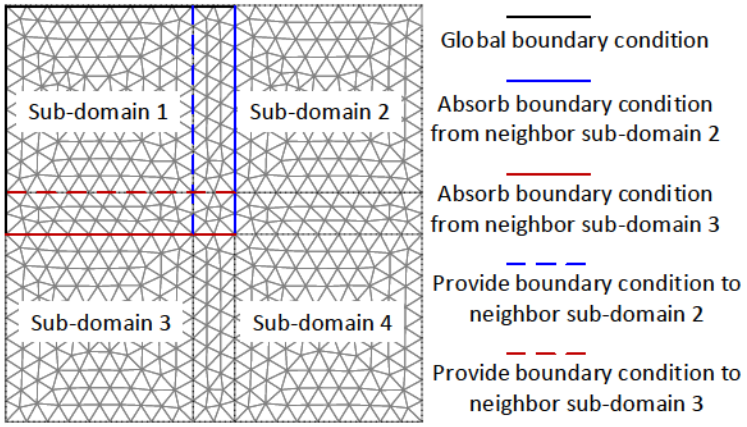


Figure 3.3: Information exchange in a sub-domain for overlapping Schwartz domain decomposition.

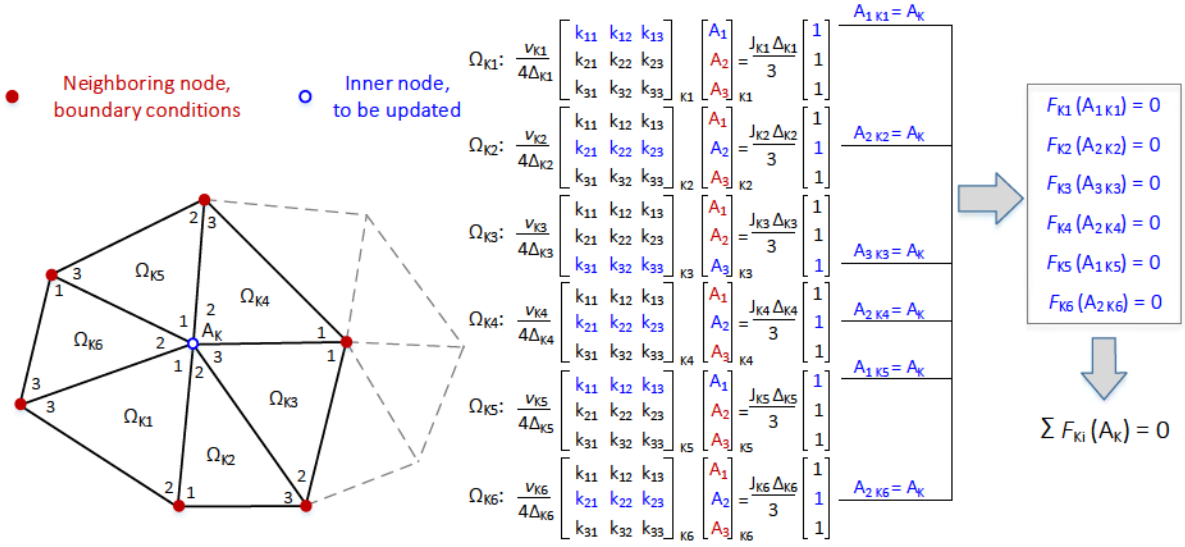


Figure 3.4: A sub-domain in NDD and its solution.

ditions (Fig. 3.3), and an iterative scheme is required to achieve the steady-state for final solutions since the values of the inner boundaries are unknown and are usually assigned guessed values.

As illustrated in Fig. 3.3, in each sub-domain, the inner nodes are solved based on the global boundary nodes and the neighboring boundary nodes from other sub-domain, and they also serve as the boundary nodes for other sub-domain. Although this neighboring information is not always correct, if all sub-domains work together repeatedly at the same time, the information exchange becomes effective to converge to the final solution. In this sense, the domain decomposition method is a *relaxation* scheme.

Note that the term *relaxation* describes the iterative nature of the solution process, namely, the independent sub-domain solver and the repeated data communication be-

tween sub-domains. Despite the necessity of iteration, the reduced problem size and the parallelism have made the domain decomposition method beneficial for parallel computing architectures such as multi-core CPUs. The number of nodes in each sub-domain, which determines the matrix size of the sub-problem, depends on the domain partition. Commonly, each sub-domain can still contain more than hundreds of nodes for a medium size FE problem [41], [42], and only partial parallelism can be achieved. However, from the perspective of a Cuda core on a GPU, which is suitable for massive parallelism, an ideal sub-domain should contain as few nodes as possible.

With the relaxation scheme, an extreme situation where each sub-domain contains only one inner node is considered. As shown in Fig. 3.1(c) and Fig. 3.4, each non-boundary node and its direct neighbors make up a sub-domain. Applying the neighbors' values from the previous iteration as boundary conditions, the value of the inner node can be updated, which is a miniature FE problem. Due to the overlapping, each node is updated based on its direct neighbors, and meanwhile serves as boundary conditions when its neighbors are updated. Thus, each inner node is updated independently and the steady-state can be reached in an iterative manner, which is perfect for massively parallel architectures.

At first glance, the miniature FE problem in Fig. 3.4 seems to be a nonlinear system with 7 unknowns, but a closer inspection reveals that it is a 1×1 nonlinear system because the other equations are overwritten by the imposed boundary conditions.

As illustrated in Fig. 3.4, all the neighboring elements Ω_{Ki} (i=1, 2, 3...) contribute to the solution of the inner node A_K, whereas not all the elemental equations are useful depending on the element-node numbering scheme. For example, for element Ω_{K1}, the node A_K to be solved is numbered 1, and A_2 and A_3 are given as boundary conditions, thus only the first elemental equation (highlighted in blue) is valid for the solution of A_K since the other two equations will be overwritten. Similar things occur in the other neighboring elements; therefore, the following equations considering all the neighbouring elements need to be solved for A_K:

$$\sum_{i=1}^{N} F_{Ki}(A_K) = 0, \tag{3.4}$$

where K is the index of the node to be solved, N is the total number of neighboring elements of node K, Ki is the element index of its i^{th} neighboring element, and F_{Ki} is one of the elemental equations of element Ki determined by the element-node numbering scheme.

To solve the 1×1 nonlinear equation (3.4), the Newton iteration is applied and the increment ΔA_K can be calculated by:

$$\Delta A_K = \frac{-\sum_{i=1}^{N} F_{Ki}(A_K)}{\sum_{i=1}^{N} \frac{\partial F_{Ki}(A_K)}{\partial A_K}}. \tag{3.5}$$

Note that the updating scheme in (3.5) works for all cases where the neighboring elements

```
typedef struct
{
	int Id;
	double Cordinate_X;
	double Cordinate_Y;
	int Number_of_Neighbour_Element;
	int Neighbour_Element_Id[8];
	int Neighbour_Element_Number[8];//1,2,3
	int Node_Type;//Boundary node indicator
	double Anew;//Current iteration
	double Aold;//Previous iteration
}NDDR_FEMNode;
typedef struct
{
	int Id;
	int I,J,K;//Element-node numbering
	double Element_Area;
	int Element_Type;//Air or Ferromagnetic
	double Ve;//Reluctivity
	double Js;//Impressed current density
	double k11,k12,k13,k22,k23,k33;//Matrix
}NDDR_FEMElem;
```

Figure 3.5: Data structure related to the FE nodes and elements for the NDD scheme.

are all linear elements, all nonlinear elements or a mixture of linear and nonlinear elements.

Since all the calculations are executed at the nodal level and no matrices are involved, a matched data structure is required for efficient data access. Based on the solution process presented in Fig. 3.4, the *structs* defined in C language are presented in Fig. 3.5. Each node or element is an entity with some attributes. Thus, the memory required for the NDD scheme increases linearly with the problem size, and all computations can be completed only with two arrays of the defined structs. Note that the maximum number of neighbors was set to 8 based on the inspection that for general 2-D triangular mesh, each node has no more than 8 neighboring triangular elements. For some extreme 2-D mesh or 3-D mesh, the limit can be adjusted accordingly.

3.4 Mixed-Type Boundary Condition for Sub-domain Solver

When A_K evolves, not only the values of its neighbors can be utilized, but also the normal component on the outer edge, representing the trend how the nodal value changes along the normal direction. A mixture of the nodal values and the normal component (third-type boundary conditions shown in Fig. 3.6) will efficiently accelerate the converging process of the evolution.

When a mixed boundary condition is applied, the sub-domain solver should also be

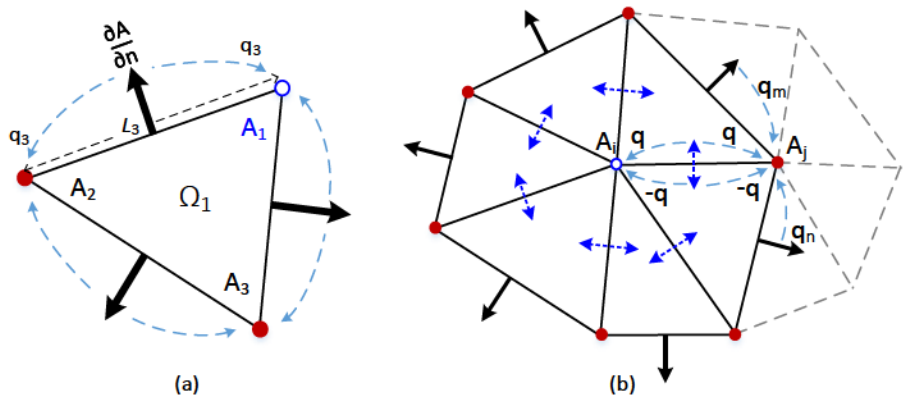


Figure 3.6: Illustration of the outward normal component (mixed boundary condition) .

adjusted. Figure 3.7 shows the details of a sub-domain solver (SDS) when mixed boundary condition

$$\frac{\partial A}{\partial n} + \gamma A = h \tag{3.6}$$

is applied. Note that A is the neighbors' nodal value from the previous iteration while $\frac{\partial A}{\partial n}$ is the flux outward the edge, which is also evaluated from the previous iteration using the adjacent sub-domain, as shown in Fig. 3.6(b). Since the central node is related to all the triangular elements that share it, the elemental equations are summed altogether to solve. Besides, not all the equations have the same weights. The equations where the central node dominates (highlighted in blue) weights 1 while the equations where the boundary nodes dominate (highlighted in red) weights $\frac{1}{\gamma}$. Similarly, a 1×1 equation is obtained to solve the central node. For a linear problem, the solution is merely a weighted summation of its neighbors' previous values while for nonlinear problem, several (typically 2-5) Newton iterations are required.

When updating the central node of a sub-domain, instead of merely using the values of its neighbors, using mixed boundary condition representing the trend how these values change along the normal direction can greatly enhance the information exchange between sub-domains and thus accelerate the convergence. Figure 3.8 shows the comparison of the converging curve after applying mixed boundary condition, implying the number of iterations required reduced from 800 to 200 roughly.

Note that the sub-domain solver with mixed boundary condition works for both linear and nonlinear triangular elements.

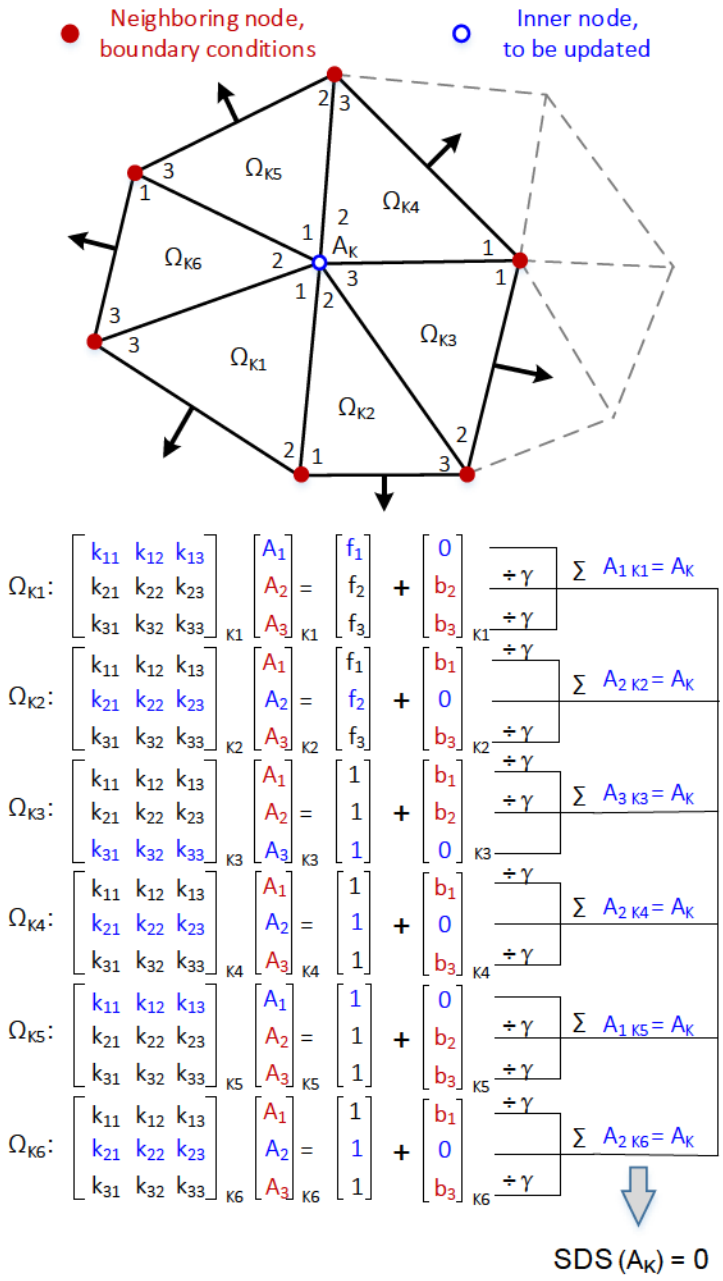


Figure 3.7: Adjusted sub-domain solver when mixed boundary condition is applied.

3.5 Case Studies

3.5.1 Finite element model of E-core transformer

The 2D E-core transformer model in Fig. 6.1 is studied. The transformer size is 5.2m×3.6m, the width of the yoke and the limb is 0.5m, the coil size is 0.25m×2m, and the coil turns are 390 for the primary side and 810 for the secondary. The conductivity of the transformer core is 10^6 S/m, and the time-varying winding currents are $I_p = 5000\sin(120\pi k\Delta t)$

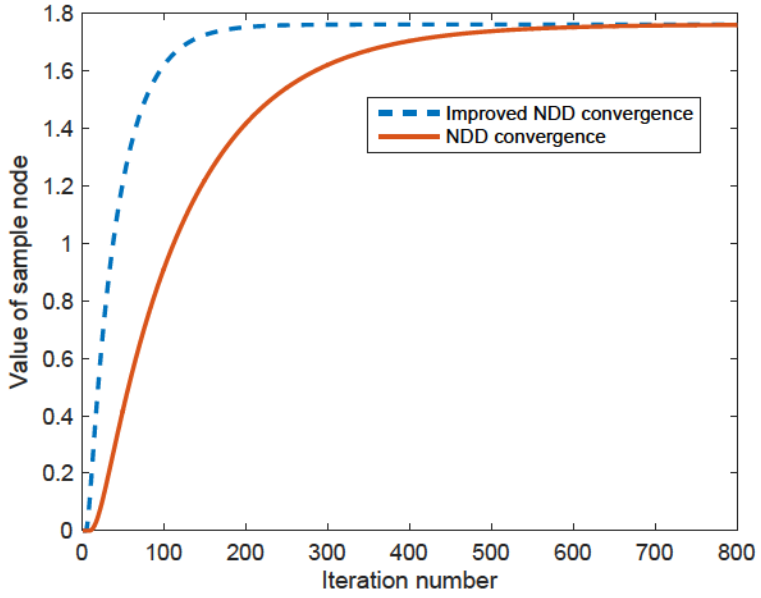


Figure 3.8: Converging curve of the NDD scheme with and without mixed boundary condition.

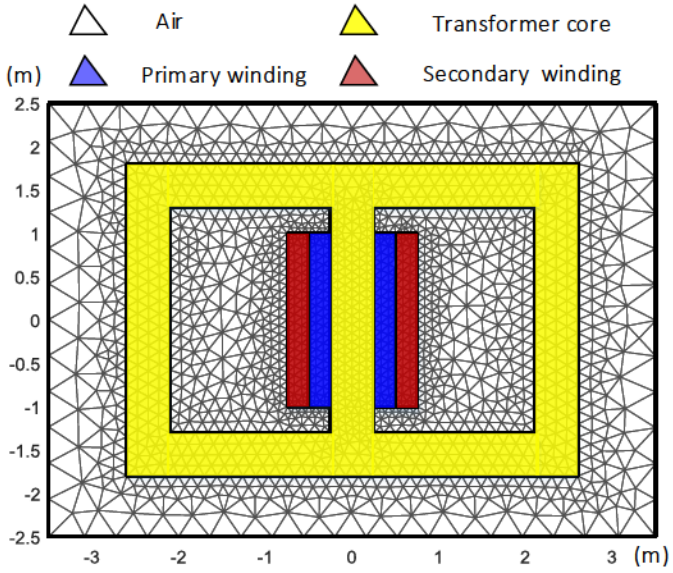


Figure 3.9: FE model of an E-core transformer for the case studies.

A, $I_s = 2000\sin(120\pi k\Delta t)$ A (k=0, 1, 2...). The transformer core material is Electrical Steel 35ZH135, and the nonlinear B-H curve can be found in [50]. Both the primary winding and the secondary winding are fed with sinusoidal current sources and the produced magnetic vector potential can be calculated with the NDD scheme. The flow chart of the NDD scheme is presented in Fig. 3.10(a).

The same FE problem was solved with the commercial software package Comsol. The domain decomposition solver settings are presented as follows: the number of available

Table 3.1: Relative tolerance ϵ_0, iteration number N, and $Error$ of the NDD scheme for different problem sizes

Cases	Number of nodes	Number of elements	Relative tolerance ϵ_0, iteration number N, and $Error$					
Case 1	528	1000	ϵ_0	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
			N	89	161	230	327	443
			$Error$	6.1%	0.55%	0.085%	0.011%	0.001%
Case 2	1273	2482	ϵ_0	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
			N	196	378	537	1104	1445
			$Error$	9.3%	0.82%	0.15%	0.038%	0.003%
Case 3	3303	6516	ϵ_0	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
			N	331	731	1456	2047	2865
			$Error$	20.5%	3.71%	0.45%	0.083%	0.013%
Case 4	4923	9682	ϵ_0	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
			N	394	1032	1663	2879	3979
			$Error$	33.7%	4.57%	0.78%	0.15%	0.065%
Case 5	10104	19956	ϵ_0	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
			N	565	1683	3067	4252	6214
			$Error$	51.7%	7.69%	0.85%	0.17%	0.085%

cores for parallel processing is set to 40; the additive Schwarz scheme is applied with 40 sub-domains and the direct linear solver is used; the nonlinear method is set to the automatic Newton method, and the termination technique utilizes a tolerance factor of 0.001 or a maximum iteration number of 25. Other settings such as damping factor and coarse preconditioning remain default. The results are regarded reliable and the efficiency of its optimized nonlinear domain decomposition solver is assumed to be state-of-the-art. Thus, the results obtained from Comsol^{TM} serve as the benchmark to evaluate the accuracy and efficiency of the proposed NDD scheme.

3.5.2 Implementation, accuracy and efficiency of magnetostatic case

To evaluate the accuracy and efficiency of a single finite element computation, a magnetostatic case is studied, where the winding currents are set to the peak values (I_p=5000A, I_s=2000A).

As an iterative relaxation scheme, the accuracy of the NDD is determined by the convergence criteria, i.e., the relative tolerance ϵ_0 between two successive iterations. The change of the problem size is also considered. The problems are solved with both the NDD scheme and Comsol^{TM} using the same mesh, respectively, and Table 3.1 provides the prescribed ϵ_0, the iteration number N required, and the relative error $Error$ of the two methods for five different problem sizes.

It can be concluded from Table I that the NDD scheme converges exactly to the same solution of Comsol^{TM} if the iteration number keeps increasing. In engineering problems, since an error of less than 1% is usually acceptable, it is safe to set the relative tolerance ϵ_0 of the NDD to 10^{-5} in all the cases in Table 3.1.

The field distributions of the magnetic vector potential and the magnetic flux density

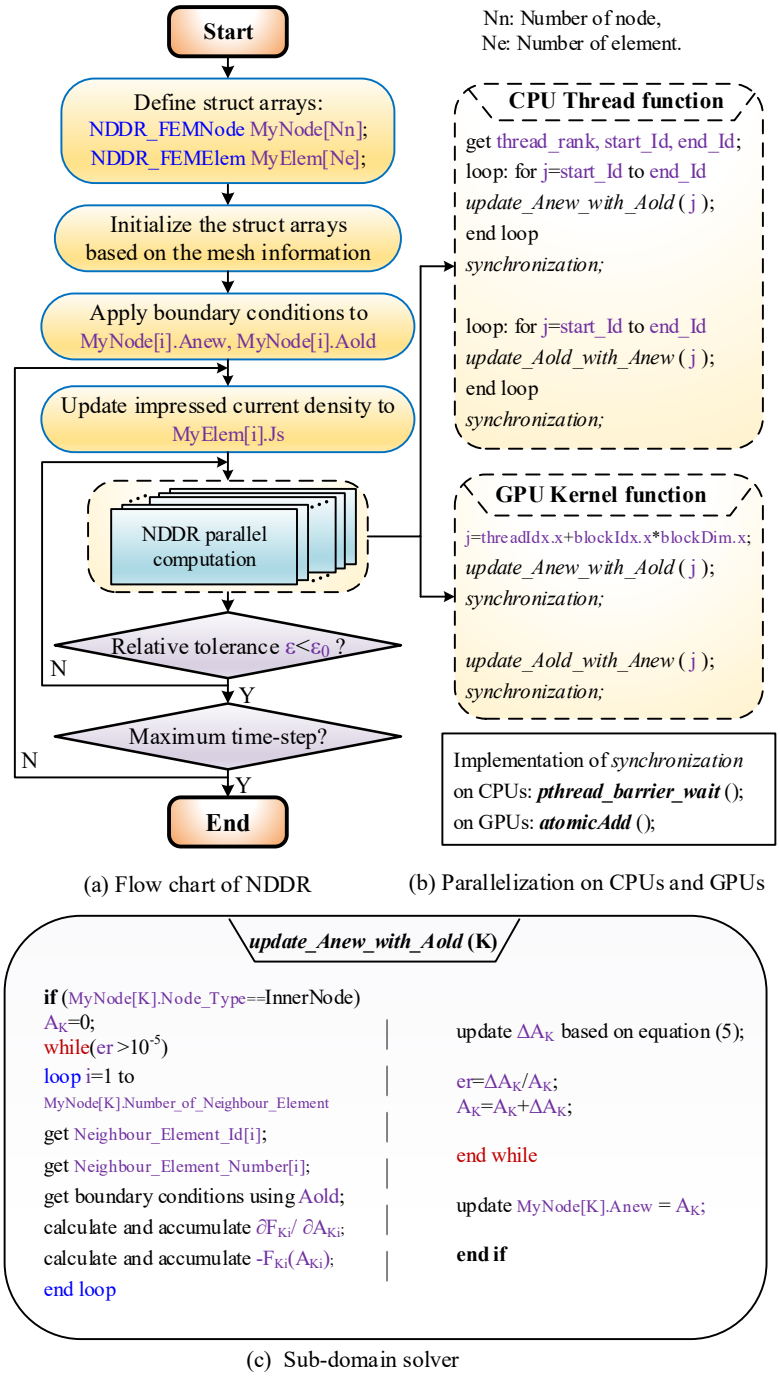


Figure 3.10: Detailed implementation of the NDD scheme on CPUs and GPU.

obtained from the NDD scheme in Case 2 with $\epsilon_0 = 10^{-5}$ are plotted in Fig. 3.11 with a relative error of 0.15% compared with Comsol^{TM}.

As mentioned before, the NDD scheme is perfectly suited for massively parallel architectures since each sub-domain can be solved independently within each iteration. The NDD scheme is implemented on a parallel workstation with multi-core CPUs and many-

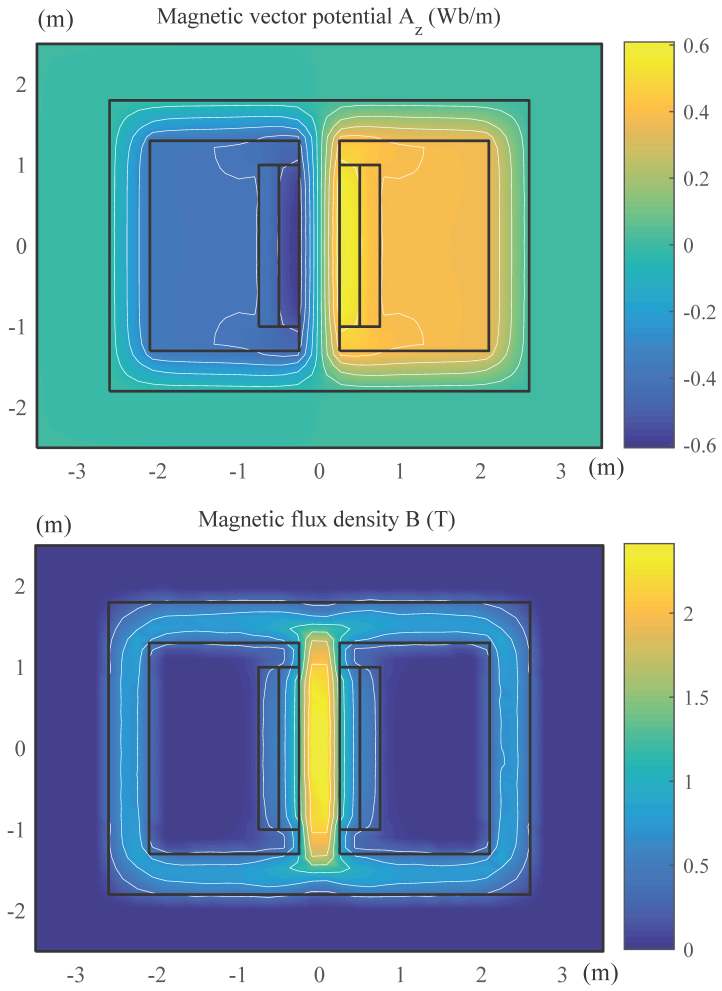


Figure 3.11: Field distribution of the NDD scheme in Case 2 with $\epsilon_0 = 10^{-5}$.

core GPU. Specifically, the workstation has dual Intel Xeon E5-2698 v4 CPUs, 20 cores each, 2.2GHz clock frequency, and 128GB RAM. The GPU is the NVIDIA Tesla V100-PCIE-16GB with 5120 Cuda cores, and details can be found in [12]. Figure 3.10(b) provides the parallel implementation concerning POSIX Threads on CPUs and Kernel on GPUs, and Fig. 3.10(c) shows the details of the sub-domain solver. For the implementation on the 40 CPU cores, each core still needs to handle hundreds of sub-domains due to the limited number of cores. Whereas, for the implementation on the GPU, each Cuda core can handle much fewer sub-domains. In fact, in Case 1-4, each Cuda core only needs to handle one single sub-domain since the number of nodes is less than the number of Cuda cores.

It is very important to mention that the run-time of Comsol^{TM} is measured using the Comsol-Matlab-Livelink^{TM}, meaning the solution phase can be timed accurately with little overhead.

The massive parallelism of the NDD scheme also results in decent computational efficiency. With the prescribed $\epsilon_0 = 10^{-5}$, the execution time and speedups of the parallel

Table 3.2: NDD execution time and speedup for CPU and GPU with $\epsilon_0 = 10^{-5}$

Cases	Comsol™ Execution Time (s)	NDD CPU Parallelization						NDD GPU Parallelization	
		Execution Time (s)-Thread_N					Speedup	Execution Time (s)	Speedup
		1	4	8	20	40			
Case 1	2.1	0.35	0.16	0.10	0.088	0.25	23.9	0.045	47
Case 2	3.4	1.92	0.87	0.54	0.37	0.57	9.2	0.107	32
Case 3	9.0	13.90	5.24	3.23	1.64	1.26	7.1	0.29	31
Case 4	10.3	21.61	8.89	4.88	2.22	1.67	6.2	0.32	32
Case 5	18.0	85.24	31.34	18.55	7.71	5.31	3.4	0.76	24

NDD scheme implemented on CPUs and GPU are provided in Table 3.2. It can be inferred that the execution time of the optimized Comsol™ solver increases almost linearly with the problem size (node number), and it is revealed from Table 3.1 that the iteration number required for NDD also increases linearly with the problem size (number of nodes).

In Case 1-4, maximum parallelism is achieved, i.e., enough hardware resources are available so that each sub-domain is projected to one single hardware core. Therefore, the execution time is only determined by the iteration number and thus also increases linearly with the node number. Compared with Comsol™, a steady speedup of more than 30 times is achieved on the GPU implementation. However, in Case 5, the speedup drops because each Cuda core has to handle two sub-domains instead of only one. Similarly, for CPU implementation, the drop of the speedup when node number increases can be also explained. With multiple GPUs in the future with more than 10,000 cores, the speedup of Case 5 with maximum parallelism can also reach more than 30 times without doubt.

3.5.3 NDD for magnetodynamic case

For the magnetodynamic case, the governing equation will include the eddy current term:

$$\nabla \cdot (\upsilon \nabla A) = \sigma \frac{\partial A}{\partial t} - J. \tag{3.7}$$

Similarly, applying the Galerkin FEM will result in the following elemental equations:

$$\begin{aligned} \frac{\upsilon^e}{4\Delta^e} \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} + \frac{\sigma^e \Delta^e}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} \frac{\partial A_1}{\partial t} \\ \frac{\partial A_2}{\partial t} \\ \frac{\partial A_3}{\partial t} \end{bmatrix} \\ = \frac{J^e \Delta^e}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \end{aligned} \tag{3.8}$$

The following algebraic equations can be obtained after time discretization with Back-

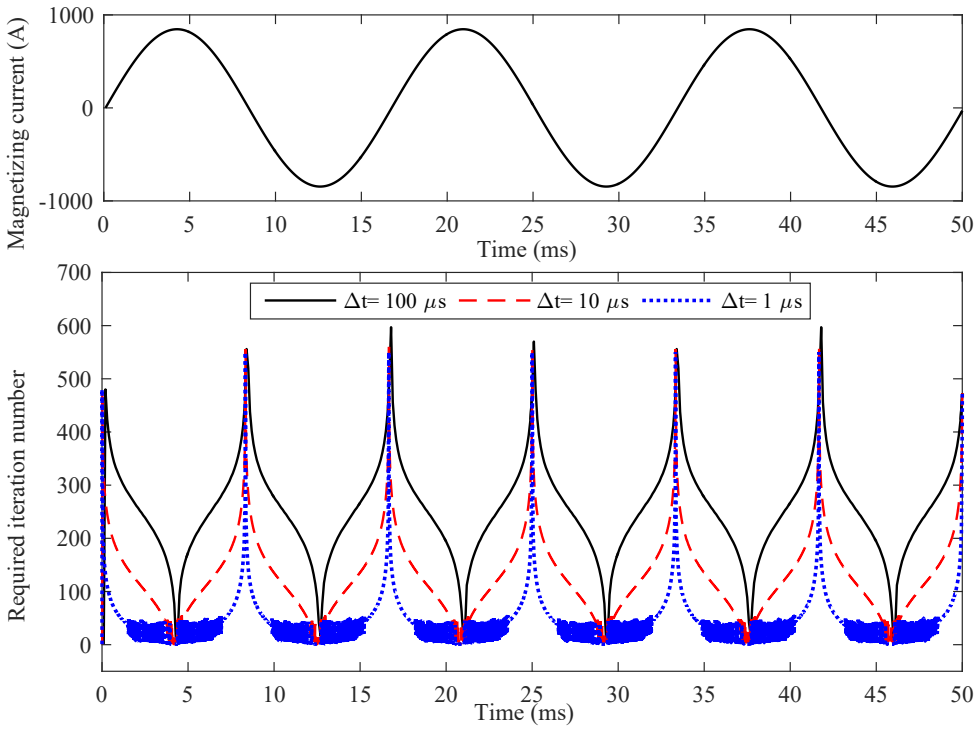


Figure 3.12: Number of iterations required for different time-steps to maintain a relative tolerance of 10^{-5} for the NDD in Case 2.

ward Euler method:

$$\begin{aligned}
&\frac{\upsilon^e}{4\Delta^e}\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}\begin{bmatrix} A_1(t+\Delta t) \\ A_2(t+\Delta t) \\ A_3(t+\Delta t) \end{bmatrix} \\
&+\frac{\sigma^e\Delta^e}{12\Delta t}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}\begin{bmatrix} A_1(t+\Delta t) \\ A_2(t+\Delta t) \\ A_3(t+\Delta t) \end{bmatrix} \\
&=\frac{J^e(t+\Delta t)\Delta^e}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}+\frac{\sigma^e\Delta^e}{12\Delta t}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}\begin{bmatrix} A_1(t) \\ A_2(t) \\ A_3(t) \end{bmatrix}.
\end{aligned} \tag{3.9}$$

Thus, the magnetic vector potentials at time point $t+\Delta t$ are unknowns and the solution procedure at each time-step is very similar to the magnetostatic case. The sub-domain equation (3.4) needs to be adjusted according to (3.9).

Note that in the magnetostatic case, the initial guess of the magnetic vector potential for each node is set to 0 since there is no information for reference before the problem is solved. The number of the required NDD iterations is less if the initial guess is closer to the correct solution. This feature is very useful for the magnetodynamic case since the final solution of each time-step can serve as the initial guess of the next time-step and contribute to the converging process. In the transient simulation where a small time-step is applied, the field usually changes slowly, implying the NDD iteration number between

two successive time-steps can be less than the magnetostatic case presented before.

For the Case 2 in Table I with 1273 nodes, 2483 elements and the prescribed relative tolerance $\epsilon_0 = 10^{-5}$, the required iteration number is 537. If the windings are fed with the time-varying sinusoidal currents provided, the initial guess of the magnetic vector potential for each time-step could be set to the solution of its previous time-step instead of 0. Figure 3.12 shows the number of iterations required for each time-step when different Δt is applied, which reveals that the required iteration number can be much less than 537 for a magnetodynamic problem. For example, the average number of iterations required is 254, 128, and 40 when the applied time-step is $100\mu s$, $10\mu s$, and $1\mu s$, respectively. And the average execution time per time-step of the Comsol$^{\text{TM}}$ time-domain solver is $2.5s$ when $\Delta t = 100\mu s$, $1.6s$ when $\Delta t = 10\mu s$, and $0.52s$ when $\Delta t = 1\mu s$. Thus, the average speedup of the NDD scheme for time-domain FE computation is 53, 60 and 70 times for the time-steps of $100\mu s$, $10\mu s$, and $1\mu s$, respectively.

3.5.4 Scalability and Limitation

As mentioned before, with unlimited parallel hardware resources, the execution time of the NDD scheme will increase linearly with the problem size and a speedup of more than 30 times is obtained compared with the commercial solver, which are very attractive features. However, in practical applications, the potential of the NDD scheme may be capped by the available parallel hardware resources. For any fixed parallel hardware resource, the execution time of the NDD scheme will eventually increase quadratically with the number of nodes N, while the time cost of the Newton-Raphson and incomplete Cholesky conjugate gradient solver will roughly increase with $N^{1.5}$, and the execution time of the commercial software only provides a linear increase with N. Therefore, to gain decent speedup for a FE problem, parallel hardware resources that match the problem size would be a prerequisite for the proposed NDD scheme.

Fortunately, the development of modern high-performance parallel computing architecture provides many possibilities for such massively parallel algorithms. In our future research, the FE problems with tens of thousands of nodes would be solved on the workstation with multiple GPUs; for larger 3-D FE problems, computer clusters would be considered, which consists of millions of computational cores.

It is also worth mentioning that due to its matrix-free property, the NDD scheme is also promising in those FE problems with dynamic mesh generation such as rotating machines with moving parts. The changing geometries only impact the attributes of the nodes and elements involved, and all the other computations remain the same because the proposed NDD scheme is essentially decentralized.

Besides, the Galerkin scheme in this chapter utilized node-based FEM, thus the subdomain is defined by a node and its neighboring triangular elements. Also, the NDD scheme can be potentially applied in edge-based FEM such as 3-D problems using tangen-

tial vector finite elements. In that case, the sub-domain consists of an edge element and all the tetrahedrons that share this edge.

3.6 Summary

In this chapter, a novel nodal domain decomposition with relaxation scheme is proposed and the massive parallelism is implemented on the prevalent parallel computing architectures. For the first time, the nonlinear FE problem in EM apparatus is solved in a decentralized manner without having to assemble a global matrix. The miniature sub-domain solver shows perfect modularity for single instruction multiple data (SIMD) programming with the specifically defined data structure, and the memory required increases linearly with the problem size. The mixed boundary condition is incorporated to the sub-domain solver to accelerate the convergence. The accuracy and efficiency of the NDD scheme implemented on both multi-core CPU and many-core GPU are discussed for different problem sizes, and comparison with the results from Comsol^{TM} shows a speedup of more than 30 times while maintaining high accuracy (error less than 0.85%). Also, for time-domain FE computation, the average speedup achieved is more than 53 times since the solution of each time-step could serve as valuable information to contribute to the convergence of the next time-step. Besides, it should be mentioned that the results of the proposed parallel method match well with the commercial software regarding the same boundary value problem. To represent the physical problem accurately, efforts will be paid to reduce the assumptions and simplifications that the finite element model depended on. Future research will focus on applying the NDD scheme in coupled electromagnetic field-transient FE computation where dynamic mesh generation is required, and extending it to large-scale 3D FE solution on compute clusters of multiple CPUs and GPUs.

4

Element-Wise Parallelism: Transmission-Line Decoupling for Transformer EMT Simulation

4.1 Introduction

This chapter provides the transmission line decoupling technique that explores parallelism at the element-level. The features including the parallelism and constant admittance matrix are elaborated and a 2D real-time finite element transformer model is implemented on FPGA using hardware concurrency and data pipelining.

Transformers are essential components in a power delivery system. The behavior of the electromagnetic field in a transformer is governed by Ampere's law, and the accurate prediction of the magneto dynamic field is commonly obtained from the FEM simulation, a powerful numerical method to deal with complex geometries and material nonlinearities.

An accurate real-time electromagnetic *field-transient* simulation of a power transformer is of great benefit for the design and testing of better control and protection schemes, cost reductions, and energy efficiency improvement. However, as the permeability of the material depends on the magnetic field strength, solving the discretized nonlinear system of field equations in *space-time* requires iterative techniques such as the Newton-Raphson algorithm within each simulation time-step. Thus, a new linear system has to be solved at each iteration due to the updated Jacobian matrix, and the dense computation makes the traditional FEM solver unbearably slow for real-time execution. In the authors' knowledge, real-time simulation is only possible by simplifying the field problem to a lumped network. The traditional lumped models [51]–[56] used for electromagnetic transient simulation of the transformer include the admittance matrix-based model and the topology-based magnetic equivalent circuit (MEC) model, both of which have been adequately utilized in real-time simulation [57], [58] for modeling various multi-winding transformers;

nonlinear effects such as hysteresis and eddy current behavior have also been sufficiently represented. Although these analytical circuit-oriented models can be calculated fast, they suffer from oversimplification and serious information loss.

The transmission line modeling method is based on Huygens's principle and was originally developed by Johns [59] to simulate wave propagation. Then the concepts of TLM *link* and *stub* were proposed for the analysis of nonlinear networks to effectively decouple the nonlinear elements from the linear system [60]–[64]. Later Lobry and Deblecker insightfully uncovered the analogy between the finite element matrix and the nodal admittance matrix of a corresponding equivalent network and successfully applied the TLM technique to the FEM solution [47], [65], [66]. The improved computational efficiency and accuracy were notably discussed.

Solid foundations have been laid by previous scholars, yet the TLM technique used in FEM can be further explored for implementation on parallel hardware architectures to achieve the most computational efficiency, especially for real-time computation. Besides, the TLM-FE models in [47], [65], [66] are all fed with current sources, and an appropriate field-circuit coupling technique is necessary to interface with the external network.

A single TLM iteration is composed of two phases: *scattering* and *gathering*. In the scattering phase, pulses are injected from the linear network to each external element. Thus, all elements are decoupled from the network and the reflected pulses of external elements can be computed individually, i.e., the scattering phase can be perfectly parallelized. In the gathering phase, the reflected pulses return and impact on the linear network. Since the admittance matrix is determined by the impedance of the TLM links and remains unchanged, only one inversion is enough in the beginning. Thus, the computation of the gathering phase includes mainly matrix-vector multiplications with a changing right-hand side vector, which is also perfectly suitable for parallel hardware architectures. In general, multiple TLM iterations are needed due to the mismatch between the impedance of the TLM link and the impedance of the element (linear or nonlinear). To decrease the number of TLM iterations required within each time-step, the impedances of the TLM links in this work are set to the memory steady-state values of the corresponding element instead of using arbitrary values.

Also, an indirect non-iterative field-circuit coupling technique is proposed to handle the external circuit. The self and mutual inductances of the transformer windings are calculated based on the magnetic vector potentials from the FE model, and these coupling coefficients are fed to the external circuit, which is then solved independently within each time-step.

In this chapter, the parallelism of the TLM-FE solution is fully explored to achieve real-time finite-element simulation of the electromagnetic transients in a 2D single-phase power transformer coupled to external circuits. The real-time simulation is performed on the Virtex UltraScale+^{TM} VCU118 FPGA board, and the results are compared with those obtained

from commercial FEM software Comsol Multiphysics™ concerning accuracy and computational efficiency.

The chapter is organized as the following. Section 4.2 presents the magneto dynamic field equations to be solved for the transformer, the TLM solution technique, and the field-circuit interface. In Section 4.3, the detailed real-time implementation of the parallelized TLM technique on the FPGA is presented. Section 4.4 provides real-time simulation results and comparisons. At last, summary and future work are provided.

4.2 Magnetodynamic Problem Formulation for Transformer FEM Simulation

4.2.1 FEM Equations

Consider a 2-D finite element model of a single-phase power transformer shown in Fig. 4.1. When the windings are excited by the external circuit, the current density in the windings produces a dynamic magnetic field, which is described by the magnetic vector potential A. Since the magnetic vector potential A and the impressed current density J only have the z-components in a 2-D problem, the nonlinear magneto dynamic problem can be defined by the following diffusion equation:

$$\nabla \cdot (\upsilon \nabla A_z) = \sigma \frac{\partial A_z}{\partial t} - J_z, \tag{4.1}$$

where υ is the field-dependent reluctivity; σ is the conductivity, a constant value for a certain material; J_z is the impressed current density, which is only nonzero in the winding zone.

The well-known Galerkin FEM [48] applied to solve the magneto dynamic problem includes the following steps: discretize the domain, form elemental equations, assemble the global matrix, and solve the system of equations.

Fig. 4.1 shows the discretized domain consisting of triangular elements and nodes. For the Galerkin method, the product of the residual and the weighted function is integrated over each element, and the element equations can be formed by forcing the integral to zero. With natural boundary conditions applied, the classical weighted-integral equation of element Ω^e can be written as [48]:

$$\begin{aligned} &\iint_{\Omega^e} \upsilon^e \left(\frac{\partial A^e}{\partial x}\frac{\partial W^e}{\partial x} + \frac{\partial A^e}{\partial y}\frac{\partial W^e}{\partial y}\right) dxdy \\ &+ \iint_{\Omega^e} \sigma \frac{\partial A^e}{\partial t} W^e dxdy = \iint_{\Omega^e} J_z^e W^e dxdy, \end{aligned} \tag{4.2}$$

where A^e is the magnetic vector potential over element Ω^e, and W^e the weighted function. In Fig. 4.1, the vertex values of the triangular element Ω^e are A_1, A_2, and A_3, and the magnetic vector potential over the whole element A^e can be expressed with the classical barycentric interpolation:

$$A^e = N_1 A_1 + N_2 A_2 + N_3 A_3, \tag{4.3}$$

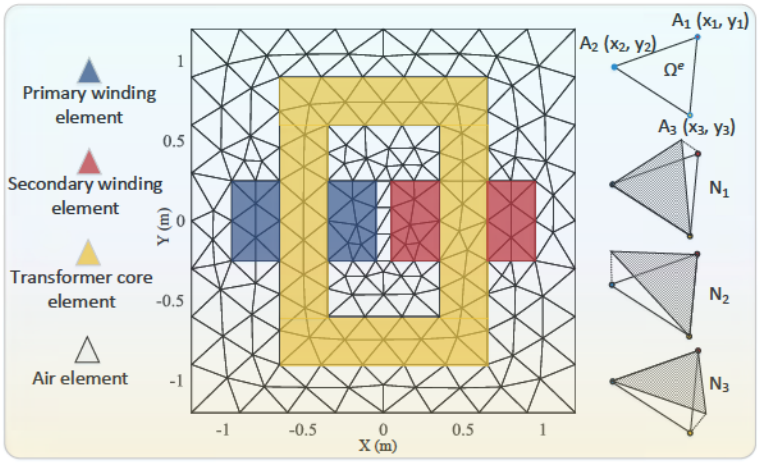


Figure 4.1: 2-D FE model of a single-phase transformer.

where N_1, N_2, and N_3 are the shape functions profiled in Fig. 4.1. The values of these shape functions at any position $(x,\ y)$ are defined by

$$N_i = \frac{1}{2\Delta^e}\,(a_i + b_i x + c_i y)(i = 1,\ 2,\ 3), \tag{4.4}$$

where Δ^e is the area of the triangular element; $a_i,\ b_i$, and c_i are related to the coordinates of the vertices:

$$\begin{aligned} a_1 &= x_2y_3 - x_3y_2,\ b_1 = y_2 - y_3,\ c_1 = x_3 - x_2, \\ a_2 &= x_3y_1 - x_1y_3,\ b_2 = y_3 - y_1,\ c_2 = x_1 - x_3, \\ a_3 &= x_1y_2 - x_2y_1,\ b_3 = y_1 - y_2,\ c_3 = x_2 - x_1. \end{aligned} \tag{4.5}$$

According to the Galerkin method, the A^e in (4.2) is substituted by (4.3) and the weighted functions W^e are set to N_1, N_2, and N_3 respectively. Thus, 3 equations with 3 unknown vertex magnetic potential values are obtained after accomplishing the integral:

$$\begin{aligned} &\frac{\upsilon^e}{4\Delta^e}\begin{bmatrix} b_1b_1 + c_1c_1 & b_1b_2 + c_1c_2 & b_1b_3 + c_1c_3 \\ b_1b_2 + c_1c_2 & b_2b_2 + c_2c_2 & b_2b_3 + c_2c_3 \\ b_1b_3 + c_1c_3 & b_2b_3 + c_2c_3 & b_3b_3 + c_3c_3 \end{bmatrix}\begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} \\ &+ \frac{\sigma^e\Delta^e}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}\begin{bmatrix} \frac{\partial A_1}{\partial t} \\ \frac{\partial A_2}{\partial t} \\ \frac{\partial A_3}{\partial t} \end{bmatrix} = \frac{J_z^e\Delta^e}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \end{aligned} \tag{4.6}$$

Note that the reluctivity υ^e, the conductivity σ^e and the current density J_z^e are constant over one finite element. The distributions of σ^e and J_z^e are given while υ^e depends on A_1, A_2, and A_3. The dependence is not straightforward and could be quantitatively expressed by introducing the magnetic flux density B. In a 2-D problem, as A is defined by $B = \nabla \times A$, in one element B and A are related by:

$$\begin{aligned} B^2 &= (\frac{\partial A^e}{\partial x})^2 + (\frac{\partial A^e}{\partial y})^2 = -\frac{1}{4(\Delta^e)^2}((b_1b_2 + c_1c_2)(A_1 - A_2)^2 \\ &+ (b_2b_3 + c_2c_3)(A_2 - A_3)^2 + (b_1b_3 + c_1c_3)(A_1 - A_3)^2). \end{aligned} \tag{4.7}$$

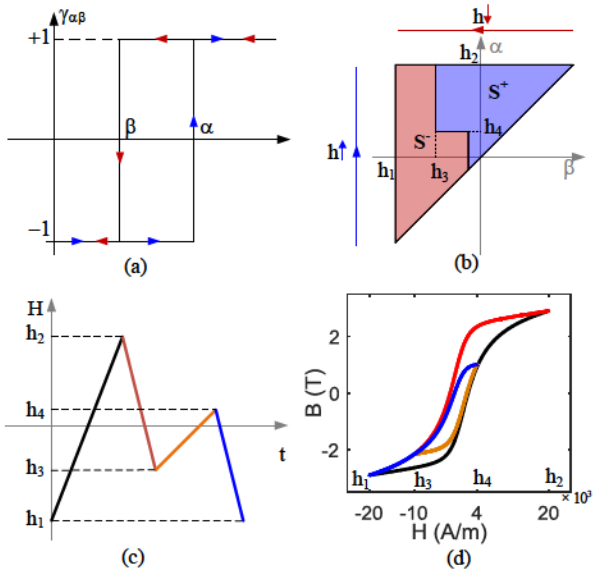


Figure 4.2: Preisach model: (a) Hysteresis operator, (b) Preisach triangle, (c) Sample input of H, and (d) Trajectory of the hysteresis loop.

The $v^e - B^2$ relationship can be found from the core material property such as the B-H curve representation. In this work, the nonlinear material property of the transformer core is represented using the Preisach model [67].

The classical Preisach model regards the ferromagnetic material as an infinite set of hysterons defined by the hysteresis operator (Fig. 4.2 (a)). Thus the magnetization moment M can be written as the following integral over the Preisach triangle (Fig. 4.2 (b)):

$$\begin{aligned} M(t) &= \iint_{\alpha \geq \beta} \mu(\alpha, \beta) \gamma_{\alpha\beta} H(t) d\alpha d\beta \\ &= \iint_{S^+} \mu(\alpha, \beta) H(t) d\alpha d\beta - \iint_{S^-} \mu(\alpha, \beta) H(t) d\alpha d\beta, \end{aligned} \tag{4.8}$$

where $H(t)$ and $M(t)$ are the input and output, the stair-shaped boundary of S^+ and S^- is determined by the historical extremum of $H(t)$, and $\mu(\alpha, \beta)$ is the Preisach distribution function that is usually obtained from the experimental results. Thus the B-H relationship can be obtained:

$$B = \mu_0 (M + H). \tag{4.9}$$

For the Newton-Raphson algorithm requiring smooth nonlinearity, it is beneficial to approximate the experimental results using an mathematical distribution function. According to [68], the following 2-D Cauchy distribution function, which is analytically integrable, can achieve an accurate approximation:

$$\mu(\alpha, \beta) = c \frac{1}{1 + (\frac{\alpha - a_0}{b0})^2} \frac{1}{1 + (\frac{\beta - a_1}{b1})^2}, \tag{4.10}$$

where a_0, a_1, b_0, b_1, and c are coefficients given in the Appendix A.

Since the magnetization moment M depends on the past history of H, a reversal point vector (RPV) storing the historical extremum (h_1, h_2, h_3, h_4, ...) is required to derive the integral. Figure 4.2 (c) and Fig. 4.2 (d) present a sample input with 4 branches and the corresponding trajectories of the hysteresis loop. The RPV is updated or wiped out (branch $h_4 \rightarrow h_1$) based on the input sequence $H(t)$, and the B-H formulation in different branches are regulated by the RPV.

The problem domain is enclosed by an artificial rectangular boundary Γ (shown in Fig. 4.1), and the boundary condition is $A_\Gamma = 0$. After time discretization, the above nonlinear elemental equations (4.6) can be assembled and are ready to be solved with the Newton-Raphson iterative scheme. Concerning the computational burden, the TLM technique provides an efficient methodology to handle these nonlinear equations.

4.2.2 Refined TLM Solution

When the TLM technique is applied to solve an electrical circuit in the time domain, the components are separated from the network by lossless transmission lines. The characteristic impedance of the transmission line Z_c is arbitrary for resistors, $\Delta t/2C$ for capacitors, and $2L/\Delta t$ for inductors, where Δt is the simulation time-step [60].

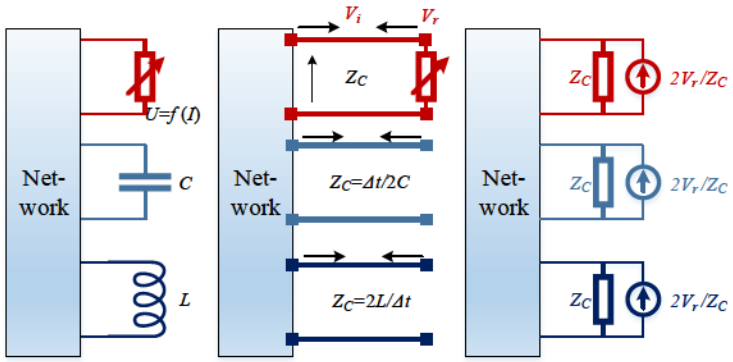


Figure 4.3: The TLM technique applied in an electrical network.

As shown in Fig. 4.3, in the scattering phase, the network transmits incident impulses V_i into each element. For linear elements, the reflected pulse can be calculated by

$$V_r = K_r V_i, \tag{4.11}$$

where the reflection coefficient K_r is equal to $(R-Z_c)/(R+Z_c)$ for resistors, 1 for capacitors, and -1 for inductors. For a nonlinear element, for example, a nonlinear resistor described as $U = f(I)$, the reflected pulse V_r is obtained by solving the nonlinear equation

$$V_i + V_r = f(\frac{V_i - V_r}{Z_c}). \tag{4.12}$$

Note that if Z_c is equal to the working state value of the linear or nonlinear resistor, the reflection coefficient is 0 and only one TLM iteration is required.

In the gathering phase, all the reflected pulses return to the network and update the values of all nodes. The transmission line and the elements are replaced with their equivalent Norton circuits. For each element, the equivalent resistance is the characteristic impedance Z_c of the transmission line and the equivalent current is $\frac{2V_r}{Z_c}$ according to the TLM theorem [69]. A global linear network needs to solve, and the nodal admittance matrix, which is determined by the characteristic impedances and is independent of the elements, remains unchanged. It is a significant feature providing an improvement in computational efficiency.

The nodal values can be updated by solving a linear network with unchanged nodal admittance matrix, then the next incident pulse can be also updated by:

$$V_i = V_d - V_r, \tag{4.13}$$

where V_d is the nodal voltage difference. Thus, the gathering phase is complete, and the scattering phase of the next TLM iteration is ready to begin.

At each time step, multiple TLM iterations described above are necessary to fully solve the nonlinearity. Note that the round-trip travel time is infinitely short for a resistive element. Therefore, at the beginning of each time step, the TLM iteration is applied repeatedly to all nonlinear resistive elements until convergence. During this process, the incident and reflected pulses of all reactive elements are not yet updated. After steady-state is reached, the incident and reflected pulses of the reactive elements are updated for the next time step.

Equation (4.6) defines an equivalent network composed of capacitors and nonlinear resistors shown in Fig. 4.4(a). The magnetic vector potential is modeled as the electric potential, the right-hand side vector of (4.6) is modeled as the current sources and the Dirichlet boundary conditions are modeled as the voltage sources. The nonlinear resistors and the linear capacitors form the coefficient matrix and hold such material properties as reluctivity and conductivity. And the values of the components are given as

$$\begin{aligned}
&G_{12} = -\frac{\upsilon^e}{4\Delta^e}(b_1b_2 + c_1c_2), \;\; Y_{G12} = -\frac{\upsilon_g^e}{4\Delta^e}(b_1b_2 + c_1c_2),\\
&G_{13} = -\frac{\upsilon^e}{4\Delta^e}(b_1b_3 + c_1c_3), \;\; Y_{G13} = -\frac{\upsilon_g^e}{4\Delta^e}(b_1b_3 + c_1c_3),\\
&G_{23} = -\frac{\upsilon^e}{4\Delta^e}(b_2b_3 + c_2c_3), \;\; Y_{G23} = -\frac{\upsilon_g^e}{4\Delta^e}(b_2b_3 + c_2c_3),\\
&C_{12} = C_{13} = C_{23} = -\frac{\sigma^e\Delta^e}{12}, \;\; Y_{C12} = Y_{C13} = Y_{C23} = -\frac{\sigma^e\Delta^e}{6\Delta t},\\
&C_{10} = C_{20} = C_{30} = \frac{4\sigma^e\Delta^e}{12}, \;\; Y_{C10} = Y_{C20} = Y_{C30} = \frac{4\sigma^e\Delta^e}{6\Delta t}.
\end{aligned} \tag{4.14}$$

where υ^e is the prior unknown reluctivity to be solved using the 3×3 Newton-Raphson scheme and υ_g^e is an initial guess value, which should be as close to υ^e as possible. With

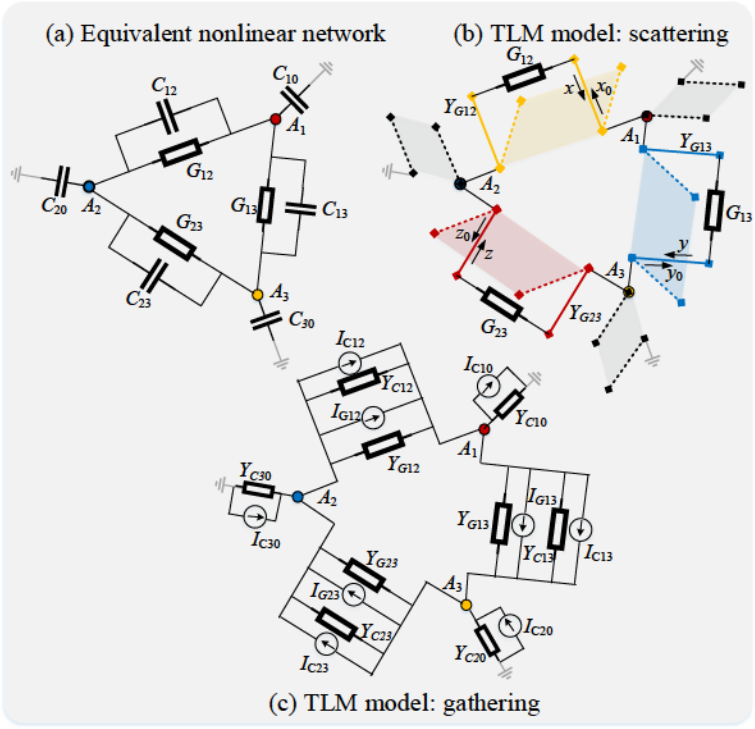


Figure 4.4: TLM technique applied to elemental equations of the FEM.

the guessed v_g^e, the nodal admittance matrix required in the gathering phase could be obtained by assembling multiple Norton equivalent circuits (Fig. 4.4(c)) based on the FE mesh information.

When the TLM technique is applied in a triangular network, three pulses based on the differences between nodal values will be injected into the three edges. Figure 4.4(b) shows the TLM links (resistors) and the TLM stubs (capacitors) in the scattering phase. The incident pulses into three edges are denoted by x_0, y_0, and z_0 and the reflected pulses to be calculated are denoted by x, y, and z, respectively. Instead of solving a single equation (4.12), the following three coupled nonlinear algebraic equations need to be solved:

$$\begin{aligned} G_{12}(x+x_0) - Y_{G12}(x_0-x) &= 0, \\ G_{13}(y+y_0) - Y_{G13}(y_0-y) &= 0, \\ G_{23}(z+z_0) - Y_{G23}(z_0-z) &= 0. \end{aligned} \tag{4.15}$$

Since G_{12}, G_{13} and G_{23} are functions of $(x+x_0)$, $(y+y_0)$ and $(z+z_0)$, and the relations can be derived from (4.7) and the Preisach model, the resulting nonlinear algebraic equations can be efficiently solved with the Newton-Raphson method, and the convergence can be sped up using an appropriate relaxation factor. When calculating the 3×3 Jacobian matrix, the following chain rule is utilized:

$$\frac{\partial G}{\partial x} = \frac{\partial G}{\partial B^2}\frac{\partial B^2}{\partial x}. \tag{4.16}$$

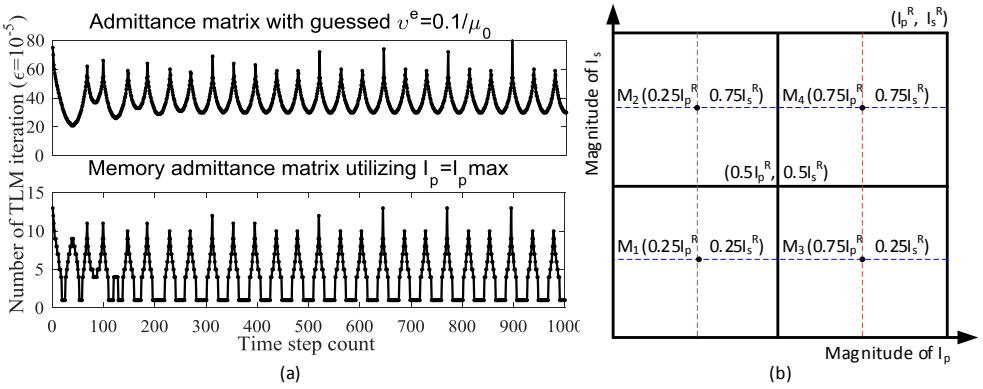


Figure 4.5: (a) TLM iteration number required utilizing guessed and memory admittance matrix, (b) Four memory matrices formation and usage.

In the gathering phase shown in Fig. 4.4(c), the reflected pulses and the TLM links or stubs are replaced with their equivalent Norton circuits. The equivalent Norton circuits make a a linear network and the nodal values can be updated after matrix-vector multiplications if the inversion of the admittance matrix is ready at the beginning.

It should be emphasized that the above descriptions of the TLM technique applied in FEM are at the triangular element level. For a problem domain composed of numerous finite elements, the main computational tasks include solving (4.15) for all nonlinear elements in the scattering phase and the matrix-vector multiplications in the gathering phase. Both tasks could be perfectly parallelized on massively parallel computing architectures.

It is acknowledged that multiple TLM iterations are needed because of the mismatch of the unknown υ^e and the guessed υ_g^e. If υ^e is equal to υ_g^e, the reflected pulse is zero and only one TLM iteration is enough for convergence. Fewer TLM iterations are needed if the two values are closer. Thus, the concept of *memory* nodal admittance matrix is proposed to decrease the mismatch and to accelerate the convergence. A memory admittance matrix is featured by the impressed current density. For the transformer, once the FE model is solved with provided I_{p0} and I_{s0}, the υ^e of each element can be extracted. After the assembly and inversion process, these values could be utilized to form a new admittance matrix, which contains very valuable information (memory) for the rest solution.

Figure 4.5(a) shows the efficiency of the memory admittance matrix in a case study (presented in SectIion IV) with secondary winding open-circuited ($I_s = 0$). As shown in the figure, even when the υ_g^e is carefully guessed, it takes more than 30 TLM iterations for each time-step while the number is around 10 utilizing the memory admittance matrix. Note that I_p^{max} in Fig. 4.5(a) could be set to the peak value of the rated primary winding current.

In this work, a total of four memory admittance matrices are applied to ensure that the TLM iteration number is always less than 5 provided that I_p and I_s change arbitrarily within each time-step, as shown in Fig. 4.5(b), where I_p^R and I_s^R are the rated currents in

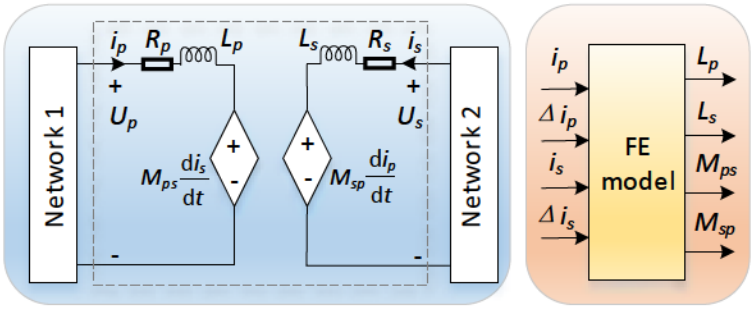


Figure 4.6: Equivalent FE transformer model coupled to external networks.

the primary and secondary windings of the transformer. Different admittance matrices are used for the gathering phase based on the combination of the I_p and I_s. The decreased TLM iteration number and the parallelism within a TLM iteration make real-time simulation possible on the parallel FPGA hardware architecture.

4.2.3 Field-Circuit Coupling

Note that the inputs of the FE model described above are the winding currents. However, since a transformer is generally excited by voltage sources in a power system, a field-circuit coupling technique is necessary to interface with the external circuit.

Direct coupling and indirect coupling approaches have been explored to interface the FE model and the external circuit: the direct coupling approach solves the field equations and the electrical network equations simultaneously while the indirect approach solves the two subsystems separately [70]. The computational burden of the direct coupling approach is heavier than the traditional FEM solver because the circuit equations destroy the matrix symmetry [71].

In this chapter, we proposed an indirect coupling technique by accurately extracting the self and the mutual inductances of the transformer from the FE model. Within each time-step, the field equations and the circuit equations are solved separately without having to apply an iterative scheme.

In electrical networks, a transformer model can be represented by self and mutual inductances with controlled sources, as shown in Fig. 4.6. The accurate values of these nonlinear inductances (depends on the winding currents) can be obtained with the FE model. The electric voltage U across a winding can be represented by the magnetic vector potential and the voltage drop of winding's resistance:

$$U = rI + \frac{Nl}{\Delta_S}\int_S \frac{\partial A}{\partial t} dS, \tag{4.17}$$

where I is the winding current, r the winding resistance, N the number of turns, l the axial length of each filament, S the winding zone, and Δ_S the area of the winding zone.

Since A is a function of the primary winding current i_p and the secondary winding current i_s based on the FE model, (4.17) can be rewritten as:

$$U = rI + \frac{Nl}{\Delta_S}\int_S \frac{\partial A}{\partial i_p} dS \frac{\partial i_p}{\partial t} + \frac{Nl}{\Delta_S}\int_S \frac{\partial A}{\partial i_s} dS \frac{\partial i_s}{\partial t}. \tag{4.18}$$

Applying (4.18) to the primary and the secondary winding respectively, the self and mutual inductances of the transformer can be obtained as:

$$\begin{aligned} L_p &= \frac{N_p l_p}{\Delta_{S_p}}\int_{S_p} \frac{\partial A}{\partial i_p} dS,\ M_{ps} = \frac{N_p l_p}{\Delta_{S_p}}\int_{S_p} \frac{\partial A}{\partial i_s} dS, \\ M_{sp} &= \frac{N_s l_s}{\Delta_{S_s}}\int_{S_s} \frac{\partial A}{\partial i_p} dS,\ L_s = \frac{N_s l_s}{\Delta_{S_s}}\int_{S_s} \frac{\partial A}{\partial i_s} dS. \end{aligned} \tag{4.19}$$

Since

$$\begin{aligned} \frac{\partial A}{\partial i_p} &= \frac{A(i_p + \Delta i_p, i_s) - A(i_p, i_s)}{\Delta i_p}, \\ \frac{\partial A}{\partial i_s} &= \frac{A(i_p, i_s + \Delta i_s) - A(i_p, i_s)}{\Delta i_s}, \end{aligned} \tag{4.20}$$

these coupling coefficients can be calculated with the FE model by setting the Δi_p and Δi_s in (4.20) to a very small value such as 0.1A.

It is known that by applying Trapezoidal rule, a constant inductor in the network can be replaced by a resistor $\frac{2L(t)}{\Delta t}$ in parallel with a current source I_{his}. When the inductor value $L(t)$ is time-varying, both $L(t)$ and $L(t + \Delta t)$ should be used. Since $L(t + \Delta t)$ is unknown, an error exists in the proposed indirect coupling scheme because the time-varying inductances at time point t, calculated from $i_p(t)$ and $i_s(t)$, are utilized to solve the external networks at time point $t + \Delta t$. To reduce the asynchronization error, the inductance values at time point $t + \frac{\Delta t}{2}$ are used with a prediction scheme and the following central difference formula is utilized:

$$\begin{aligned} \frac{\partial A}{\partial i_p}(t + \frac{\Delta t}{2}) &\approx \frac{A(i_p(t + \Delta t), i_s(t)) - A(i_p(t), i_s(t))}{i_p(t + \Delta t) - i_p(t)}, \\ \frac{\partial A}{\partial i_s}(t + \frac{\Delta t}{2}) &\approx \frac{A(i_p(t), i_s(t + \Delta t)) - A(i_p(t), i_s(t))}{i_s(t + \Delta t) - i_s(t)}. \end{aligned} \tag{4.21}$$

where $i_p(t + \Delta t)$ and $i_s(t + \Delta t)$ are unknown and can be predicted from their historical values:

$$\begin{aligned} i_p(t + \Delta t) &= 2i_p(t) - i_p(t - \Delta t), \\ i_s(t + \Delta t) &= 2i_s(t) - i_s(t - \Delta t). \end{aligned} \tag{4.22}$$

Simulation results show that the mean absolute relative error is less than 3% with $\Delta i_p = i_p(t) - i_p(t - \Delta t)$, $\Delta i_s = i_s(t) - i_s(t - \Delta t)$ and more than 5% using small values (0.1A) of Δi_p and Δi_s.

Note that the calculation of $A(i_p, i_s)$, $A(i_p, i_s + \Delta i_s)$, and $A(i_p + \Delta i_p, i_s)$ can be also parallelized with enough hardware resources. Then, these parameters are fed to the external electrical network for simulation using the electromagnetic transients program (EMTP) approach [1].

4.3 Real-Time Hardware Emulation of the Finite Element Transformer Model

The hardware design was implemented on the Xilinx Virtex UltraScale+^{TM} VCU118 board with the xcvu9p-flga2104-e-es1 FPGA using high-level synthesis and optimization. The sequential C and C++ functions can be compiled down to hardware IP blocks with inputs, outputs, and handshake signals for interfacing, which can be optimized based on user's preferences by adding the directives such as array partitioning, loop unrolling and pipelining.

According to the TLM-FE solution and the field-circuit coupling technique, six hardware blocks are generated and optimized to achieve deep data pipelining on the FPGA. The detailed real-time implementation on the FPGA is shown in Fig. 4.7. The block connections, pseudo-codes, and the finite state machine are all illustrated. The functionalities of these blocks are presented as the following:

1. *Assemble Current*: The currents generated by the reflected pulses in each triangular element (Fig. 4.4) and the impressed source currents are assembled to obtain the total current injected to each node. The assembly of each node is parallelized and pipelined based on the node-element connection matrix.

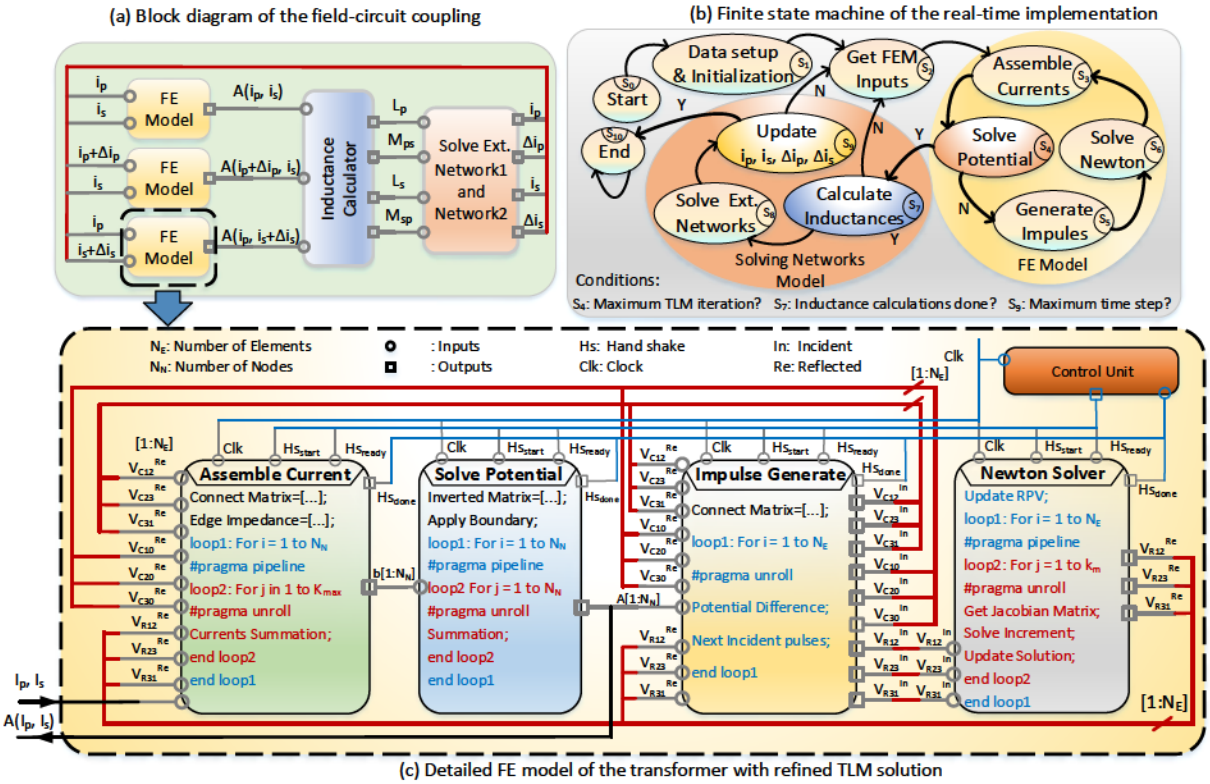
2. *Solve Potential*: With the assembled current vector and the inverted nodal admittance matrix, the magnetic potential of each node is calculated by the matrix-vector multiplication under the Dirichlet boundary conditions. The multiplications could be parallelized and the summation could be accomplished using a tree adder with a latency of $O(log_2 N)$.

3. *Impulse Generator*: Based on the updated magnetic vector potential and the reflected pulses, the new incident pulses are generated for each triangular element. For each element, these operations are unrolled and executed in parallel.

4. *Newton Solver*: For each triangular element, the 3 incident pulses are taken as inputs to obtain the 3 reflected pulses. The Newton solver is also unrolled and fully pipelined to treat multiple elements. In this work, by using a relaxation factor of 1.2, 6 iterations are enough for all elements to maintain a relative tolerance of 10^{-5}.

5. *Inductance Calculator*: Once $A(i_p, i_s)$, $A(i_p, i_s + \Delta i_s)$, and $A(i_p + \Delta i_p, i_s)$ are available, the self and mutual inductances can be calculated by the integration in (4.19).

6. *Solve External Networks*: The transformer is converted to the equivalent model in Fig. (4.6) with all inductances known, and the external electrical networks are solved in the time-domain. Note that although the execution time of the external network solution is slight compared with the FE model, the extracted inductances are also time-varying and will change the admittance matrix of the electrical network.



(c) Detailed FE model of the transformer with refined TLM solution

Figure 4.7: Real-time hardware emulation of the finite-element computation for the transformer on FPGA.

Table I shows the FPGA hardware utilization and the timing report of the case study with an FE model composed of 196 nodes and 358 elements, which are compiled on both Xilinx Virtex UltraScale+^{TM} xcvu9p and xcvu13p FPGAs, and the details of the case study are presented in the next section.

Although the FPGA clock frequency of each hardware block can be set to more than 115MHz, the maximum clock frequency of the design can reach 93.1MHz for xcvu9p and 94.2MHz for xcvu13p after the hardware blocks are interconnected. Since 5 TLM iterations are required for reasonable accuracy, according to the timing report, the execution time of the FE hardware block is $5 \times 1523 \div 93.1 = 81.8\mu s$ on the xcvu9p FPGA and $5 \times 1536 \div 94.2 = 81.6\mu s$ on xcvu13p while the latency of the circuit model is $1.7\mu s$. According to the hardware utilization report, two FE hardware blocks can be generated on the xcvu9p FPGA and three FE hardware blocks on the xcvu13p FPGA. Therefore, for the design on the xcvu9p FPGA, two of $A(i_p, i_s)$, $A(i_p, i_s + \Delta i_s)$, and $A(i_p + \Delta i_p, i_s)$ can be calculated in parallel and the other one is executed sequentially thereafter, and the total execution time of the implementation is, therefore, $166\mu s$. For the xcvu13p FPGA, the calculations of $A(i_p, i_s)$, $A(i_p, i_s + \Delta i_s)$, and $A(i_p + \Delta i_p, i_s)$ can all be executed in parallel at the same time and the total execution time of the implementation is $84\mu s$.

Although an execution time of $166\mu s$ of the design on UltraScale+^{TM} xcvu9p FPGA in

Table 4.1: Hardware Resource Utilization and Timing Report

FPGA device	Module	Resource Utilization				Latency (clock cycles)
		BRAM	DSP	FF	LUT	
Xilinx UltraScale+ xcvu9p	Assem. curr.	84	512	130316	253004	393
	Solv. poten.	168	842	86841	64301	206
	Imp. gener.	0	24	4418	21316	157
	Newt. solv.	9	1646	194969	200509	767
	Subtotal (FEM)	6%	44.2%	17.6%	45.6%	1523
	Calc. induc.	0	33	4971	5237	71
	Solve netw.	0	5	1611	1758	82
	Subtotal (Netw.)	0%	1%	0%	0%	153
Xilinx UltraScale+ xcvu13p	Assem. curr.	84	512	131490	257507	399
	Solv. poten.	168	842	86876	64805	208
	Imp. gener.	0	24	5187	21524	159
	Newt. solv.	9	1646	214112	203558	770
	Subtotal (FEM)	4.9%	24.6%	12.7%	31.7%	1536
	Calc. induc.	0	33	4991	5287	72
	Solve netw.	0	5	1811	1958	84
	Subtotal (Netw.)	0%	1%	0%	0%	156

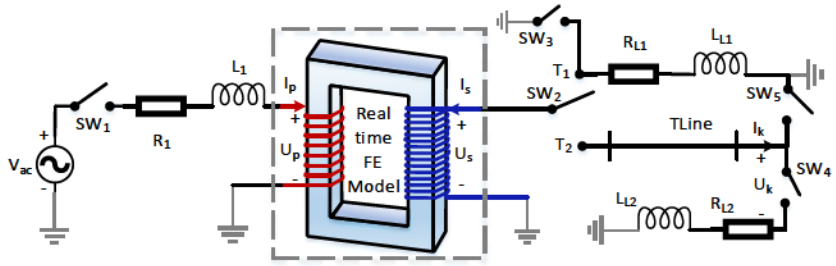


Figure 4.8: Schematic of the transformer FE model coupled with external networks for case studies.

our real-time implementation is still too large for the electromagnetic transients requiring a time-step of $50\mu s$, it should be pointed out that the proposed indirect coupling scheme works perfectly for multi-rate real-time simulation, which means that the external networks can utilize a much smaller time-step than that of the FE model. For example, in Case Study II presented in Section 4.4, the time-step is $180\mu s$ for the FE transformer model and $45\mu s$ for the external networks, implying that the transformer inductances used in the solution of the external network module are updated every 4 time-steps. In our future work implemented on the UltraScale+^{TM} xcvu13p FPGA, the applied simulation time-step for the FE model could be $84\mu s$.

4.4 Case Study and Results

4.4.1 Setup for Case Study

A single-phase power transformer rated at 37.5kV/202kV was studied in this work, and the geometry and mesh (196 nodes and 358 elements) are shown in Fig. 4.1. The schematic

of the real-time FE model coupled to external networks is shown in Fig. 4.8. The primary winding is excited by a 60Hz AC voltage source, and a Bergeron line model is included between the secondary winding and the load. The detailed simulation parameters are provided in the Appendix A.

Two case studies are presented: a single-rate case study and a multi-rate case study. For the Case Study I, the total simulation time is 600ms and the applied time-step is 180μs for the FE transformer model and the external networks. The following events are simulated:

1. t=0, SW_1 is turned on and the transformer is energized while the secondary winding is open-circuited.
2. t=150ms, SW_2 is turned to T_1 and the transformer works with a load of R_{L1} and L_{L1}.
3. t=300ms, the second and the fourth harmonics are injected into the voltage source V_{ac}.
4. t=450ms, SW_3 is turned on and the secondary winding is short-circuited.

For Case Study II, the total simulation time is 600ms and the applied time-step is 180μs for the FE transformer model and 45μs for the external networks. The following events are simulated:

1. t=0, SW_1 is turned on and SW_2 is turned to T_2, the transformer is connected to a transmission line with an open-circuited receiving end.
2. t=150ms, SW_4 is turned on and the the transmission line is loaded with R_{L2} and L_{L2}.
3. t=300ms, the second and the fourth harmonics are injected into the voltage source V_{ac}.
4. t=450ms, SW_5 is turned on and the transmission line is short-circuited.

4.4.2 Results and Validation

The off-line simulation with the same case study, setup and mesh were also performed on the commercial FEM software Comsol$^{\mathrm{TM}}$. Mesh dependency test has been performed in Comsol$^{\mathrm{TM}}$ and it turned out that for this 2-D problem, 196 nodes are enough to maintain reasonable accuracy compared with a finer mesh. The off-line results are regarded as reliable and are thus used as the benchmark for comparison for the real-time results. The real-time simulation results of the primary winding current I_{p}, the primary winding voltage U_{p}, the secondary winding current I_{s}, the secondary winding voltage U_{s}, the transformer inductances, the time-varying winding loss, eddy current loss, hysteresis loss, and total loss are all presented in Fig. 4.12 for Case Study I and Fig. 4.13 for Case Study II, respectively, with a comparison of those results from Comsol$^{\mathrm{TM}}$. Note that all the errors in the figures denote the mean absolute relative error. The zoomed-in plot of the simulation

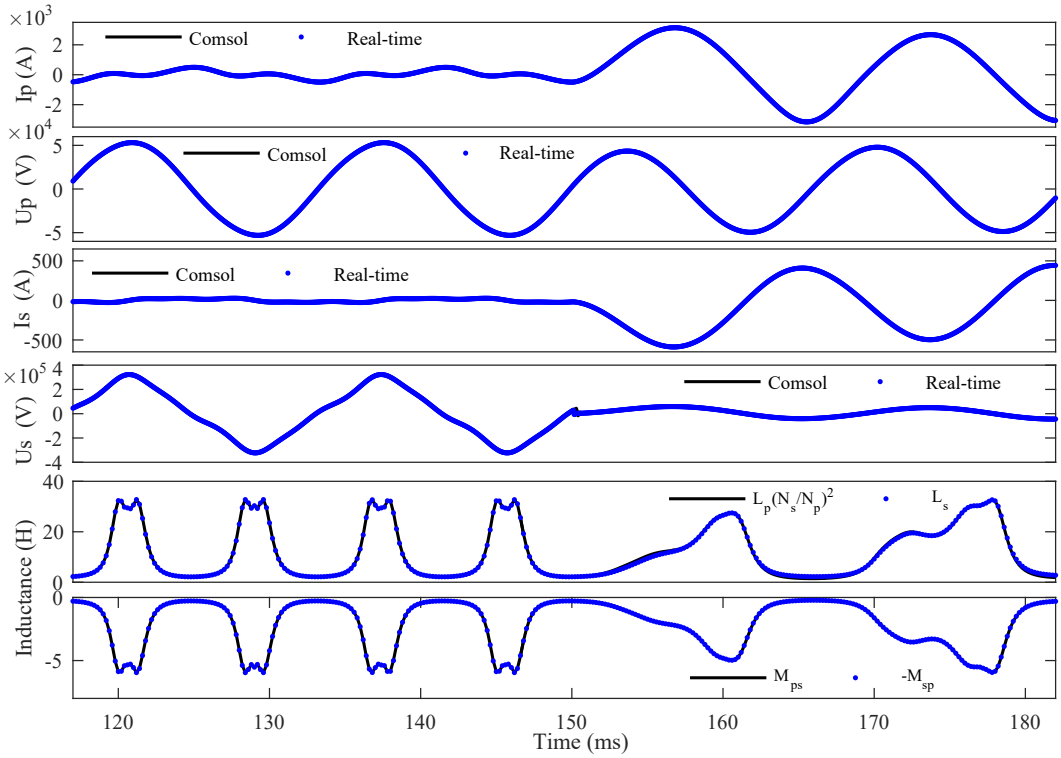


Figure 4.9: Zoomed-in time-domain result comparison in Case Study II between 117ms and 182ms.

results is shown in Fig. 4.9. For all practical purposes, the real-time results are identical to the Comsol^{TM} results, and the harmonics from Fast Fourier transform of the results also match in Fig. 4.12 and Fig. 4.13.

As can be seen from the results, when the secondary winding is open-circuited, $\mathrm{I_p}$ is the magnetizing current and the phase difference of $\mathrm{I_p}$ and $\mathrm{U_s}$ is always 90 degree. The inrush current of the primary winding occurs at $\mathrm{t} = \mathrm{t_1}$ while the inrush current of the secondary winding occurs at $\mathrm{t} = \mathrm{t_5}$. The inrush currents may cause some unwanted phenomena such as saturation in the transformer, which is possible to be monitored in real-time with the proposed implementation.

The direct output of the FE model is the magnetic vector potential A, and Fig. 4.10 shows the node solution of the real-time implementation compared with the Comsol^{TM} at $\mathrm{t} = \mathrm{t_1}$ in Case Study I and at $\mathrm{t} = \mathrm{t_5}$ in Case Study II. Other variables can also be derived after post-processing. For example, the induced electric field E and the eddy current loss L_{ec} can be calculated with

$$E = \frac{\partial A}{\partial t}, \quad L_{ec} = \iint_{\Omega_{core}} \sigma^e E^2. \tag{4.23}$$

Fig. 4.11 shows the magnetizing current and the trajectories of the hysteresis loop in a sample element in Case Study I, and the hysteresis loss can be also calculated based on these loops, which are regulated by the RPV. The detailed field distributions of the

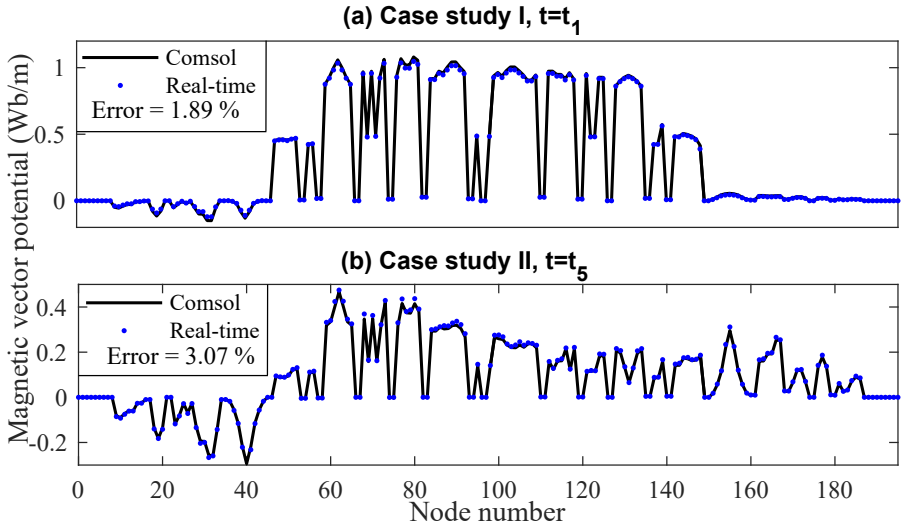


Figure 4.10: Magnetic vector potential comparison at t_1 in Case Study I and at t_5 in Case Study II.

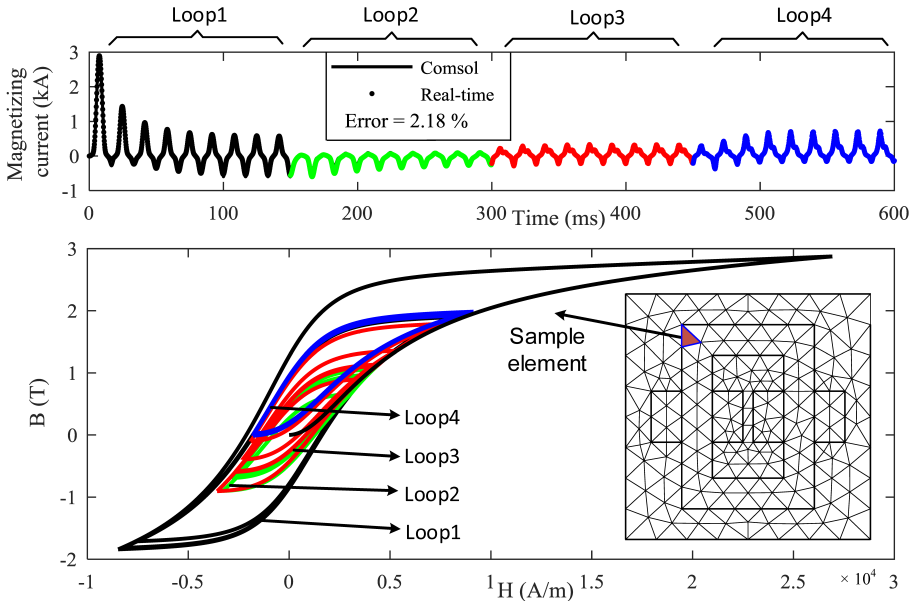


Figure 4.11: Magnetizing current and the trajectories of the hysteresis loop of a sample triangular element in Case Study I.

magnetic vector potential A (Wb/m), the magnetic flux density B (T), the magnetic field strength H (A/m), and the eddy current density J (A/m^2) at $t = t_1$ in Case Study I and at $t = t_5$ in Case Study II are also captured and presented in Fig. 4.12 and Fig. 4.13, respectively.

Note that the relative error compared with Comsol^{TM} in the case studies is caused by two factors: the FE-TLM solution and the non-iterative coupling scheme. It turned out that given the same I_p and I_s, the error of the magnetic vector potential from the real-time

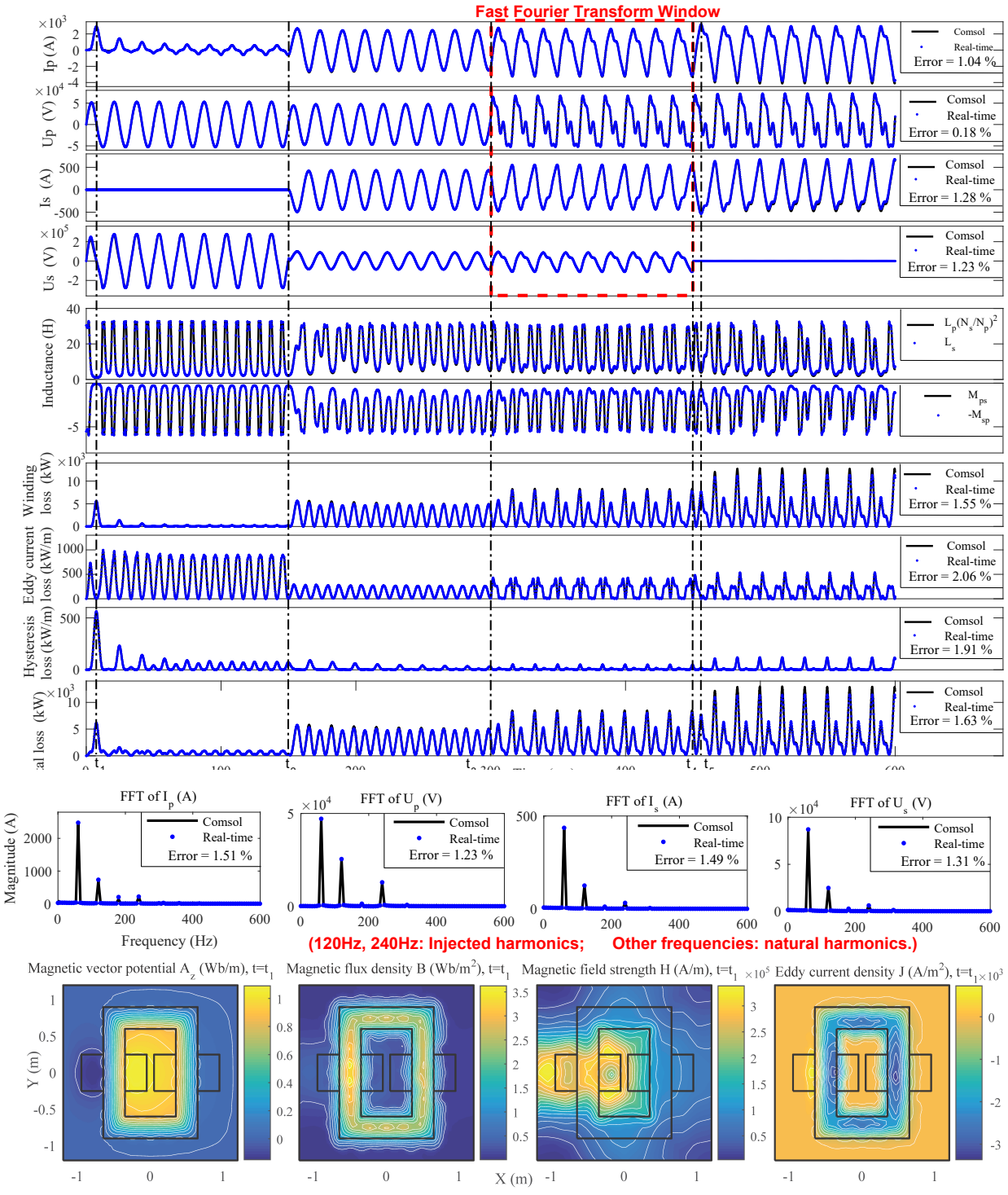


Figure 4.12: Real-time simulation results and the comparison with Comsol^{TM} off-line results for **single-rate** Case Study I: time-domain, frequency-domain, and field distributions.

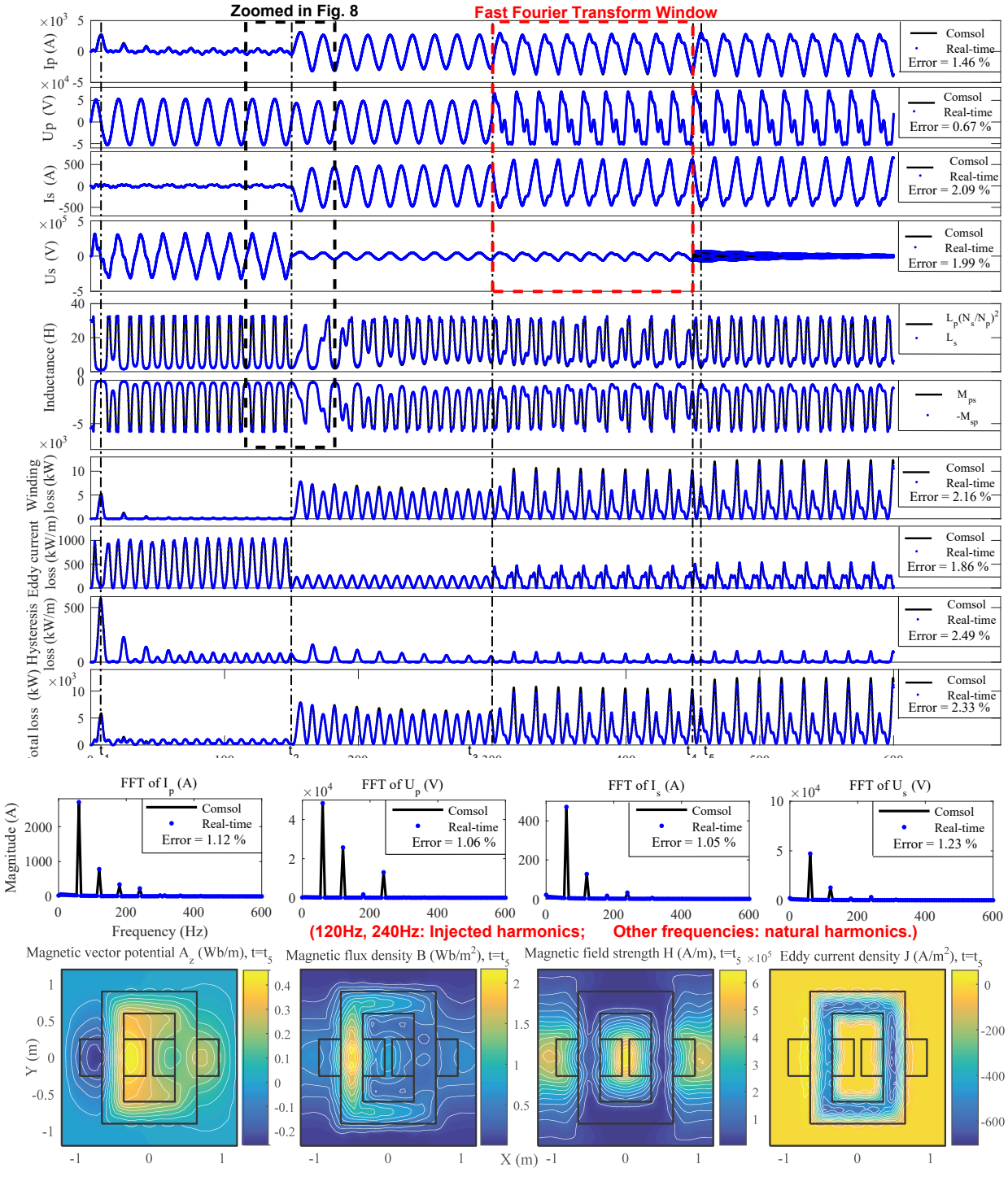


Figure 4.13: Real-time simulation results and the comparison with Comsol^{TM} off-line results for **multi-rate** Case Study II: time-domain, frequency-domain, and field distributions.

results is less than 0.1% compared with the Comsol™ solver. Further increasing the TLM iteration and the Newton iteration will make this error smaller. Thus we believe the error in our work mainly comes from the non-iterative coupling scheme.

4.4.3 Speed-up and Scalability

The off-line simulation with Comsol™ was conducted on a PC with the CPU Intel Xeon E5-2620, 16 cores, 2.1GHz clock frequency, and 32GB RAM. The tolerance of the Comsol™ nonlinear solver is 10^{-5} and the maximum iteration number is set to 20. The order of the finite element is 1. It takes $134s$ to run 3333 simulation time-steps, and the execution time per solution step is $40ms$. By sufficiently exploring the parallelism and deep data pipelining, the speed-up of the real-time hardware implementation can reach 241 times.

The proposed real-time approach also has good scalability. Although a simple circuit is used in the case study, the real-time FE model can be easily coupled to a larger complicated external network, which can be solved independently with the accurate inductances obtained from the FE model. When applied for complex geometries that require increasing the number of nodes and elements, the resource usage of the *Assemble Current* and the *Newton Solver* hardware blocks remains unchanged and only the pipelined loop count increases, and the hardware resources usage and latency of other blocks increase linearly or logarithmically.

4.5 Adaptive Admittance Matrix

The unchanged admittance matrix implies that only one LU decomposition (or matrix inversion) at the beginning of the program is enough, and for the gathering phase, the rest numerical operations to solve the linear network are merely backward and forward substitution (or matrix-vector multiplication). However, in real applications, the real value of the nonlinear reluctivity $\upsilon^e(t)$ is always time-varying. Thus the extent of mismatch between the constant υ_g^e and the changing $\upsilon^e(t)$ will fluctuate, and the required number of TLM iterations may also vary from tens to thousands.

In fact, at time-point t, the solution can provide the real values of $\upsilon^e(t)$ for all triangular elements. If this information is used to update the transmission-line impedances, implying $\upsilon_g^e(t) = \upsilon^e(t)$, the required TLM iteration number for the next time-step $t + \Delta t$ will be substantially decreased because the values of $\upsilon_g^e(t)$ and $\upsilon^e(t + \Delta t)$ within each triangular element are very close.

It has been tested that for a larger finite element problem, hundreds of iterations are required using constant admittance matrix, while only 2-4 iterations with adaptive admittance matrices. Whereas, due to the modified $\upsilon_g^e(t)$ at each time-step, the admittance matrix will also keep changing although only 2-4 iterations are required. In the future work, some iterative sparse solvers such as the conjugate gradient solver, which can also

be implemented on GPU, can be utilized to achieve fast matrix solver and reduce TLM iterations at the same time.

4.6 Summary

In this chapter, real-time finite-element computation of electromagnetic transients in a transformer is attempted for the first time. The transmission line modeling method successfully decoupled the nonlinear elements from the linear network so that the computation can be massively parallelized. The parallelism of the different phases of the TLM technique is fully explored and implemented on an FPGA board with deep data pipelining. The proposed field-circuit coupling enabled the real-time FE model to effectively interface with the external circuit to capture the transient behavior of interest. Tests conducted in comparison with a commercial FEM package prove excellent accuracy and computational efficiency of the real-time FEM approach, which provides unprecedented data detail of the simulated transformer. It should be mentioned that the results of the proposed parallel method match well with the commercial software regarding the same boundary value problem.

Potential applications of the proposed real-time FE model are many, allowing detailed modeling of the power transformer in AC and DC networks. To represent the physical problem accurately, efforts have been paid to reduce the assumptions and simplifications that the finite element model depended on. For example, the proposed approach has been extended for 3D nonlinear finite element modeling with edge element [72].

5

Parallel Finite-Difference Computation of Ionized Field Around Transmission Line

5.1 Introduction

High-voltage direct current has such advantages as lower cost and lower power loss over alternating current for bulk power transmission over long distance that new projects are spouting up world-wide. Environmental problems caused by the occurrence of corona on the high voltage conductor have received much attention with the widespread use of HVDC transmission lines in the last few decades [73].

Since these transmission lines are generally operated above their corona onset voltage, space charges are generated around the energized conductor. These space charges migrate in a manner determined by the electric field; at the same time, the electric field is modified by these space charges. The mutual interaction of electric field and space charges eventually leads to a sustained steady-state, which is governed by Poisson's equation and current continuity equation for unipolar lines [74]. The physical details of the corona and the mutual interaction process are often deemphasized, therefore investigators usually focused on obtaining the solution of the mathematical model, which can be necessarily described by coupled nonlinear partial differential equations (PDE) for 2-D problems.

The analytical solution of an ionized field was first obtained by Townsend in 1914 although it was only applicable for cases with regular geometry such as concentric spheres or coaxial cylinders [75]. For such 2-D problems as conductor-to-ground arrangements, the solutions of the coupled nonlinear PDEs relied on some simplifying assumptions, among which Deutsch's assumption was exclusively employed by investigators before the 1970s [74]. Deutsch's assumption reduced the 2-D problem to 1-D computation along flux lines by assuming that the space charges affect only the magnitude and not the direction of

the electric field [76]. This assumption is still utilized in methods such as the flux tracing method owing to its simplicity [77]–[79]. Janischewskyj et al. for the first time solved the PDEs without resorting to Deutsch's assumption using the finite element method in 1979 [5]. Then Takuma et al. in 1981 proposed the upstream FEM to overcome numerical instability caused by the accumulated error in each iteration [80]. To handle the nonlinearity of the problem, both [5] and [80] solved the coupled equations iteratively based on a predictor-corrector algorithm. Thereafter FEM was dominantly used in ionized field calculation. More complicated configurations which considered the effect of wind velocity, bipolar conductors and bundled conductors were investigated using FEM in [81]–[87]. Improved Galerkin method based FEM, combination of FEM with the method of characteristics or the finite volume method, and other numerical techniques and iterative strategies were proposed to make the calculation more stable and more efficient in [88]–[93]. Recently 3-D FEM was also explored to solve the ionized field in the presence of human bodies and buildings in [73], [94]–[96].

Nevertheless, it's generally highly acknowledged that FEM is CPU and memory intensive, particularly for those cases where a large number of discretized nodes and repeat calculations are necessary. Nowadays high-performance parallel computing is being explored to speed up iterative linear solvers, among which the conjugate gradient (CG) solver is most widely employed. Based on [97], the Jacobi method has advantages over CG method when applied in MapReduce framework, which is based on a single instruction stream multiple data stream (SIMD) paradigm. However, the prerequisite of Jacobi method that the system of linear equations should be diagonally dominant restricted its use in a FEM solver. Besides, graphics processors (GPUs) as another commonly used SIMD paradigm are limited by the device memory for large-scale FEM problems [26]. GPUs have been exploited for the simulation of large-scale power systems in several areas including transient stability simulation, electromagnetic transient simulation, and dynamic state estimation [98]–[101].

In this chapter, a novel finite difference relaxation (FDR) method is proposed to solve a unipolar and a bipolar conductor-to-ground problem without Deutsch's assumption. This method requires much less memory than FEM because the finite-difference equations need not be assembled, and FDR can be massively parallelized in the GPU. The parallel implementations are carried out on multi-core CPU and many-core GPU, and the results are compared with those obtained from commercial FEM software Comsol Multiphysics^{TM} concerning accuracy and computational efficiency. Additionally, for the bipolar case, the current continuity equations are regarded as PDEs on electric potential rather than on ion density, and the solution process using the FDR scheme is unconditionally stable.

This chapter is organized as follows. Section 5.2 presents the assumptions, the governing equations, the boundary conditions, and the iterative scheme. Section 5.3 provides the implementation details of the FDR method with differentiated grid size. Section 5.4 gives

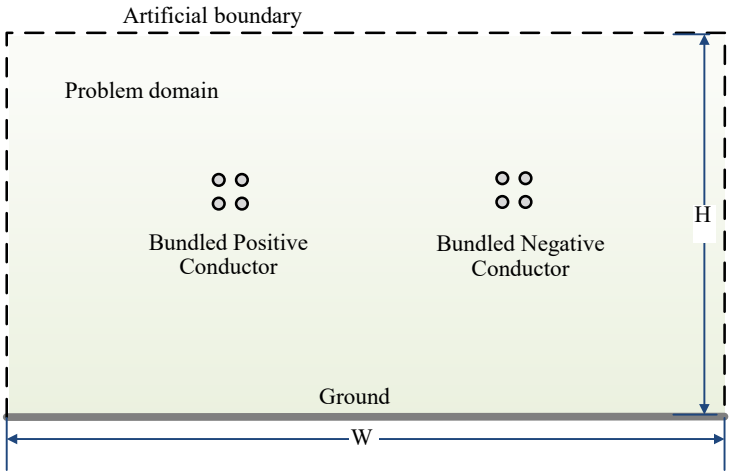


Figure 5.1: Computational domain for bipolar conductor-to-ground arrangement.

the unipolar and bipolar case studies and the result comparison of the FDR method and the FEM. Finally, Section 5.5 presents the summary.

5.2 Problem Description

5.2.1 Assumptions for Modeling

Since corona and the ionized field are complicated physical phenomena, appropriate assumptions are necessary to build a solvable mathematical model. The assumptions employed in this work are the following:

1. The ionized field is time-independent. All parameters involved do not vary along the direction of the transmission line, i.e. the problem is two-dimensional.
2. The thickness of the ionization layer around the conductor is so small that it can be neglected.
3. Ionic mobility is constant, and ion diffusion is neglected.
4. The electric potential of those nodes on the artificial boundary is determined by the space charge free field, which is defined by Laplacian equations.

The 2-D ionized field problem is a boundary value problem (BVP), and the problem domain can be illustrated as in Fig. 5.1. The rectangular domain may contain bipolar bundled conductors or a single centered conductor for different cases.

5.2.2 Governing Equations

The governing equations of the bipolar ionized field are [5]

$$\nabla^2 \cdot \varphi = \frac{\rho^- - \rho^+}{\epsilon_0}, \tag{5.1}$$

$$\nabla \cdot (k^+ \rho^+ \nabla \varphi) = \frac{R\rho^+\rho^-}{e}, \tag{5.2}$$

$$\nabla \cdot (k^- \rho^- \nabla \varphi) = \frac{R\rho^+\rho^-}{e}, \tag{5.3}$$

where ρ is the space charge density, ϵ_0 the permittivity of free space, φ the electric potential, k the ionic mobility, R is the recombination rate, e the charge of the electron, and the superscript + for positive and - for negative.

The three unknowns to be solved in the bipolar ionized field problem are electric potential φ and the space charge densities ρ^+ and ρ^-.

In the case of unipolar ionized fields, all the space charges have the same polarity as the conductor. The governing equations of the unipolar ionized field with only positive conductor can be obtained by forcing the negative space charge density to 0. The unknowns to be solved are electric potential φ and the positive space charge density ρ^+. Equations (5.1)-(5.3) are reduced to the following:

$$\nabla^2 \cdot \varphi = -\frac{\rho^+}{\epsilon_0}, \tag{5.4}$$

$$\nabla \cdot (\rho^+ \nabla \varphi) = 0. \tag{5.5}$$

5.2.3 Boundary Conditions

The product of two unknowns in the current continuity equation makes the problem nonlinear. The solution of the BVP can be obtained if the boundary conditions (BCs) are well-posed, i.e., neither undetermined nor overdetermined. Substituting (5.4) to (5.5) yields the following equation:

$$\nabla \cdot ((-\epsilon_0 \nabla^2 \cdot \varphi) \nabla \varphi) = 0. \tag{5.6}$$

Equation (5.6) is a nonlinear third-order PDE on φ, and three boundary conditions on φ are required:

1. The electric potential on the conductor is the applied voltage V_0 (Dirichlet type):

$$\varphi_C = V_0. \tag{5.7}$$

2. The ground is taken as the reference (Dirichlet type):

$$\varphi_G = 0. \tag{5.8}$$

3. The third boundary condition is usually selected from the following two conditions. Only one can be applied or the problem will be overdetermined.

(a) The electric field strength at the conductor surface is constant at the corona onset value (Kaptzov's assumption, Neumann type):

$$\frac{\partial \varphi}{\partial n} = E_0. \tag{5.9}$$

(b) The charge density at the conductor surface is known as ρ_0 from experimental results:

$$\rho_C = \rho_0. \tag{5.10}$$

Also, truncation of the domain is necessary. The above boundary conditions together with the fourth assumption mentioned previously can define the problem well.

5.2.4 Predictor-Corrector Strategy

For the unipolar case, the problem could not be resolved directly because the governing PDEs containing the two unknowns are coupled. The predictor-corrector algorithm proposed by [5] and [80] is based on the fact that both (5.4) and (5.5) can be solved once one unknown is assigned an initially predicted distribution. Then the predicted distribution is corrected progressively by iteratively solving the two PDEs. One most common and simple iterative procedures in [5] can be described as in Fig. 5.2. The iterative strategy used

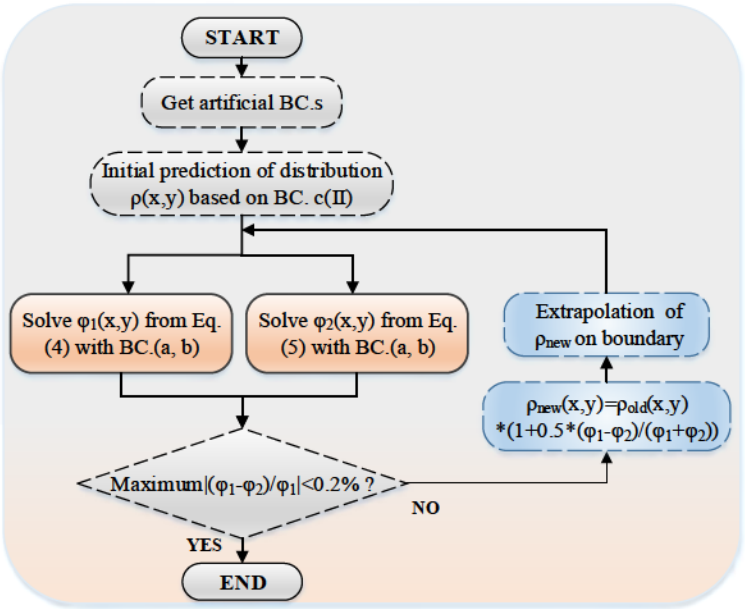


Figure 5.2: Flow chart of the predictor-corrector algorithm.

may be a bit different yet the same solution can be achieved. For example, in [80], $\varphi_1(x, y)$ solved from (5.4) was sequentially utilized by (5.5) to generate a new $\rho(x, y)$. In this work, the flow chart shown in Fig. 5.2 is employed for the unipolar case.

For the bipolar case, the iterative strategy to handle φ, ρ^+ and ρ^- is very similar. The modified iterative strategy for the three unknowns is presented in Section 5.5.

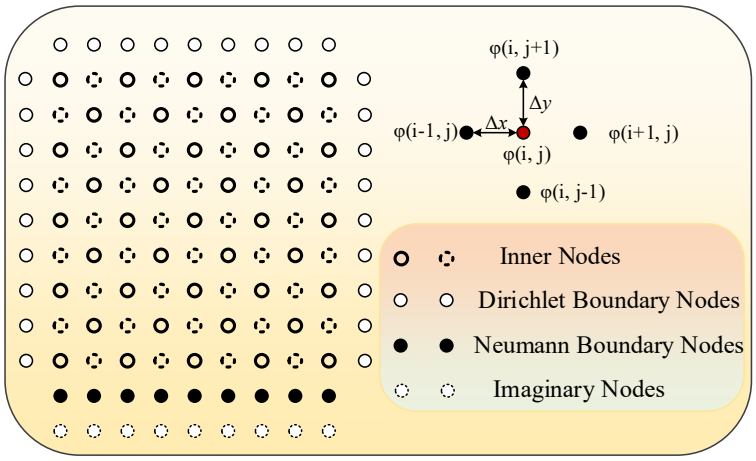


Figure 5.3: Domain discretization for 2-D boundary value problem.

5.3 Finite-Difference Relaxation Methodology

5.3.1 Domain Discretization and FDR

The weighted residual method in FEM is widely utilized instead of tediously constructing a functional based on the variational principle. The product of residual and weighted function is integrated over the domain of each element, and a set of equations associated with nodes of each element can be obtained by forcing the integration to be zero. Then these equations are assembled and solved based on certain boundary conditions.

On the contrary, the finite-difference method is convenient when forming equations by replacing derivative with difference quotient in classic formulation; at the same time, a regular grid is required. Figure 5.3 shows a simple 2-D discretized domain. The order of the PDE determines the number of nodes required for difference equation derivation. For a second order PDE, the five-node mode is sufficient. As shown in Fig. 5.3, every inner node is surrounded by four nodes (either inner or boundary nodes). A very important observation in [5] is that both the Poisson's equation and the current continuity equation can be rewritten as the following second-order PDE on φ:

$$\nabla \cdot (\alpha \nabla \varphi) = \beta. \tag{5.11}$$

Equations (5.1)-(5.5) can be obtained by setting different α and β. For example, (5.1) in 2-D domain can be rewritten as the following form when α=1 and β=$(\rho^- - \rho^+)/\epsilon_0$:

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = \frac{\rho^- - \rho^+}{\epsilon_0}. \tag{5.12}$$

The second derivative in (5.12) can be replaced with a difference equation at node (i, j) using central difference scheme in the five-nodes mode as:

$$\frac{\partial^2 \varphi}{\partial x^2} = \frac{\varphi(i-1, j) + \varphi(i+1, j) - 2\varphi(i, j)}{(\Delta x)^2}, \tag{5.13}$$

$$\frac{\partial^2\varphi}{\partial y^2} = \frac{\varphi(i,j-1)+\varphi(i,j+1)-2\varphi(i,j)}{(\Delta y)^2}. \tag{5.14}$$

Thus Poisson's equation (5.1) at node (i,j) can be written as:

$$\begin{aligned}\varphi(i,j) =& \frac{(\Delta x)^2(\Delta y)^2}{2((\Delta x)^2+(\Delta y)^2)} \cdot (\frac{\varphi(i-1,j)+\varphi(i+1,j)}{(\Delta x)^2} \\ &+ \frac{\varphi(i,j-1)+\varphi(i,j+1)}{(\Delta y)^2} + \frac{\rho^+(i,j)-\rho^-(i,j)}{\epsilon_0}).\end{aligned} \tag{5.15}$$

Similarly, setting α=ρ^+ and β=0, the current continuity equation (5.5) can be rewritten as:

$$\begin{aligned}\varphi(i,j) =& \frac{(\Delta x)^2(\Delta y)^2}{2((\Delta x)^2+(\Delta y)^2)} \cdot (\frac{\varphi(i-1,j)+\varphi(i+1,j)}{(\Delta x)^2} \\ &+ \frac{\varphi(i,j-1)+\varphi(i,j+1)}{(\Delta y)^2} + \frac{\rho(i,j)}{\epsilon_0} \\ &+ \frac{(\rho^+(i+1,j)-\rho^+(i-1,j))(\varphi(i+1,j)-\varphi(i-1,j))}{4(\Delta x)^2} \\ &+ \frac{(\rho^+(i,j+1)-\rho^+(i,j-1))(\varphi(i,j+1)-\varphi(i,j-1))}{4(\Delta y)^2}).\end{aligned} \tag{5.16}$$

For those nodes on the boundary, the BCs can be either Dirichlet type or Neumann type. Nodes located exactly on regular boundary (such as a line) are straightforward while nodes near the circular conductor need special attention. For example, in this work, those nodes satisfying the following conditions define the approximated conductor surface (Fig. 5.4(b)):

$$r < d < r + \sqrt{(\Delta x)^2+(\Delta y)^2} \tag{5.17}$$

where d is the distance between a node and the conductor center, r the conductor radius, and Δx, Δy are the spatial increment. The approximation is more accurate when the grid layer is finer. The values of the nodes on Dirichlet boundary are fixed and only used as known value when updating the adjacent inner nodes. Thus the finite difference equation applies only for inner nodes and no equations are necessary for boundary nodes. For nodes on Neumann boundary, the equation can be written with the help of imaginary nodes, which are associated with inner nodes and $\frac{\partial\varphi}{\partial n}$.

By writing the difference equation for each node, a set of linear equations can also be obtained as in FEM. In FEM assembling all the element equations often produces a large matrix which calls for more memory, although the sparsity of the matrix may enable saving of memory by exploring special data storage methods. Undoubtedly, the tedious tasks undermine the prospect for massively parallel computation. The solution phase of FDR is quite different.

Indeed calculating one node based on the adjacent 4 nodes by (5.15) can be regarded as a form of communication or information exchange. At first glance, a single communication between one node and its neighbors is probably meaningless because it never knows

whether the neighbors are of the desired solutions or not. However, if all inner nodes update themselves repeatedly, these communications can be very beneficial to find the final solution. Intuitively, the prescribed value such as information on the boundary will gradually flow into the entire domain by iteratively updating each node. Convergence can be expected when all values of inner nodes satisfy (5.15) if the problem is well-posed. Each node is concerned with its computation based on the finite-difference equations and eventually a converged solution satisfying all nodes can be achieved. Thus, the process is called *finite-difference relaxation*. The following section will reveal that the convergence of the FDR scheme is a mathematical certainty.

5.3.2 Jacobi method and Convergence Condition

There are two classes of algorithms for solving a linear system of equations. Direct methods like Gauss elimination, or equivalently LU factorization followed by back-substitution can provide the exact solution after a finite sequence of operations. Iterative methods such as the conjugate gradient method and the Jacobi method are commonly used as they provide solutions for desired error tolerances for a large-scale linear system. Indeed, for Poisson's equation, the set of finite-difference equations can be written as the general form:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{5.18}$$

where $\boldsymbol{A}$ is a known sparse matrix associated with the coefficients of each unknown nodes and its neighbors indicated in (5.15), $\boldsymbol{x}$ the unknown vector containing $\varphi(i,j)$ of each unknown node (boundary node excluded) and $\boldsymbol{b}$ the known vector related to $(\rho^-(i,j) - \rho^+(i,j))/\epsilon_0$ and prescribed boundary values.

Note that most unknown nodes are surrounded by four other unknown nodes, and the nodes right adjacent to the boundary nodes have only two or three unknown neighbors.

Solution of the system can be obtained by the iterative expression:

$$\boldsymbol{P}\boldsymbol{x}^{k+1} = (\boldsymbol{P} - \boldsymbol{A})\boldsymbol{x}^k + \boldsymbol{b}, \tag{5.19}$$

where $\boldsymbol{P}$ is the preconditioner and $\boldsymbol{x}^k$ is the k^{th} approximation of $\boldsymbol{x}$.

For the Jacobi method, the preconditioner $\boldsymbol{P}$ is set the diagonal matrix of $\boldsymbol{A}$. Indeed, $\boldsymbol{A}$ can be decomposed into a diagonal matrix $\boldsymbol{D}$, and the remainder $\boldsymbol{R}$. Thus the iteration formula for Jacobi method can be written as:

$$\boldsymbol{x}^{k+1} = -\boldsymbol{D}^{-1}\boldsymbol{R}\boldsymbol{x}^k + \boldsymbol{D}^{-1}\boldsymbol{b}. \tag{5.20}$$

The standard convergence condition is when the spectral radius of the iteration matrix is less than 1 [102], namely

$$\rho(\boldsymbol{D}^{-1}\boldsymbol{R}) < 1. \tag{5.21}$$

It is acknowledged that (5.21) is sufficiently satisfied if the matrix is diagonally dominant, i.e., the absolute value of the diagonal term is greater than the sum of absolute values of other terms:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|. \tag{5.22}$$

It can be observed from (5.15) that the coefficient of each to-be-updated inner node is equal to the summation of coefficients of its four unknown neighbors, and the coefficient of those to-be-updated nodes right adjacent to the boundary nodes is greater than the summation of coefficients of its two or three unknown neighbors. Thus, for most rows in coefficient matrix $\boldsymbol{A}$, the following equation is satisfied:

$$|a_{ii}| = \sum_{j \neq i} |a_{ij}|. \tag{5.23}$$

If (5.23) is satisfied for all rows of $\boldsymbol{A}$, the spectral radius is equal to 1. However, for those nodes next to the known boundary nodes, (5.22) is well satisfied for the corresponding row of $\boldsymbol{A}$ and determine that the spectral radius is less than 1.

5.3.3 Differentiated Grid Size

Few attempts using finite-difference method have been reported in the literature due to the inflexibility when handling irregular geometries and disproportional sizes. For example, in the ionized field problem handled by FEM, the mesh size near the conductor is fine enough to ensure accuracy while a relatively coarse mesh is applied for the rest of the vast domain to save computational resources. A square grid is the basis of FDR, and it is one drawback compared with FEM. To depict the contour of the thin conductor, high node density (or fine grid) is needed around the conductor. And if the whole domain is filled with these dense nodes, the merits of FDR will be impaired seriously. Whereas differentiated grid sizes can be explored and the details are described below.

Fig. 5.4(a) shows a simplified scheme for applying two layers with differentiated grid sizes. Figure 5.4(b) shows irregular geometries can be described accurately if the grid is fine enough. Undoubtedly different mesh sizes will cause the communication problem on the boundary separating the two layers. As shown in Fig. 5.4(c), updating the outermost nodes of the fine grid will require the value of some non-existing nodes next to them, which are the dashed nodes located out of the fine grid layer. However, even though these dashed nodes are imaginary, their values can be predicted as they are located in a cell (a square consisting of four adjacent nodes) in the coarse grid. Therefore, an interpolation technique similar with that of FEM is employed to predict the desired value based on the nodes in the coarse grid layer. Figure 5.4(d) shows a local coordinate system. Assume the electric potential at nodes A, B, C and D are φ_A, φ_B, φ_C and φ_D. For any coordinate (ξ, η),

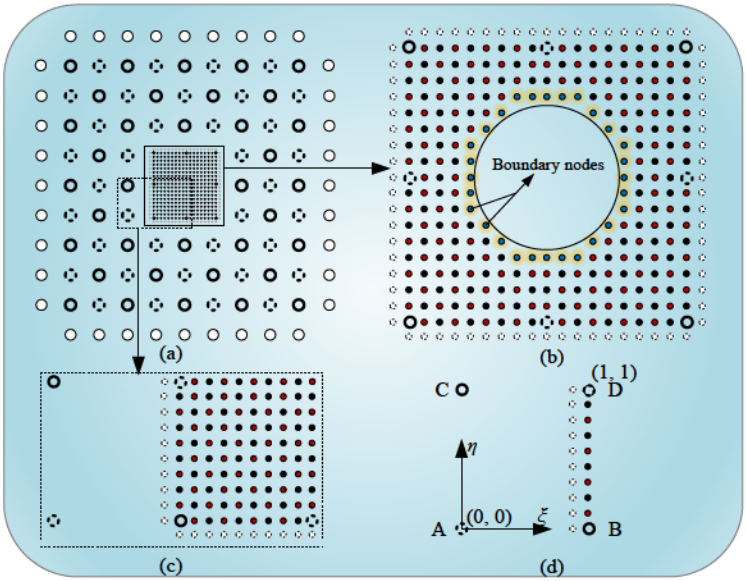


Figure 5.4: Differentiated grid size and interpolation in FDR.

the contribution (shape function) of each node N_A, N_B, N_C and N_D can be written as:

$$\begin{cases} N_A = (\xi - 1)(\eta - 1), \\ N_B = \xi(1 - \eta), \\ N_C = \eta(1 - \xi), \\ N_D = \xi\eta. \end{cases} \tag{5.24}$$

Thus the interpolated value at any coordinate (ξ, η) can be predicted as:

$$\varphi(\xi, \eta) = N_A\varphi_A + N_B\varphi_B + N_C\varphi_C + N_D\varphi_D. \tag{5.25}$$

By interpolation, the fine grid layer can obtain information from the coarse grid layer. It should be noted that after all nodes are updated, a similar process called retrieval is necessary so that the coarse grid layer can obtain information from the fine grid layer. The detailed process and the according node type are described in the flow chart in Fig. 5.5. Thus, for each iteration, the following three phases are always necessary for differentiated grid sizes:

1. Interpolation: information flows from the coarse grid layer to the fine grid layer.
2. Updating: each node is updated based on its neighbors.
3. Retrieval: information flows from the fine grid layer back to the coarse grid layer.

5.4 Massively Parallel Implementation

5.4.1 Data Dependency and Parallelism

As discussed above, once $\boldsymbol{x}^k$ is available, $\boldsymbol{x}^{k+1}$ can be obtained. Though nodes can be calculated independently, synchronization is necessary between iterations to avoid updating one node with its yet-to-be-updated neighbors multiple times. Additionally, the interpolation and retrieval phase can also be parallelized in each iteration, yet synchronization is also needed between these phases.

In the FDR process, two matrices are needed for each variable to store the node values of both coarse grid layer and fine grid layer. To elaborate the process flowchart clearly, different node types are classified based on the location (shown in Fig. 5.5(a)). For example, type A, a, B and b are the solid and dashed inner nodes in the two layers. Type C and c are boundary nodes while type D and d represent the nodes to be updated in the retrieval phase. The main flowchart is shown in Fig. 5.5(c).

5.4.2 Parallelization on CPU and GPU

The FDR program is parallelized on both multi-core CPU and many-core GPU.

For CPU parallelization, multiple threads are launched by the master thread and handle different parts of the tasks. Open multi-processing (OpenMP™) and POSIX Threads (Pthread™) are commonly used application programming interfaces (API) for shared memory multiprocessor programming. OpenMP™ is relatively higher level, and thus easier to use. The high-level feature results in inflexibility because it is difficult to control each thread. Moreover, for the FDR iterative scheme, repeatedly launching and joining threads between iterations greatly increases overhead for OpenMP™. On the contrary, Pthread™ is a lower level API that takes extremely fine-grained control over threads. Each thread is launched and will not be joined until the iteration process ends. Therefore, Pthread™ is utilized to implement the CPU parallelization in this work. Considering the number of cores available in CPU, the row-wise parallel implementation is employed, and the flowchart of the thread function is described in Fig. 5.5(b).

Indeed, massive parallelization is perfectly suitable for the updating phase because node calculations do not depend on each other. CUDA is chosen for massive-thread parallel programming on the GPU. A CUDA program separates the hardware resources into CPU side (host) and GPU side (device). There is no shared memory for two sides and thus copy operations are necessary for data exchange.

In the CPU parallelization, each variable needs a copy operation to store the updated value. A more efficient strategy is explored for GPU implementation to reduce the required memory and accelerate the convergence. Indeed, it can be observed that for all inner nodes in Fig. 5.4, all solid nodes are surrounded by four dashed nodes and vice versa. Thus the calculation phase can be separated into two steps: calculating all solid nodes in parallel,

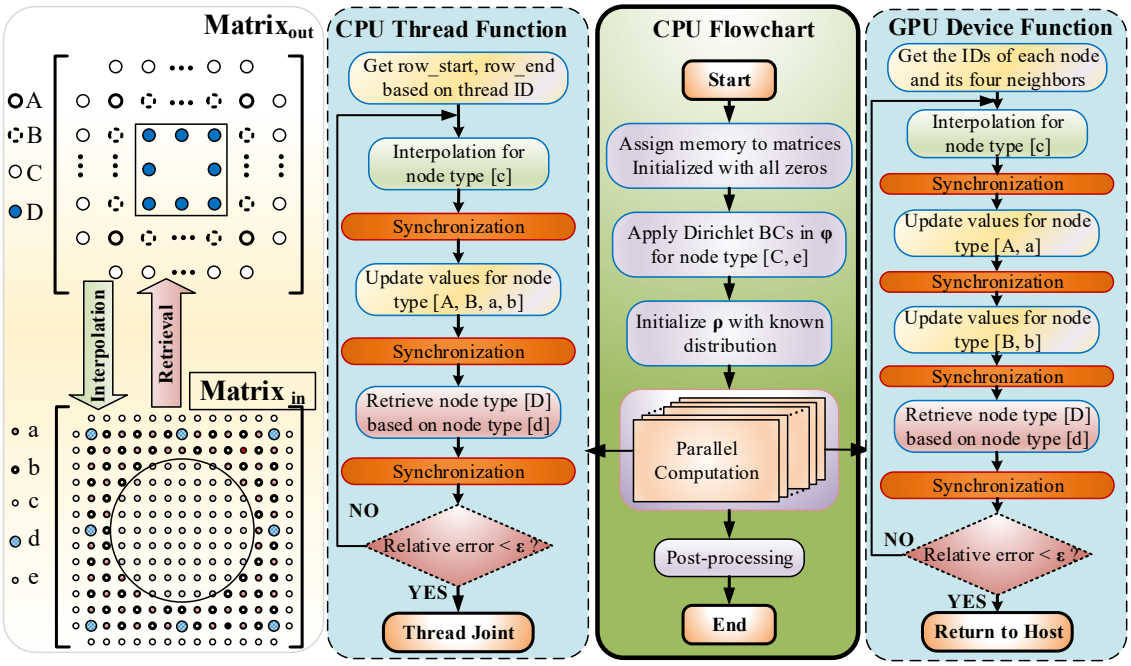


Figure 5.5: Massively parallel implementation of FDR scheme on CPU and GPU.

and sequentially updating all the dashed nodes in parallel. In other words, the updated values of the solid nodes are utilized by updating the dashed nodes within one iteration. It is applicable for both the coarse grid layer and the fine grid layer. This scheme is similar to the Gauss-Seidel method, and the convergence is faster than that of the Jacobi method. The flowchart of the device function is shown in Fig. 5.5(d). The detailed parameters of the CPU and GPU are described in Appendix B.

5.5 Case Study and Results Comparison

5.5.1 Unipolar Case Study

5.5.1.1 Result Comparison of FDR vs FEM

Both Poisson's equation (5.4) and current continuity equation (5.5) can be solved with FDR if the distribution of space charge is provided. On the other hand, the problem can also be solved with the equation-based modeling in Comsol Multiphysics^{TM}. Poisson's equation is chosen in the case study to comprehensively compare the FDR method and the FEM. The parameters are shown as belows:

The domain width is 2m, the domain height 2m, the conductor radius 0.005m, the applied voltage on conductor 20kV, and the space charge density on conductor 2e-6C/m^2. The initial distribution of space charge density is guessed as $10^{-8}/\sqrt{(x-1)^2+(y-1)^2}$C/m^2. The sample line is defined by two points: (0, 0) and (1, 1).

To obtain reliable results from FEM, mesh dependency test is done on a sample point. It turned out that the FEM results can be regarded as stable and reliable when the number of nodes is more than 1500 in the case. Thus the FEM results are assumed correct and can serve as a benchmark for evaluating the accuracy of FDR. Since the problem domain is 2-D,

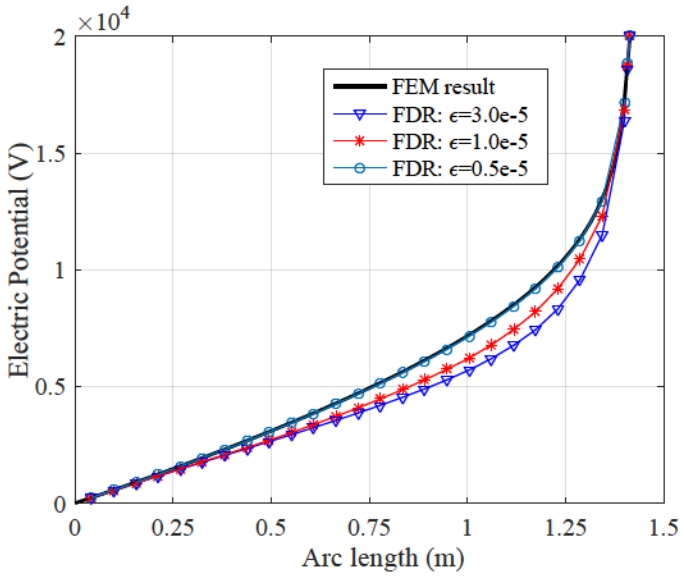


Figure 5.6: Electric potential comparison of FEM and FDR for unipolar case.

the sample line shown in Fig. 5.1 is selected to plot the results. The electric potential along the selected path solved from both FEM and FDR is shown in Fig. 5.6. When the number of nodes is 10,000, the maximum relative difference between the FEM and the FDR method is around 3%.

The results of FDR converge to the results of FEM when the maximum relative error ϵ between iterations decreases. As an iterative method, the solution phase of FDR can be described by the ϵ-iteration curve shown in Fig. 5.7. The convergence speed is determined by the spectral radius mentioned above; convergence is faster when the spectral radius is small. When the problem size increases, the percentage of boundary nodes decreases, and the spectral radius comes closer to 1. However, to the author's knowledge, it is difficult to quantify the relationship of node number, spectral radius, and ϵ. It was found from experience that the iteration can be deemed convergent when ϵ is less than 10^{-6} if the number of nodes is less than 50,000.

5.5.1.2 Accuracy and Efficiency Comparison of FDR vs FEM

A comprehensive comparison of FDR and FEM concerning computation time and accuracy is presented in Table I. For the CPU parallel implementation utilizing 16 processor cores, the maximum speed-up is greater than 14 under different node numbers. For GPU parallel implementation, the speedup is 30 times. As shown in the last column of Table 5.1,

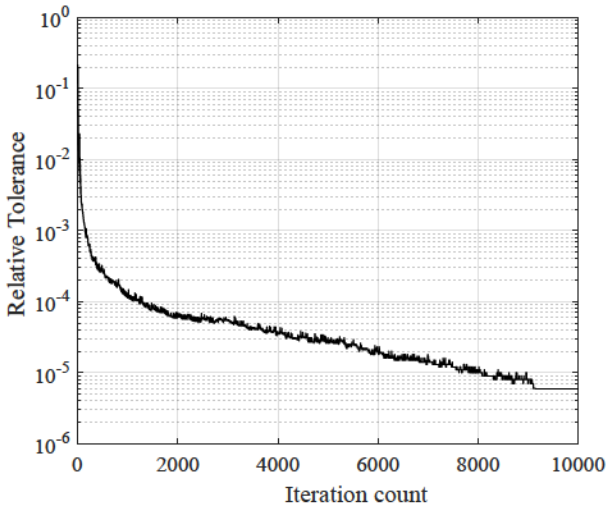


Figure 5.7: Relative error vs iteration number for the FDR method.

Table 5.1: Efficiency and accuracy comparison of proposed FDR method with FEM

Node number	FEM solution time (s) Comsol Multiphysics™	FDR solution time (s) and speed-up							Relative error
		Multi-core CPU					Many-core GPU		
		Solution time - thread count				Speed-up	Solution time	Speed-up	
		1	4	8	16				
3600	0.9	0.182	0.065	0.050	0.050	18	0.028	32.1	4.63%
10,000	1.78	0.700	0.210	0.140	0.124	14.4	0.05	35.6	3.15%
40,000	5.56	2.960	0.816	0.512	0.38	14.6	0.18	30.8	1.68%

the results of FDR come closer to the correct solution as the number of nodes increases.

Note that the built-in direct solver of Comsol Multiphysics™ is applied, which turned out to be faster than its iterative solver for the cases presented in the table. For example, the iterative solver consumed 7.1s while the direct solver consumed 5.56s when the node number is 40,000. Thus the speedup is concerning the execution time of the commercial software Comsol Multiphysics™, which can be regarded as highly optimized and sufficiently efficient.

Similarly, the current continuity equation was simulated with the proposed FDR scheme. Applying the iterative strategy in Fig. 5.2, the final solution was obtained. The initial distribution of $\rho(x, y)$ and the solved $\varphi_1(x, y)$ and $\varphi_2(x, y)$ are presented in the first row of Fig. 5.8. When the iteration converges, the final solution of $\rho(x, y)$, $\varphi_1(x, y)$ and $\varphi_2(x, y)$ is shown in the second row of Fig. 5.8.

5.5.2 Practical Bipolar Case Study

5.5.2.1 Application of FDR

In practical HVDC applications, bipolar bundled conductors are usually utilized for power transmission. The case is more complicated, whereas the computation can still benefit from the merits of the FDR scheme. The following section will elaborate how the FDR scheme

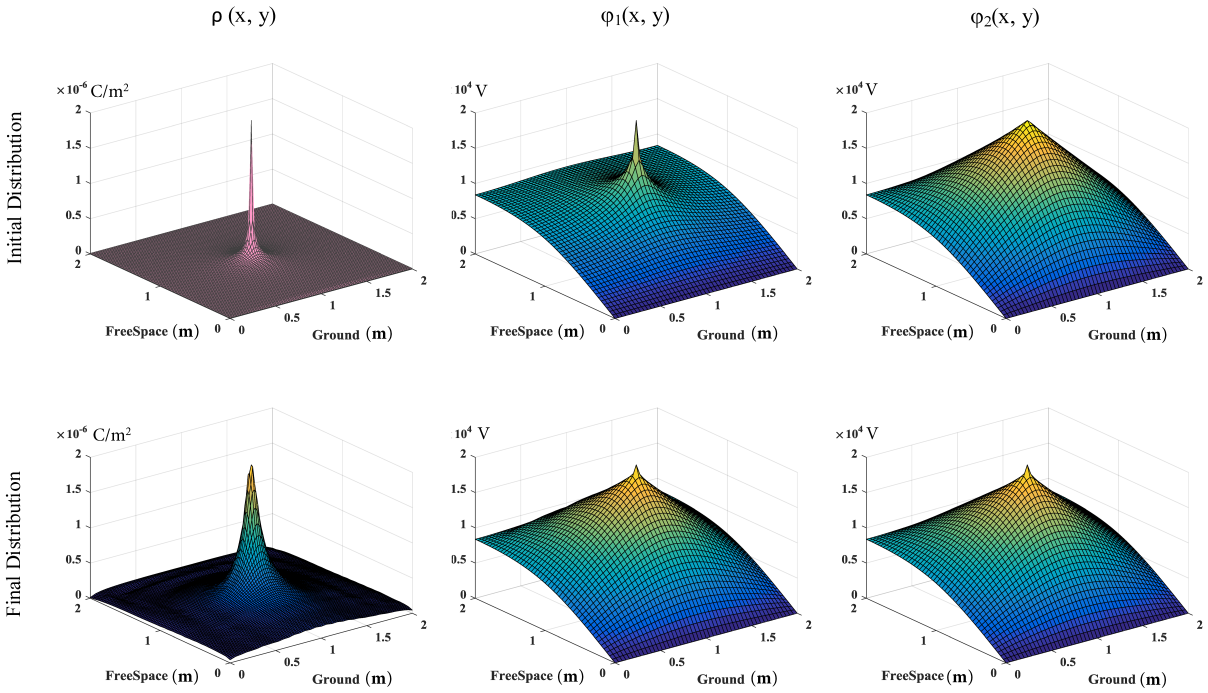


Figure 5.8: Final converged solution of the unipolar ionized field attained from the proposed FDR method.

is applied for the full-scale bipolar bundled conductors. Figure 5.9 shows the structure of a typical ±500 kV HVDC lattice tower of the Eastern Alberta HVDC line built by ATCO Electric Ltd., and the geometric parameters are available online [103].

For the bipolar case, space is filled with ions of both polarities. The ions of both polarities migrate to the ground, at the same time, they migrate to the conductors with the opposite polarity.

In each iteration of the solution phase, the φ in (5.1) is solved with the guessed or updated ρ^+ and ρ^-; and then the ρ^+ in (5.2) and the ρ^- in (5.3) are solved respectively based on the obtained φ in (5.1). However, the solution process of the current continuity equations is likely to become unstable because of the accumulated error of first-order derivative. Thus to counter instability, it calls for numerical techniques like the upwind scheme in [80]. The current continuity equation in (5.2) and (5.3) can be seen as a first order PDE on ρ (either ρ^+ or ρ^-), and it can also be regarded as a second order PDE on φ. Mathematically, the stability of a first order PDE on u is conditional and depends on the coefficients of u, u_x and u_y. However, the second order PDE on φ in (5.11) can be solved with the FDR scheme efficiently if the distributions of α and β are given (either guessed value or constant) and the solution of the FDR scheme is unconditionally stable. That is why the numerical stability issue is not a concern in [5] as well as in the unipolar case studied above. Since solving ρ^+ based on known φ is unstable while solving φ based on known ρ^+ is unconditionally stable, the iterative strategy in Fig. 5.2 is improved to solve

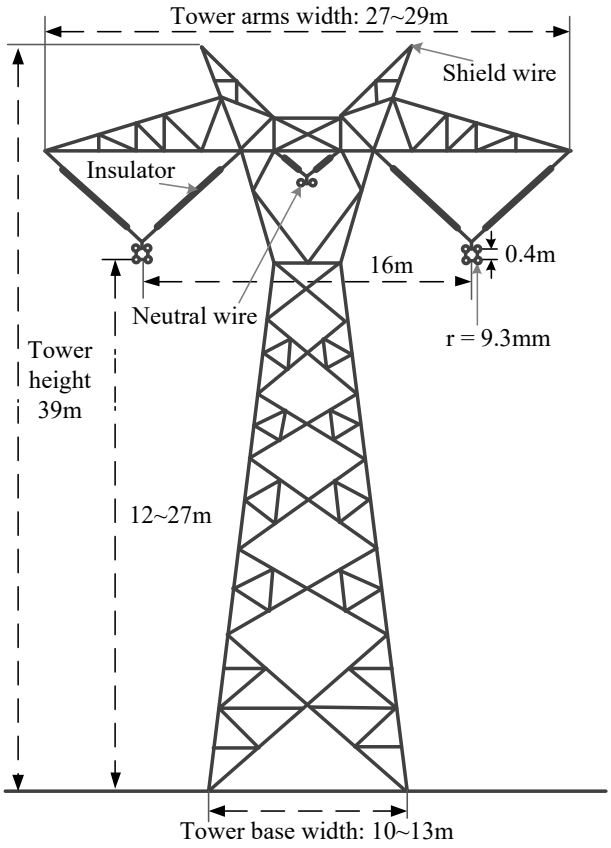


Figure 5.9: Structure of $\pm$500 kV DC lattice tower of the Eastern Alberta HVDC Line.

the bipolar case to avoid numerical instability.

The improved iteration process can be described with the following steps:

1. Initial estimate of space charge ρ^+ and ρ^- are provided based on boundary conditions.

2. Solve φ in (5.1), (5.2) and (5.3) respectively with the FDR scheme based on the known ρ^+ and ρ^-. The results are stored as φ, φ^+ and φ^-.

3. Update ρ^+ based on $\varphi - \varphi^+$ and ρ^- based on $\varphi - \varphi^-$.

4. Go to Step 2 with the updated ρ^+ and ρ^-. The process is repeated until the maximum relative error between φ^+ and φ^- is smaller than the prescribed value ε.

It is worth mentioning that solving (5.11) with the FDR scheme is no different for either the unipolar case or the bipolar case. For the case with multiple conductors, multiple Dirichlet boundary conditions should be applied. Similarly, differentiated grid sizes are employed in consideration of the thin conductor in a vast space domain. As shown in Fig. 5.10, the fine grid layer is applied around each conductor and the coarse grid layer

is applied for the rest domain. The nodes nearest to the conductor in the fine grid layer are defined as Dirichlet boundary nodes and the values are fixed in the FDR scheme. The node updating (communication) pattern stays the same, and both the interpolation and retrieval phase should be applied to every boundary that separates the coarse grid layer and the fine grid layer.

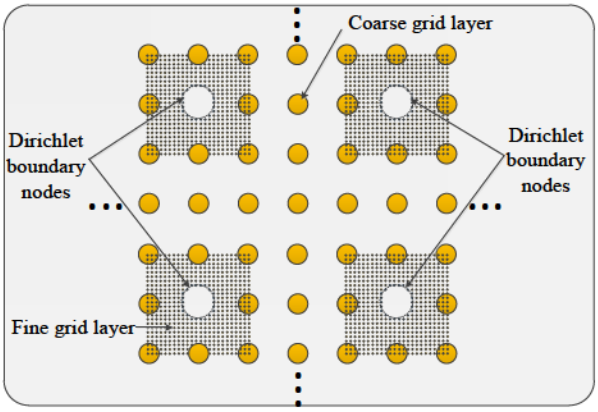


Figure 5.10: Differentiated grid layer for multiple Dirichlet boundaries of the 4-conductor bundle.

5.5.2.2 Results and Discussion

According to the structure shown in Fig. 5.9, the minimum height of the transmission line is 12m at the mid-span between towers. Thus the computational domain is obtained by spatial truncation. The width of the truncated problem domain is 80m and the height 25m. The geometric parameters are shown in Fig. 5.9 and other necessary parameters are presented as belows:

The applied voltage on conductor is $\pm$500 kV, the permittivity of free space 8.854e-12F/m, the positive ionic mobility 1.4e-4m^2/V·s, the negative ionic mobility 1.8e-4m^2/V·s, the recombination rate 2e-12m^2/s^2, the charge of electron 1.602e-19C, and the space charge density on each conductor 2e-6C/m^2. The sample line is defined by two points: (0, 12.5) and (80, 12.5).

Following the iterative steps listed above, the bipolar problem with bundled conductors can be resolved with both the FEM and the FDR scheme. It turned out the FDR scheme can be perfectly applied in problems with multiple Dirichlet boundary conditions. The result comparison of the calculated electric potential along the sample line using FDR and FEM is shown in Fig. 5.11. The electric field strength near the ground is shown in Fig. 5.12 and the contour agrees well with that in [94].

The calculated distribution of the electric potential is shown in the Fig. 5.13.

The distribution of the positive and negative ion density distribution are presented in the Fig. 5.14. The results agree with the physical facts that ions of both polarities migrate

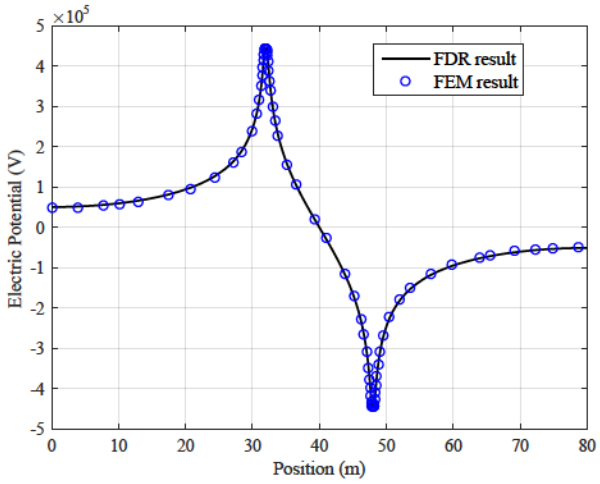


Figure 5.11: Result comparison of electric potential along the sample line.

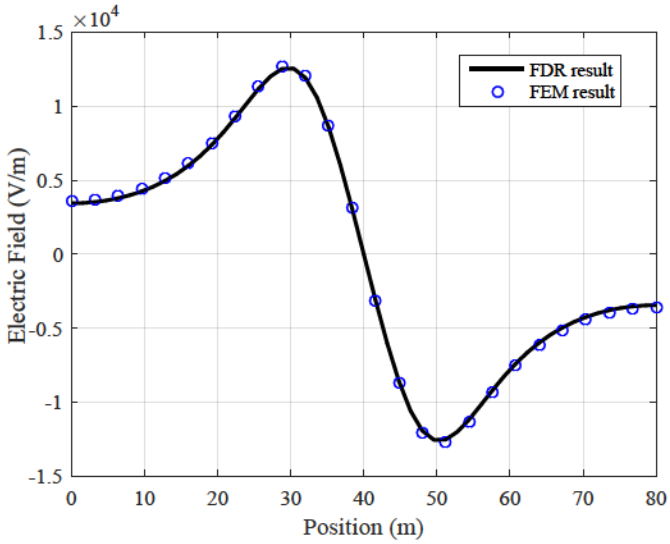


Figure 5.12: Result comparison of electric field distribution on the ground.

to the ground and to the conductor with the opposite polarity.

The speed-up of the bipolar case is similar to the unipolar case and can be inferred from Table I. For both cases, repeatedly solving φ in (5.11) based on α and β is the critical part of the work and consumes most of the computational time. The α and β may vary for different cases, however, the performance improvement of the FDR compared with FEM regarding accuracy and speed-up is independent of α and β provided that the problem is well-posed.

Note that in the bipolar case study, the ion density on the conductor surface is set as the boundary condition. This boundary condition is usually replaced by the Kaptzov's assumption. In that case, the ion density on the conductor should be updated in each iteration based on $E_k - E_0$, where E_k is the calculated electric field strength on the conductor

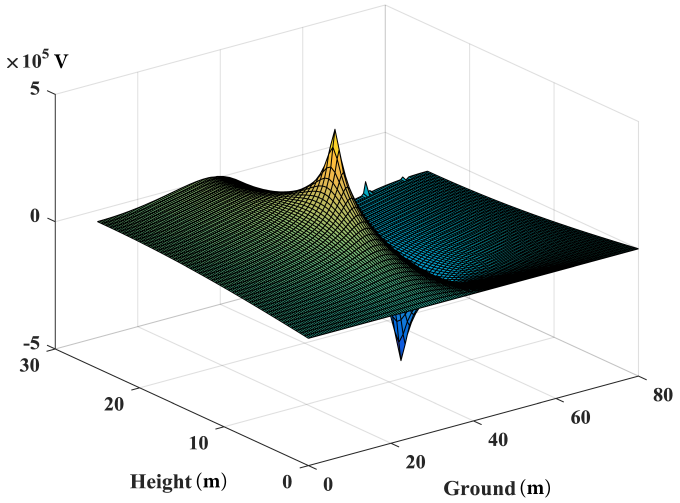


Figure 5.13: Electric potential distribution for the bipolar case.

surface in the k^{th} iteration and E_0 the corona onset value.

5.6 Summary

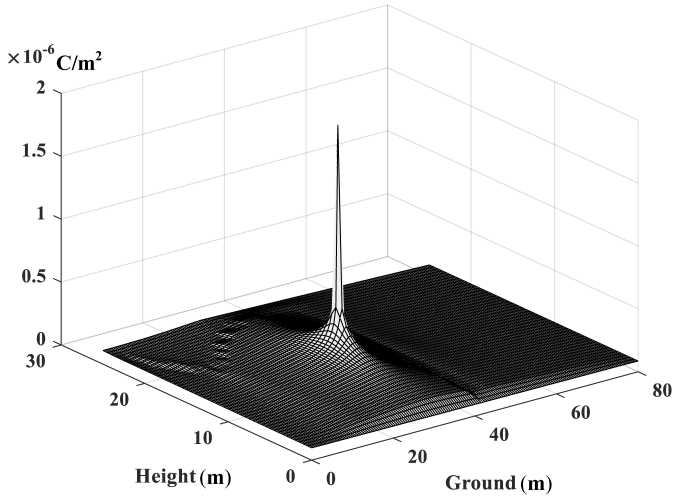
In this chapter, a finite-difference relaxation (FDR) method is proposed for the computation of both unipolar and bipolar ionized fields in HVDC transmission lines. Instead of solving the current continuity equation as a first order PDE on ion density, this work solves it as a second order PDE on electric potential. The numerical instability problem is perfectly solved because the FDR scheme applied for (5.11) is unconditionally stable. The proposed FDR method has the following advantages over the finite element method.

(1) The scheme is suitable for massively parallel computation: compared with the commercial FEM software Comsol Multiphysics^{TM}, the speedup is more than 14 times in CPU parallelization and 35 times in GPU parallel implementation. The maximum relative difference compared with the FEM is around 3%, and acceptable for engineering computation.

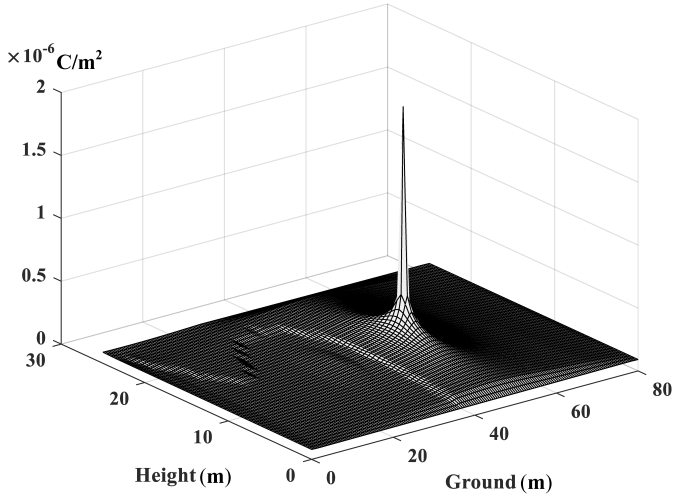
(2) The set of equations in the FDR scheme does not have to be assembled. Instead, it is solved by a relaxation scheme and requires much less memory than FEM. For n nodes, the necessary memory required is $\Theta(n)$ for the FDR method and $\Theta(n^2)$ for the FEM.

(3) Differentiated grid size and interpolation are employed to improve the accuracy and scalability of FDR applied to a vast domain containing a disproportionately thin conductor. Thus FDR can be more flexible when used to handle any domain containing irregular geometries or disproportionate geometry sizes. It is reasonable to conclude that all well-posed second-order PDE having the form of (5.11) can benefit from the proposed FDR scheme instead of FEM concerning computational efficiency, accuracy, and scalability.

It should be mentioned that the results of the proposed parallel method match well



(a) Positive ion density distribution



(b) Negative ion density distribution

Figure 5.14: Ion density distribution for the bipolar case.

with the commercial software regarding the same boundary value problem. Future work will focus on reducing the assumptions and simplifications employed to represent the physical problem accurately.

6 Application for Electromagnetic Transient Studies

6.1 Introduction

Multi-physics simulation software combining finite element analysis (FEA) and electromagnetic transient (EMT) program has gained increasing significance for the design, prototyping, evaluation, and validation of components in high power applications [104]. The device-level modeling enables more accurate and comprehensive information that is unavailable in system-level study, such as the magnetic field distribution revealed by a finite element power transformer model for saturation analysis and eddy current distribution for total loss calculation, the exact performance of a power electronic switching unit and the impacts including junction temperature rise.

However, the notorious computational burden of the finite element solver and the efficiency of the EMT program are usually the main bottleneck, especially in large-scale AC/DC grids with multiple transformers where the field-circuit coupling is involved [105]–[108]. Simulation tools for the analysis of EMT in power systems and power electronic circuits can be classified into two categories: system-level (such as EMTDC/PSCAD®, Simulink, EMTP-RV®) and device level (such as SaberRD®, PSpice®, and Ansys Simplorer®). On the other hand, field-oriented tools such as FLUX CEDRAT®, Comsol® AC/DC and Ansys Maxwell® usually focus on the physical details such as geometries and material properties of electromagnetic devices. For a system-level simulation including FE models, a field-circuit coupling technique is normally required. Although FLUX CEDRAT® and Comsol® AC/DC provide the field-circuit interface, the built-in electrical components in their libraries are very limited and only include basic R, L, and C elements. Co-simulation between Ansys Simplorer® and Ansys Maxwell® is capable of large-scale power elec-

tronic circuit simulation including finite element components [108]–[110], however, the execution time will be prohibitively long to run a large number of time-steps.

Modern high-performance massively parallel architectures such as graphics processing units (GPU) and CPU clusters have been utilized to improve the efficiency of field computation [111]–[115] and power electronics [116], [117], and the key of performance improvement, according to Amdahl's law [13], is determined by the parallelism of the program. Considering the computational load balance and potential to be parallelized, the well-known transmission-line matrix (TLM) modeling, which solves the nonlinearity at the elemental level, has been employed in the solution phase of nonlinear finite element problems [66], [118], to exploit parallelism and improve the computational efficiency.

So far, no research has been reported to study the interaction of the modular multilevel converter (MMC) and FE transformer model, not to mention high-performance massively parallelized codes running on GPU. Under this background, an integrated thermo-electromagnetic model combining the FE transformer model and the MMC is established to study their transient interaction in this chapter. The FE method is utilized to calculate both Ampere's law for magnetics and Fourier's law for heat conduction, and the interfaces between the magnetic field, thermal field, and electrical networks are fully considered. The integrated model can provide detailed field distributions within the transformer and device-level information of the MMC under transient conditions. The codes are massively parallelized for execution on NVIDIA Tesla V100 GPU [12] and the run-time is over 47 times faster than Ansys Simplorer® and Ansys Maxwell® co-simulation.

The contributions of this chapter are listed as follows:

- The transmission line modeling, which solves the nonlinearities at the elemental level and thus is decentralized in nature, is employed to solve the nonlinear finite element equations in a massively parallel manner.
- The electrical network, thermal field, and magnetic field are fully coupled by exchanging coupling coefficients. The thermal impact of winding loss and eddy current loss are considered for the computation of magnetic field and electrical network.
- Fully-detailed modeling of the MMC whose EMT topology is reconfigured using circuit partitioning which separates all the submodules (SMs) from their arms to create a substantial number of independent sub-circuits that caters to parallel processing.
- The interaction of FE transformer model and MMC is implemented with an indirect field-circuit coupling scheme, and the FE transformer is represented by nonlinear self and mutual inductances whose values can be extracted from the FE element computation.
- The massive parallelism of the integrated codes is sufficiently explored to execute on massive parallel GPU architectures.

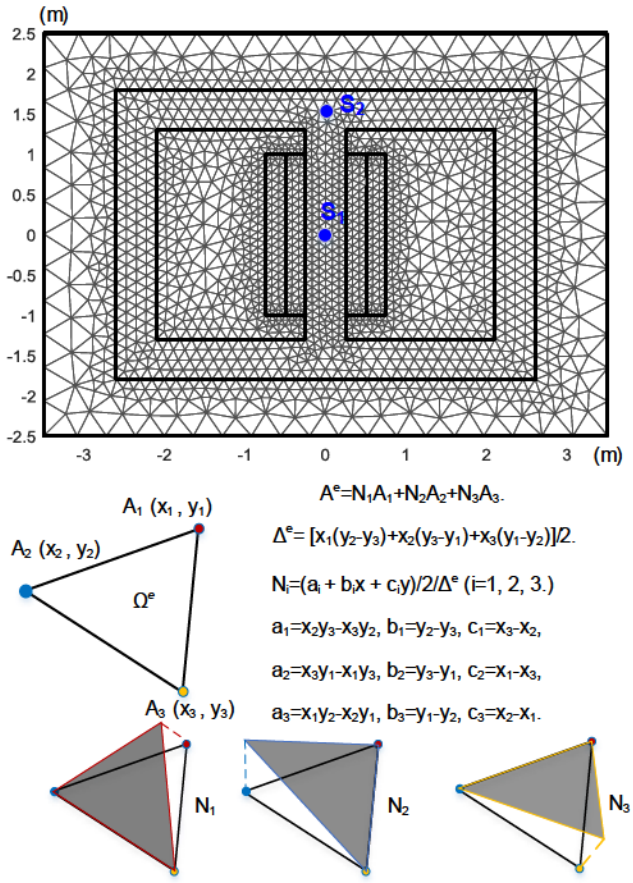


Figure 6.1: 2-D transformer model using the Galerkin finite element meothod.

This chapter is organized as follows: Section 6.2 describes the integrated thermo-electromagnetic transformer model and the coupling techniques in details, and Section 6.3 provides the electro-thermal model of the IGBT in the MMC. Section 6.4 illustrates the parallel implementation of the integrated model on GPU. Then, Section 6.5 gives the case studies and result comparisons while Section 6.6 discussed the parasitic capacitance extraction for high frequency conditions. Finally, Section 6.7 presents the summary.

6.2 Coupled Thermo-Electromagnetic Model of Converter Transformer

6.2.1 Finite Element Model for Magnetic Field

Consider the 2-D finite element model in Fig. 6.1 for a power transformer rated 230kV/110kV and 400MVA. For low and medium frequency application, the magnetic field generated by the winding currents is governed by the Ampere's law with unknown magnetic vector potential:

$$\nabla \cdot (\upsilon \nabla A) = \sigma \frac{\partial A}{\partial t} - J, \tag{6.1}$$

where υ is the field-dependent reluctivity; σ is the electrical conductivity; J is the impressed current density.

After applying the Galerkin finite element method, the elemental equation for each triangular element can be obtained as [118]:

$$\frac{\upsilon^e}{4\Delta^e}\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}\begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} + \frac{\sigma^e \Delta^e}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}\begin{bmatrix} \frac{\partial A_1}{\partial t} \\ \frac{\partial A_2}{\partial t} \\ \frac{\partial A_3}{\partial t} \end{bmatrix} = \frac{J_z^e \Delta^e}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{6.2}$$

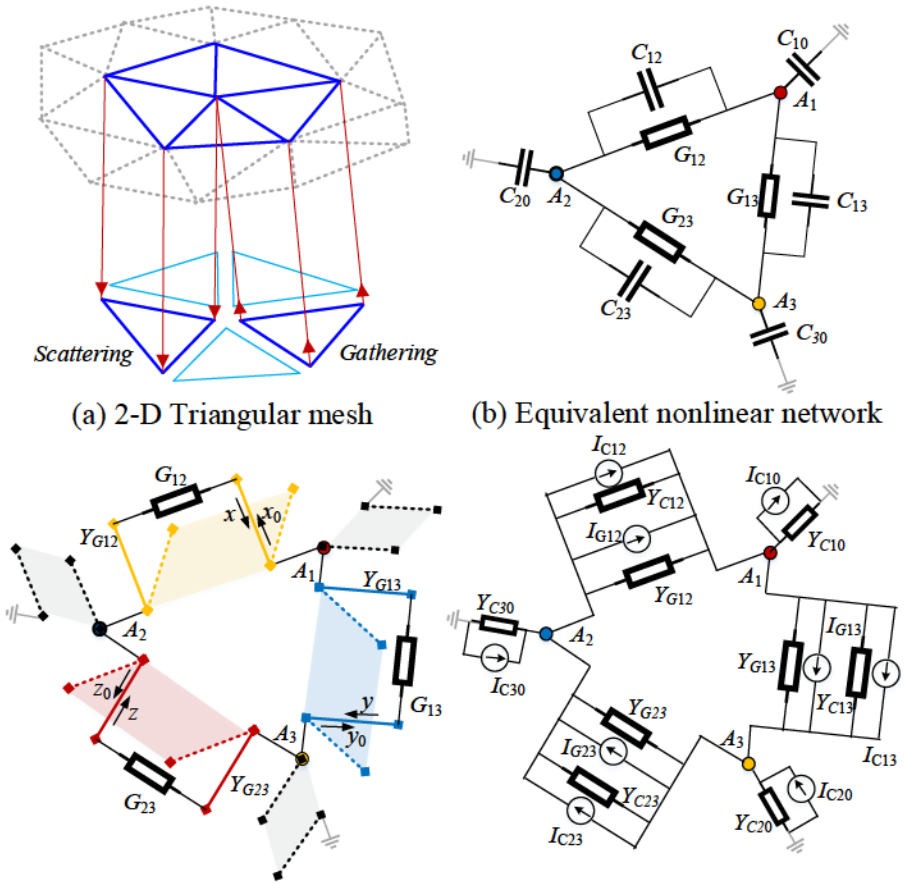
The nonlinear B-H curve of the transformer core can be found in [50], and the magnetic vector potentials on the artificial boundaries are assumed 0.

The elemental equations can be assembled to a global nonlinear system to solve, or alternatively, the equation is equivalent to the nonlinear electrical network in Fig. 6.2 (b) and can be solved using the TLM scheme wherein the component values are given as follows:

$$\begin{aligned}
&G_{12} = -\frac{\upsilon^e}{4\Delta^e}(b_1b_2 + c_1c_2), \;\; Y_{G12} = -\frac{\upsilon_g^e}{4\Delta^e}(b_1b_2 + c_1c_2), \\
&G_{13} = -\frac{\upsilon^e}{4\Delta^e}(b_1b_3 + c_1c_3), \;\; Y_{G13} = -\frac{\upsilon_g^e}{4\Delta^e}(b_1b_3 + c_1c_3), \\
&G_{23} = -\frac{\upsilon^e}{4\Delta^e}(b_2b_3 + c_2c_3), \;\; Y_{G23} = -\frac{\upsilon_g^e}{4\Delta^e}(b_2b_3 + c_2c_3), \\
&C_{12} = C_{13} = C_{23} = -\frac{\sigma^e\Delta^e}{12}, \;\; Y_{C12} = Y_{C13} = Y_{C23} = -\frac{\sigma^e\Delta^e}{6\Delta t}, \\
&C_{10} = C_{20} = C_{30} = \frac{4\sigma^e\Delta^e}{12}, \;\; Y_{C10} = Y_{C20} = Y_{C30} = \frac{4\sigma^e\Delta^e}{6\Delta t}.
\end{aligned} \tag{6.3}$$

The TLM scheme has perfect parallelism in the scattering phase and constant admittance matrix in the gathering phase. In the scattering phase (Fig. 6.2 (c)), the incident pulses based on the nodal solution are injected into each triangular element and the reflected pulses can be calculated individually within each element, thus the nonlinearity is treated in a massively parallel manner. In the gathering phase (Fig. 6.2 (d)), the reflected pulses enter the linear network as new incident pulses, and to obtain the new reflected pulses, a linear network is required to be solved, of which the admittance matrix is featured by the characteristic impedance of the imagined transmission-lines and generally remains unchanged. Thus, repeatedly updating and factorizing the Jacobian matrix in traditional nonlinear FE solver is circumvented, and the TLM technique is felicitous for parallel computing architectures.

Note that the source term in (6.1) is the impressed current density determined by the winding currents. If the electrical networks are connected to the primary or secondary windings (Fig. 6.1), the winding currents are not known, thus a field-circuit coupling



(a) 2-D Triangular mesh (b) Equivalent nonlinear network

(c) TLM model: scattering (d) TLM model: gathering

Figure 6.2: TLM solution for the nonlinear 2-D finite element problem.

scheme is required. Besides, the electrical conductivities of both the winding and transformer core are usually altered by the Joule effects of the winding loss and the eddy current loss, thus a thermal model is required.

6.2.2 Finite Element Model for Heat Conduction

The Fourier's law governing the heat transfer can be expressed by the following partial differential equation [119]:

$$\nabla \cdot (\lambda \nabla T) = \rho C \frac{\partial T}{\partial t} - q, \tag{6.4}$$

where λ is the thermal conductivity; ρ is the volumetric mass; C is the heat capacity; q is the heat source.

The elemental equation has similar form to (6.2), and the natural convective boundary conditions are employed:

$$\lambda \frac{\partial T}{\partial n} = h(T - T_0), \tag{6.5}$$

where n denotes the outward normal direction on the boundary, h is the convection coefficient, and T_0 is the external environmental temperature.

Although the thermal conductivity λ depends on the temperature, implying that (6.4) is also nonlinear. Since the temperature field changes much slower than the electromagnetic transients, thus for simplification, the historical value of λ from the previous time-step is utilized so that the thermal problem becomes linear.

The heat sources originate from the Joule loss in the winding and eddy current loss in the transformer core, and the changing temperature also alters the electrical conductivities of the transformer. In this work, the winding material is copper and the transformer core is made of electrical steel, and their conductivities $\sigma(T)$ altered by the temperature can be represented [7] with

$$\frac{1}{\sigma(T)} = \frac{1}{\sigma_0} + \alpha(T - T_0). \tag{6.6}$$

where T_0 is the ambient temperature and the parameters are provided in the Appendix C.

6.2.3 Multi-Domain Interfacing

Since the integrated model consists of three domains: external electrical network, the magnetic field, and the thermal field, there exist three kinds of interfaces between them, which are illustrated in Fig. 6.3(a). The interfaces of the thermal field with the other two domains are quite direct: the external electrical network can provide the time-varying winding currents, thus the Joule-type loss in the copper windings is available; the time-varying magnetic field will induce electrical field in the steel transformer core, and the eddy current losses can also be obtained after post-processing. Naturally, both the winding loss and eddy current loss are fed to the thermal field as heat sources. In turn, the thermal field updates the temperature distribution at each time-step, and the conductivities of the copper winding and steel transformer core are altered due to change of temperature.

For the external network and magnetic field interface, the coupling coefficients are the self and mutual inductances. According to the Faraday's law, the induced winding voltage can be calculated by:

$$U = rI + \frac{Nl}{\Delta_S} \int_S \frac{\partial A}{\partial t} dS, \tag{6.7}$$

where I is the winding current, r the winding resistance, N the number of turns, l the axial length of each filament, S the winding zone, and Δ_S the area of the winding zone.

Equation (6.7) can be rewritten as the following based on the partial differential chain rule:

$$U = rI + \frac{Nl}{\Delta_S} \int_S \frac{\partial A}{\partial i_p} dS \frac{\partial i_p}{\partial t} + \frac{Nl}{\Delta_S} \int_S \frac{\partial A}{\partial i_s} dS \frac{\partial i_s}{\partial t}. \tag{6.8}$$

Applying (6.8) to the primary and secondary windings respectively, the self and mutual inductances of the transformer can be extracted as:

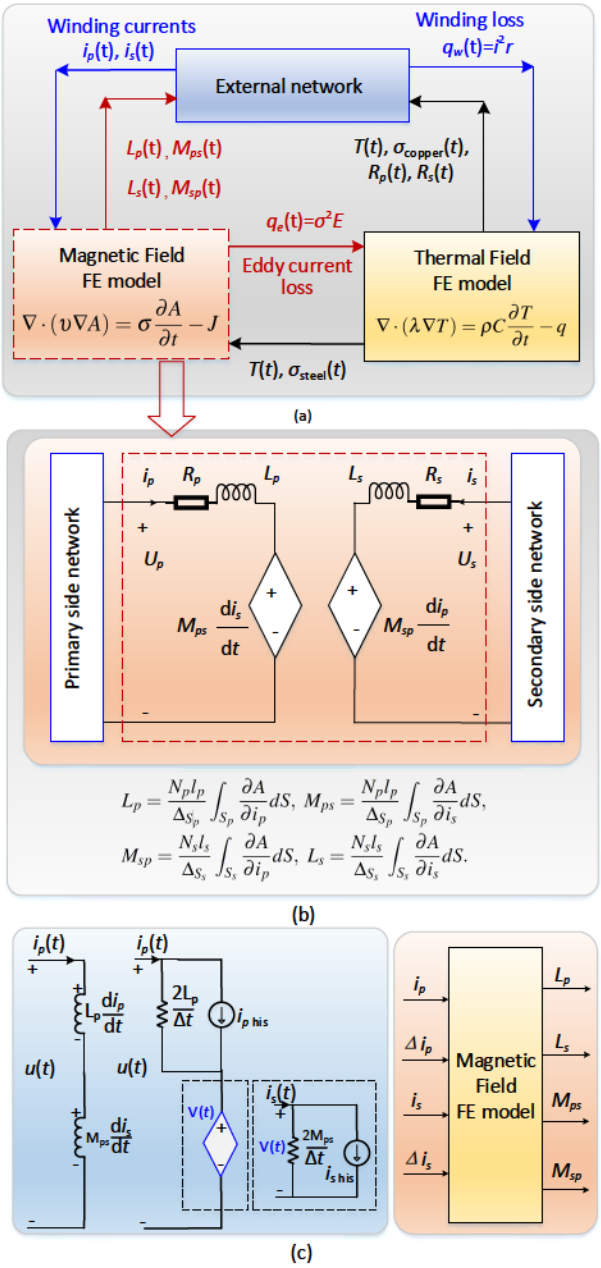


Figure 6.3: (a) Interface of the external electrical network, magnetic field, and thermal field; (b) finite-element transformer represented by self and mutual inductances in electrical networks; (c) Trapezoidal rule applied for discretization of self and mutual inductances.

$$
\begin{aligned}
L_p &= \frac{N_p l_p}{\Delta_{S_p}} \int_{S_p} \frac{\partial A}{\partial i_p} dS,\ M_{ps} = \frac{N_p l_p}{\Delta_{S_p}} \int_{S_p} \frac{\partial A}{\partial i_s} dS, \\
M_{sp} &= \frac{N_s l_s}{\Delta_{S_s}} \int_{S_s} \frac{\partial A}{\partial i_p} dS,\ L_s = \frac{N_s l_s}{\Delta_{S_s}} \int_{S_s} \frac{\partial A}{\partial i_s} dS.
\end{aligned}
\tag{6.9}
$$

Fig. 6.3(b) shows how the finite element transformer model is represented by an equiv-

alent network composed of winding resistance, self, and mutual inductances, to be coupled with the external network, and these nonlinear inductance values are updated by the finite element computation for different winding currents at each time-step.

In EMT simulation, an inductance L can be represented by an impedance of $2L/\Delta t$ in parallel with a historical current term using the Trapezoidal rule. Similarly, as shown in Fig. 6.3(c), the mutual inductance is equivalent to a current-differential controlled voltage source whose value is determined by the mutual inductance and the corresponding branch current after applying the Trapezoidal rule. For example, the KVL equation for the branch in Fig. 6.3(c) can be written as:

$$u(t) = \frac{2L_p}{\Delta t}(i_p(t) - i_{p\ his}) + \frac{2M_{ps}}{\Delta t}(i_s(t) - i_{s\ his}). \tag{6.10}$$

Thus, the mutual inductances serve as active sources and the magnetic field and electrical network are fully coupled. Also, (6.10) implies that the primary side network and the secondary side network are fully coupled by the mutual inductances, and therefore should be solved simultaneously as one system.

6.3 Electro-Thermal Modeling of MMC

In addition to the thermo-electromagnetic FE transformer model described above, the main contributor of nonlinearity to the MMC shown in Fig. 6.4(a) is the power semiconductor switch IGBT and its anti-parallel diode. As the layout corresponds to a large admittance matrix due to the cascaded submodules, they are separated by circuit partitioning using a pair of coupled voltage and current sources [120], and the reconfigured MMC EMT arm model is given in Fig. 6.4(b). The arm current I_{arm} is sent to each MMC submodule, and in return, the SM terminal voltage V_p is fed back to the arm. It can also be seen that after splitting each submodule, the remaining circuit of the MMC is linear.

The prevalent ideal switch model [122] taken as a two-state resistor falls short of revealing a higher-than-normal current stress and the power dissipation during the IGBT switching period. Therefore, the dynamic curve-fitting model which reproduces the shapes of transient waveforms is selected for the comprehensive electro-thermal simulation. The rise and fall times denoted by t_r and t_f respectively are two key parameters reflecting the switching period. Their sensitivity to operation conditions such as collector current, gate resistance, and the junction temperature is tested and provided in the datasheet by the manufacturer. Therefore, they can be uniformly expressed by the following polynomial function

$$t_{r,f}(x_1, x_2, x_3) = A_0 \cdot \prod_{i=1}^{3} x_i + \sum_{k,j=1,2,3}^{k\neq j} A_k x_k x_j + \sum_{m=1}^{3} C_m x_m + C_0, \tag{6.11}$$

where A and C are coefficients, and x represents factors affecting the switching time of the IGBT.

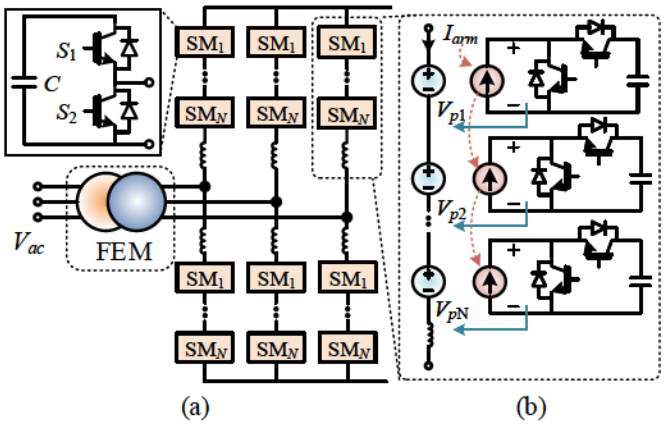


Figure 6.4: Schematic of three-phase MMC-based HVDC converter station: (a) Half-bridge MMC in connection with FEM-based converter transformer, (b) MMC submodule partitioning.

The actual switching waveforms can be divided into multiple sections depending on the curvatures, and then approximated by first-order circuits. For example, the experimental current waveform of the selected IGBT module 5SNA 2000K450300 [123] in Fig. 6.5(a) shows that the initial part of the turn-on process can be taken as a straight line. Afterward, the rising rate decreases and eventually the current falls till entering the steady state. Therefore, the turn-on current can be simulated by an R-L circuit excited by a per unit voltage source V_e, and the inductor current is regulated by changing the resistor, as Fig. 6.5(b) shows. The linear current at stage S_1 is simulated by charging a pure inductor, followed by a varying slope which is approximated by the R-L circuit with an exponential function. The descending waveform starting at t_2 is realized by removing the voltage source V_e so that the inductor discharges exponentially. Therefore, the inductor current in the per unit circuit can be summarized as

$$i_L = \begin{cases} \frac{V_e}{L}(t_1 - 0), & 0 \le t \le t_1 \\ \frac{V_e}{R}(1 - e^{\frac{-t}{\tau}}), & t_1 \le t \le t_2 \\ \frac{I_0}{I_0} + i_L^{max} e^{\frac{-t}{\tau}}, & t_2 \le t \le t_3 \end{cases} \tag{6.12}$$

where I_0 is the steady-state current, and i_L^{max} is the inductor's maximum current in per unit. Then, the inductor current is amplified by K times to simulate the actual device current. Similarly, the device turn-off transients can also be modeled by the first-order circuits.

As a critical part to compose an integral device-level switch model, the transient electro-thermal impedance is based on the following analytical function

$$Z_{th} = \sum_{i=1}^{n} R_{th(i)}(1 - e^{-\frac{t}{\tau_i}}), \tag{6.13}$$

where the impedance $R_{th(i)}$ along with the time constant τ_i can be realized by a paralleled

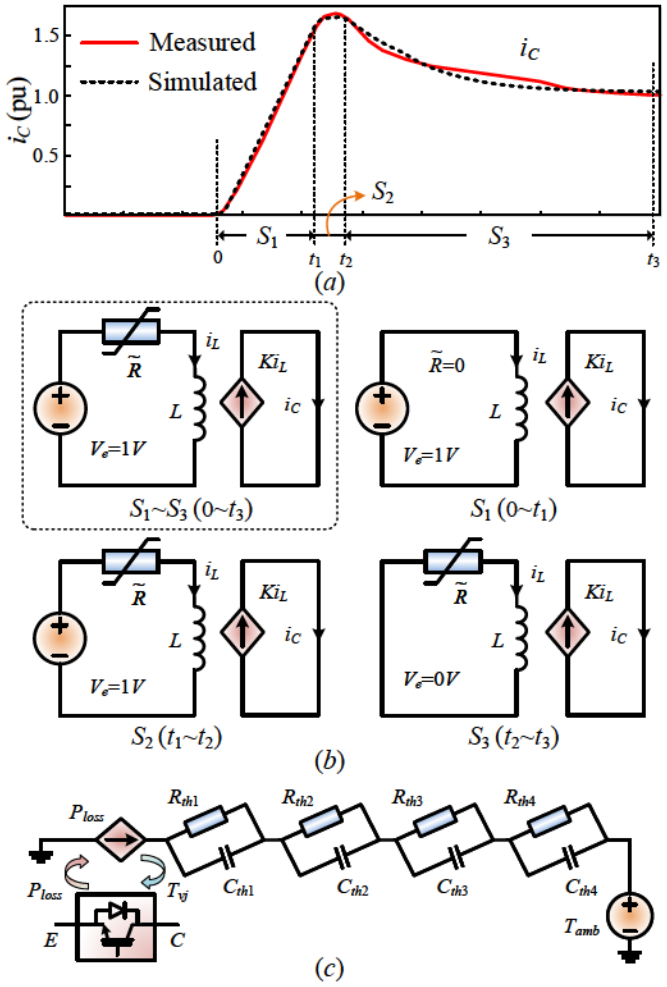


Figure 6.5: IGBT electro-thermal curve-fitting model: (a) Turn-on current [121], (b) realization by first-order circuit, and (c) EMT model of the transient thermal impedance.

R-C pair for EMT simulation, as given in Fig. 6.5(c) where

$$C_{th(i)} = \frac{\tau_i}{R_{th(i)}}. \tag{6.14}$$

In the equivalent circuit of the transient thermal impedance, the input controlled current source is numerically equal to the power loss, and its terminal voltage is deemed as the semiconductor's junction temperature T_{vj}. With the inherent cooling mechanism, the IGBT is exposed to the environment and therefore, the other terminal of the *R-C* pairs is connected to a constant voltage source denoting the ambient temperature T_{amb} which is 25°C.

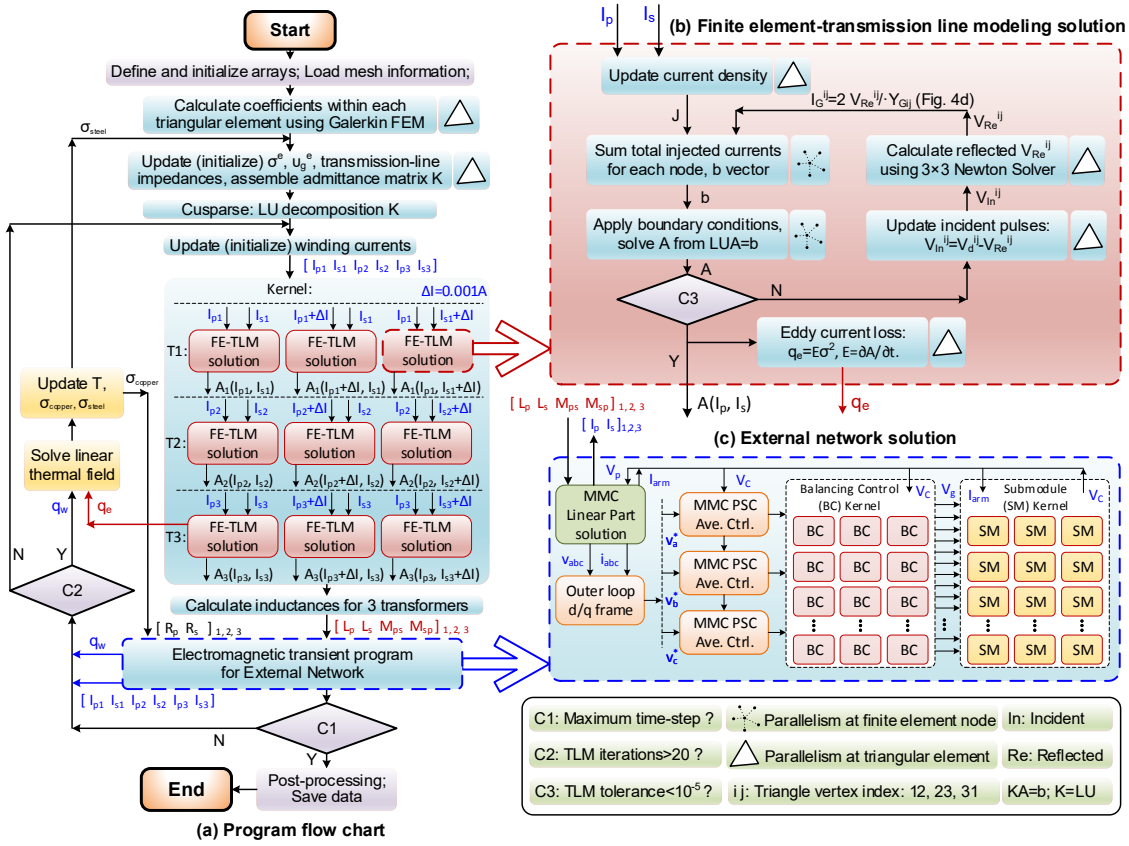


Figure 6.6: Detailed massively parallel implementation of the integrated thermo-electromagnetic model on GPU.

6.4 Parallel Implementation of Integrated Field-Circuit Model on GPU

The detailed program flow, data path, coupling coefficients, and block connections are illustrated in Fig. 6.6. The finite element transmission-line modeling (FE-TLM) solution of the nonlinear magnetic field is provided in details, and the parallelism of each functional block, either at the nodal or the elemental level, is also noted. These blocks are all optimized to fit the Kernel functions in Cuda codes.

The program starts with loading the mesh information such as nodal coordinates and node-element connections, thus the coefficients of Galerkin's FEM in (6.2) and (6.3) can be obtained. According to Fig. 6.2(d) and (6.3), the admittance matrix of the linear network in the gathering phase is determined by two factors: the guessed reluctivities υ_g^e and conductivities σ^e in each triangular element. Due to the thermal effects, the σ^e of transformer core may be altered by the temperature, implying that the admittance matrix should be reassembled and decomposed occasionally. Meanwhile, the adaptive transmission-line impedances, using the solved υ^e of the previous time-step instead of guessed υ_g^e, can be also incorporated into the admittance matrix to efficiently reduce the required number of

TLM iterations. The thermal field has a much larger time constant compared with the electromagnetic field, meaning that the temperature $T(t)$ and $\sigma^e(T)$ change much slower than electromagnetic parameters, so the admittance matrix does not have to be updated at each time-step. To balance the required number of TLM iterations and the computational costs of matrix decomposition, the criteria to update the admittance matrix is determined by whether the required number of TLM iterations is more than 20 or not, which is described as C2 (Condition 2) in Fig. 6.6.

Since the thermal field is assumed a linear FE problem, thus its description is simplified in the flow chart.

The field-circuit coupling scheme described before applies to every single-phase transformer independently, and each consists of three finite element blocks to calculate the partial differentials in (6.9). Figure 6.6 shows three single-phase transformers in an MMC-based HVDC station with nine finite element blocks executed in parallel on the GPU. Once the magnetic vector potentials are calculated, the self and mutual inductances in (6.9) can be obtained for each transformer.

The details of each FE-TLM block are also provided, which includes TLM iterations: the scattering phase with elemental parallelism and gathering phase with nodal parallelism. The summation of total currents into each node can be executed in parallel with appropriate information of node-element connections, and the LU solution process, which essentially is mere backward and forward substitutions, can also be parallelized. The mathematical operations are quite simple to update the next incident pulses while relatively dense for the Newton solver, which requires several (generally 5-15) iterations of updating and solving the 3×3 Jacobian matrix.

The MMC model is then computed following the acquirement of the transformers' inductances. The linear subsystem as part of the field-circuit interface is connected directly to the transformers while other parts, including the controller, are interactive internally. The solution of the interface gives the transformers' primary and secondary currents, which are returned to C2. Inside the MMC, the outer-loop controller regulates either power or DC voltage after transforming the grid voltage and current into the d-q frame. The inner-loop controller adopting phase-shift control (PSC) strategy [124] contains two parts: the averaging control and the balancing control (BC). The former corresponds to the MMC phase, so its kernel has only 3 threads for each HVDC station; while the output of the latter is gate signal V_g that drives a switch in the submodule directly, and therefore, the number of threads that the BC kernel and the SM kernel can invoke is much larger. The only signals that the MMC linear part receives from other MMC blocks are the submodule terminal voltages V_p to complete the computation.

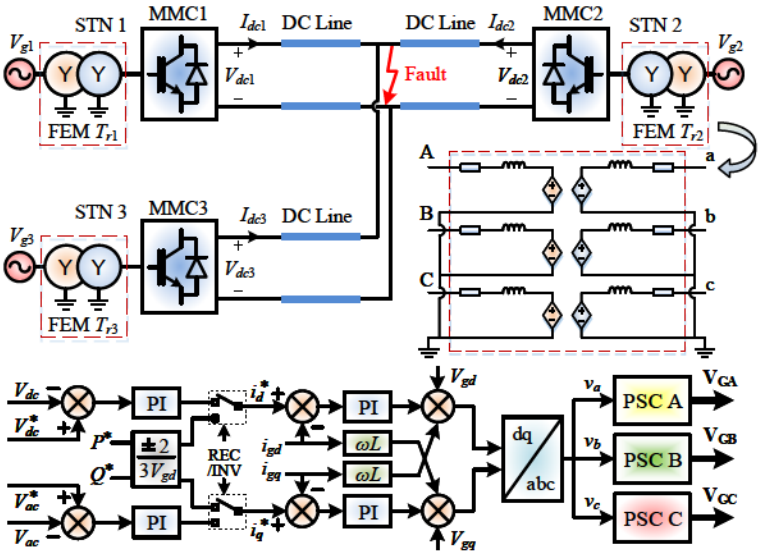


Figure 6.7: Three-terminal HVDC system involving FEM transformer model.

6.5 Case Study and Results

6.5.1 Case Description and Setup

Fig. 6.7 shows the 3-terminal DC system where the MMC-based stations $STN1$ and $STN3$ are rectifier stations, while the inverter station is denoted as $STN2$. All the converter transformers T_{r1-3} adopt the proposed FE model, and the geometric parameters of the FE transformer in Fig. 6.1 are presented in the Appendix C. Each three-phase transformer is built from three single-phase transformers as shown in Fig. 6.6, therefore there are 9 independent single-phase transformers in total, and the inductance calculations can be executed in parallel. The HVDC converter control is conducted in the d-q frame, and depending on the role, the rectifier station regulates the active power in the d-axis, while the inverter station controls the terminal DC voltage V_{dc2}. Since these converter stations are connected by the DC transmission lines, the DC voltages at the two rectifier stations V_{dc1} and V_{dc3} are slightly higher than those of their counterpart. After the inverse Park transformation, the three-phase signals $v_{a,b,c}$ are sent to individual MMC inner-loop controller which adopts phase-shift control strategy where the gate signals for the IGBTs are eventually generated.

6.5.2 External Network Simulation Results

The device-level performance of the IGBT is given in Fig. 6.8 where SaberRD® simulations results are also provided for validation. Figure 6.8(a) gives the experimental IGBT turn-on and turn-off waveforms and the simulated results of the proposed dynamic curve-fitting method. It shows that the linear and nonlinear curves at various stages can be approximated by the proposed multi-section functions as in (6.12). Figure 6.8(b)-(c) are obtained

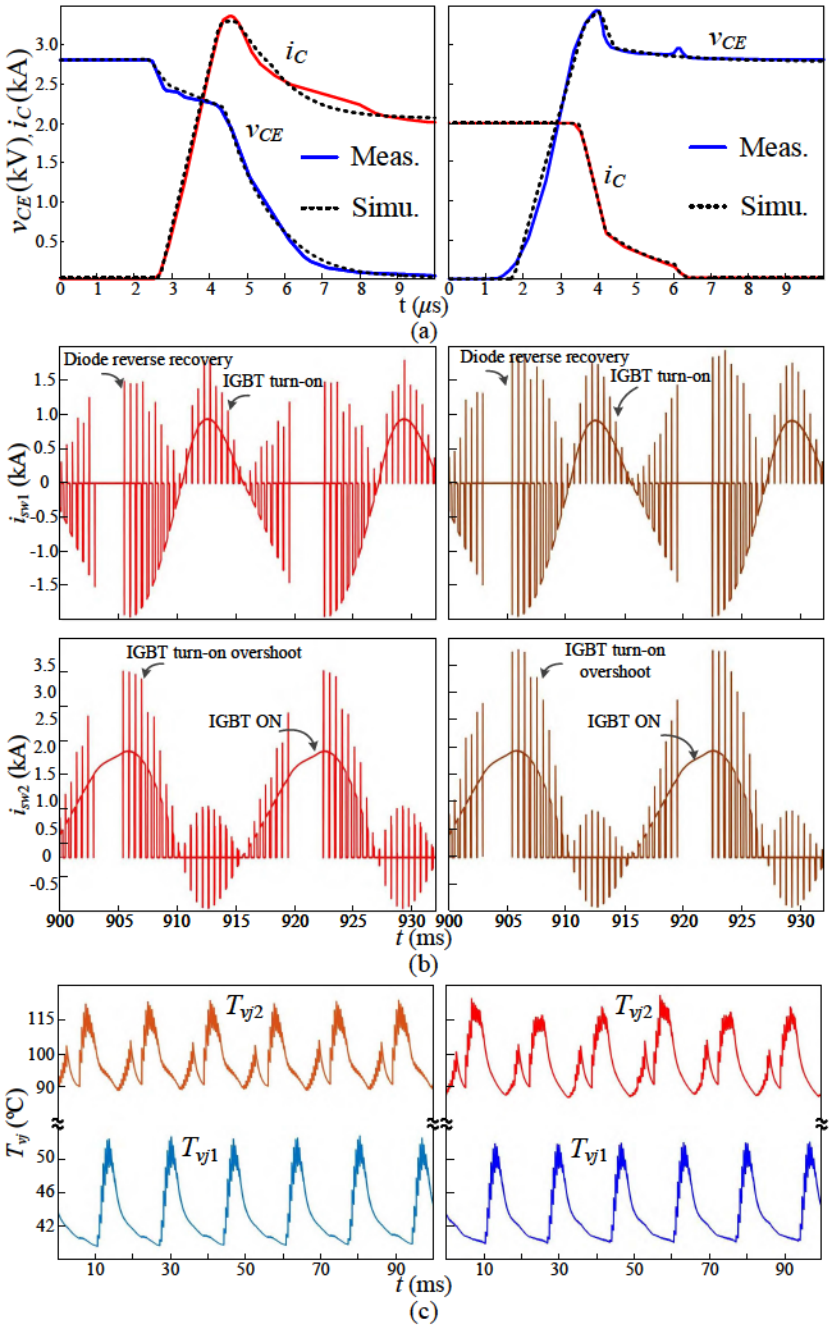


Figure 6.8: IGBT device-level waveforms: (a) Switching transients, (b) upper switch current, (c) lower switch current, and (d) junction temperatures.

from a single-phase 5-level MMC with a DC link voltage of 16kV. The scale of the system is smaller than an HVDC station due to the limited capability of SaberRD® in solving IGBT nonlinearities; nevertheless, it is sufficient for validating the device-level performance of the proposed IGBT model. The turn-on current overshoot and diode reverse recovery current can be observed in the upper and lower switches in an MMC submodule, and the

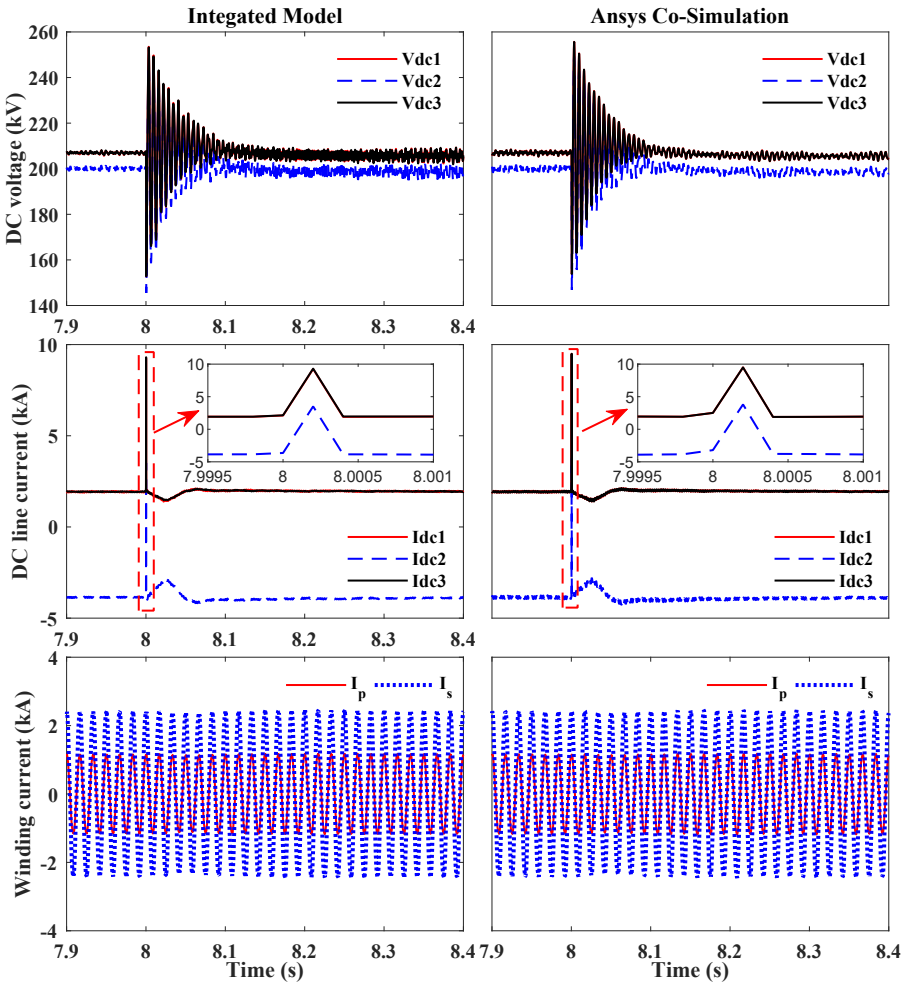


Figure 6.9: MTDC system performance under short-term DC line-line fault.

results from proposed model demonstrate a great similarity to those of the off-line simulation tool, including the junction temperatures.

Fig. 6.9 shows the system-level performance of the 3-terminal DC system subjected to short-term DC line-line fault. Initially, the DC voltage at the inverter station is exactly 200kV, and the two rectifier stations have a slight margin as a result of power transfer, as it can be seen that the DC currents are around 2kA at these two stations and 4kA at the inverter station. At t=8s, the fault lasting 200μs occurs in the middle of the line, and consequently, the profound impacts can be observed in the DC network. The DC voltages witness severe oscillations before being restored by the inverter 100ms later; while the currents first see a dramatic rise and then restore after the fault disappears – the inverter current even witnesses a polarity reversal. On the other hand, the impact on the AC side is negligible since the currents on both sides of the transformer maintain throughout the entire process. The above statements are validated by the right column sub-figure where identical waveforms from Ansys® co-simulation are given.

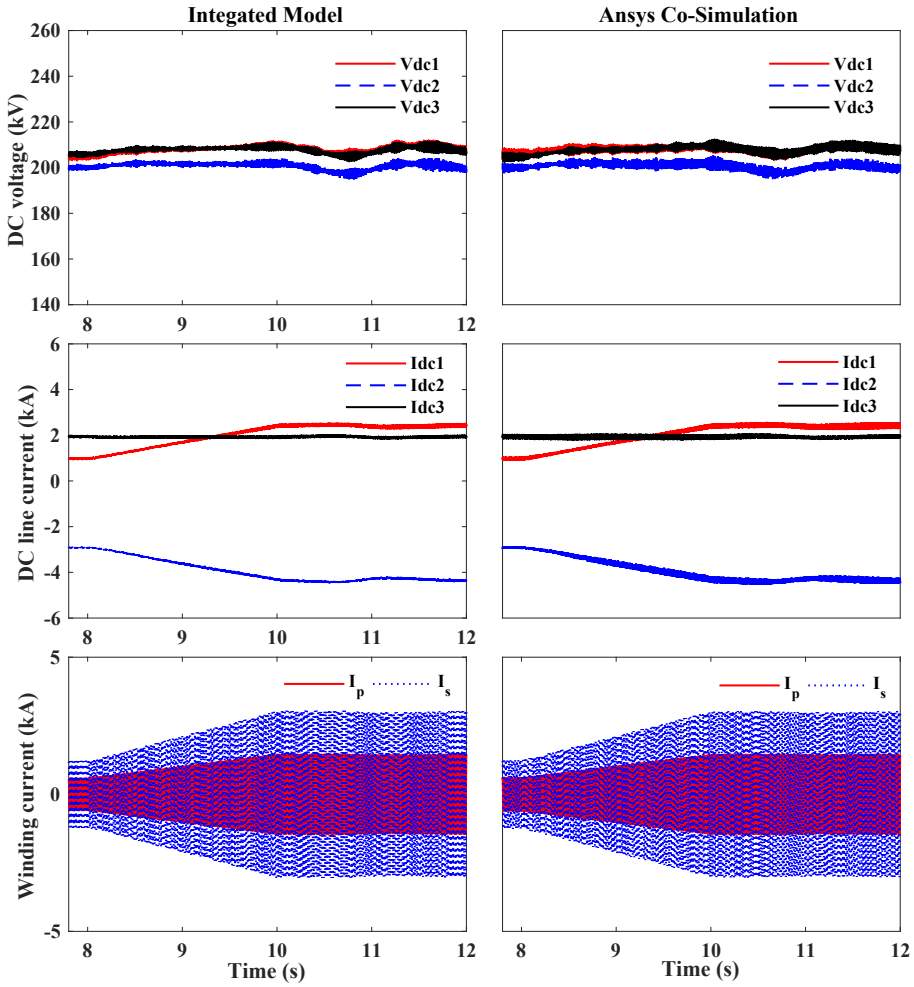


Figure 6.10: MTDC system performance under power reversal.

The power reversal as a typical system-level operation is also conducted in Fig. 6.9. Starting at t=8s, the power reference at Station 1 is ordered to rise from 200MW to 500MW in a time gap of 2s, while Station 3 as the other rectifier maintains its power order. It is observed that during the process, the DC voltages are largely stable, with only slight perturbations. As a consequence, the DC current at Station 1 rises from 1kA to around 2.5kA, and the power at the inverter station increases correspondingly as a combination of its counterparts. At the AC side of the rectifier conducting power reversal, the transformer currents on both sides ramp up due to the increase in the power order. Ansys co-simulation which shows the same results on the right column demonstrates the accuracy of the proposed integrated model.

6.5.3 Finite Element Simulation Results

As mentioned before, the dense computation of FE models can contribute to a more comprehensive view of the physical details within the transformer. With the integrated thermo-

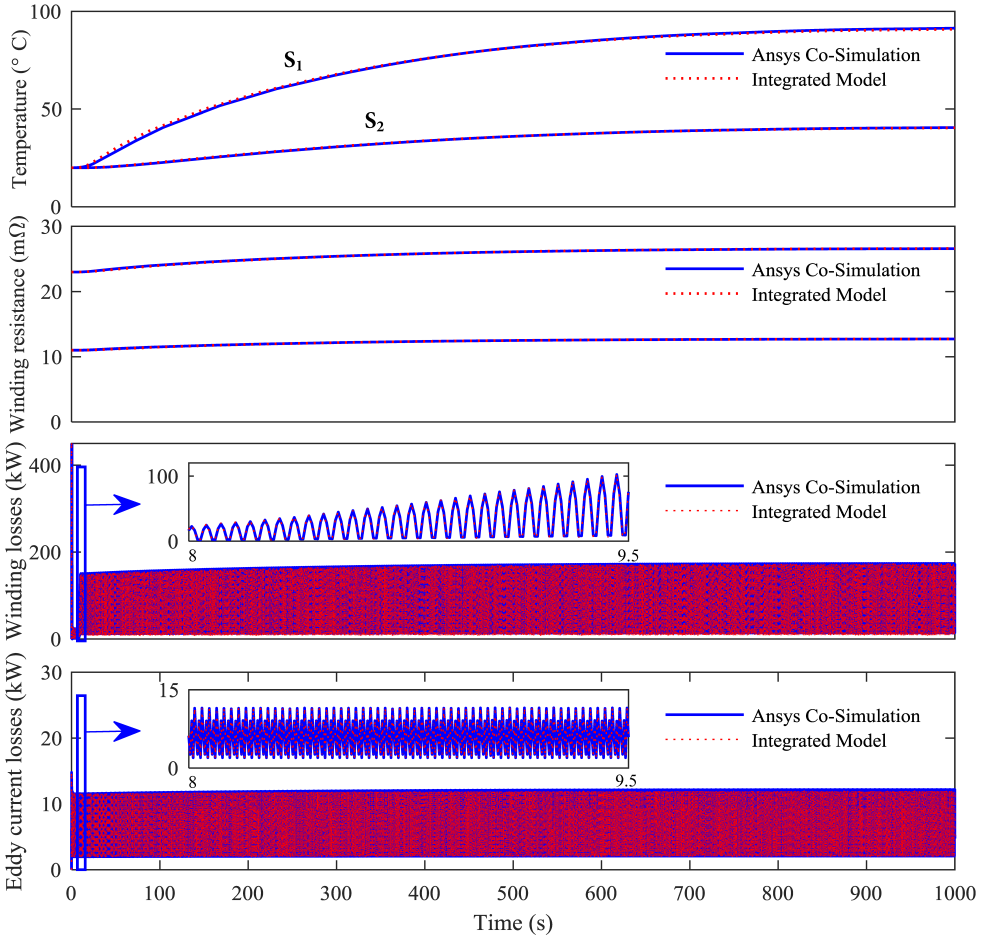


Figure 6.11: Time-varying temperature, winding resistances, winding losses, and eddy current losses of the FE transformer model.

electromagnetic model, all the transient fields can be available at any time point after post-processing. For example, at $t = 1000s$ in the power reversal case study, the field distributions of the magnetic vector potential, magnetic flux density, eddy current density, winding loss and eddy current loss, temperature, and the conductivities altered by temperature, are all plotted in Fig. 6.12.

Undoubtedly, these transient fields in practical working conditions can be very beneficial for designers to make better decisions on transformer problems such as loss reduction, saturation, and over-heating. Besides, the transient information related to such fields can be also monitored. Figure 6.11 shows the time-varying temperature of sample locations S_1 and S_2 noted in the transformer core, how the winding resistance changes due to the thermal effects, the time-varying winding losses, and total eddy current losses. The steady-state thermal field is achieved after 20 minutes with the highest temperature of 95 $^{\circ}C$ in the transformer core, and the total losses also increase until the material conductivities are not altered by temperature anymore.

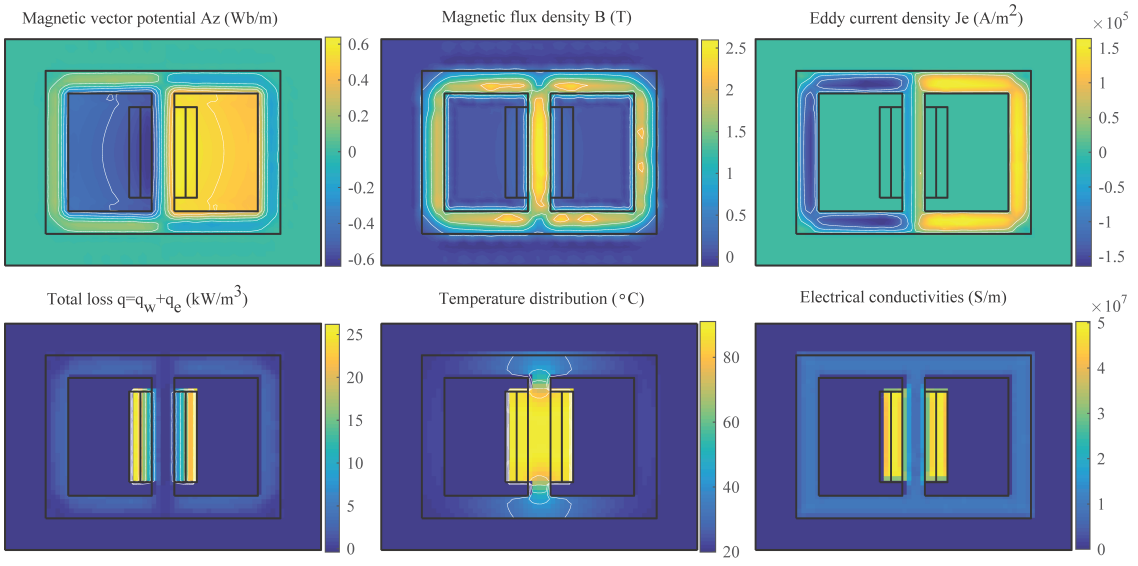


Figure 6.12: Field distributions within the FE transformer (T_{r1} phase A) at $t = 1000s$.

The Ansys® co-simulation results are also plotted to evaluate the accuracy of the integrated model, and it turned out the maximum error is less than 5%. Note that this error is mainly caused by the interfaces and the inherent assumptions between the thermal field, magnetic field, and external networks, because the FE-TLM block itself is quite accurate, and the error is less than 0.1% compared with Ansys Maxwell® when given the same input winding currents.

The massively parallelized codes executed on the NVIDIA Tesla V100 GPU with 5120 Cuda cores have high computational efficiency. The Ansys® co-simulation was carried out on a workstation with dual Intel Xeon E5-2698 v4 CPUs, 20 cores each, 2.2GHz clock frequency, and 128GB RAM. The Ansys HPC (High-Performance Computing) license was utilized and the available number of cores set to 40.

Table 6.1 shows the run-time comparison of Ansys HPC co-simulation and the integrated model on GPU for finite element problems of different sizes, and the total simulation time is 2s with a time-step of 20μs for the system. Although a time-step of 1μs is applied to the MMC submodules, the coupled voltage and current sources exchange information every 20μs. It takes Ansys® co-simulations several days or even weeks to run 10^5 time-steps, while only several hours for the integrated model on GPU, and the speed-up is more than 47 times. On the other hand, the time MMC needs to run a simulation duration of 2s is much shorter than that of the transformer, only 0.11 hour and 0.14 hour when the MMC voltage levels are 5 and 513, respectively. Therefore, the simulation speedup is largely determined by the transformer's FEM model, as the overall speedup is still around 50 in all three cases.

Thus, since the codes are massively parallelized and executed on modern GPU accelerating card, the integrated model can not only provide more comprehensive physical details

Table 6.1: Execution Time and Speedups of Integrated Model Parallelized on GPU (10^5 Time-steps)

Cases	Number of FE Nodes	Ansys Co-Sim Runtime	Integrated Model Runtime			Speedup
			FEM	MMC (5-513L)	Total	
Case 1	505	40.8h	0.72h	0.11-0.14h	0.86h	47.4
Case 2	1973	152.2h	2.53h	0.11-0.14h	2.67h	57.0
Case 3	4923	438.9h	7.94h	0.11-0.14h	8.08h	54.3

within transformers but also substantially decrease the design and test cycle for engineers to run an integrated simulation in AC/DC grids.

6.6 Parasitic Capacitance Extraction

Note that the displacement current term in 2.3 is usually so tiny that can be ignored in low frequency applications. And for medium to high frequency cases, this term can be considered by solving the electric field and magnetic field simultaneously for the full-wave maxwell equation with either the finite-element time-domain (FETD) method [2] or the finite-difference time-domain (FDTD) method [32]. However, considering the computational complexity of solving the coupled Maxwell equations and the field-circuit interface, it seems feasible to solve the electric and magnetic fields separately. Since the electric field does not involve nonlinearity caused by materials while the magnetic field does, a linear electrostatic and a nonlinear magneto-dynamic problem are solved, and the constant parasitic capacitances from the electrostatic problem and the time-varying transformer inductances from the magneto-dynamic problem are extracted to interface the wide-band transformer model with electromagnetic transient studies.

Thus, instead of handling the full-wave (2.3) directly, the following two equations with unknown magnetic potential A and electric potential Φ are to be solved:

$$\nabla \times (\upsilon \nabla \times A) = J - \sigma \frac{\partial A}{\partial t}, \tag{6.15}$$

$$\nabla^2 \phi = -\frac{\rho}{\upsilon_0} \tag{6.16}$$

Fig. 6.13 shows a simplified 2D finite element transformer model rated 220kV/65kV and 100MVA.

According to the electrostatic theory, each conductor is an equipotential object and there exists capacitance between any two conductors with different electric potential. For the 2D transformer in Fig. 6.13, there are parasitic capacitances between four conductors: the primary winding, the secondary winding, the transformer core, and the grounded shielding box, which are indexed as conductor 1, 2, 3, and 0, respectively. In the 2D case, the windings are enclosed by the transformer core, thus no capacitance exists between

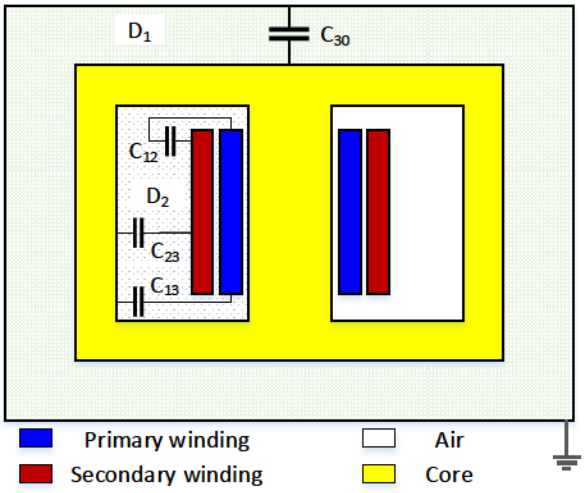


Figure 6.13: Parasitic capacitance existing between conductors.

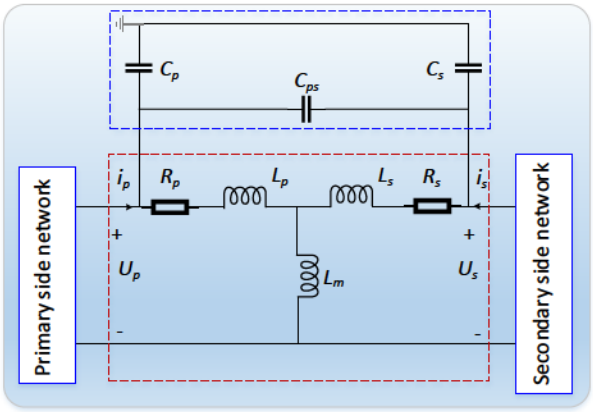


Figure 6.14: Information exchange in a sub-domain for overlapping Schwartz domain decomposition.

the windings and the ground due to the shielding effect. The calculation of the parasitic capacitances shown in the figure is as follows:

To calculate the capacitance C_{30} between the core and ground, one electrostatic case Φ_3 should be solved with such a boundary condition that the electrical potentials on the transformer core are set to 1V. And to calculate the capacitance C_{12}, C_{13}, C_{23} between conductors 1, 2, and 3, two electrostatic cases Φ_1 and Φ_2 are computed with the electrical potentials set to 1V as boundary conditions on the primary and secondary winding, respectively. Note that because of the shielding effect of the transformer core, the calculation of C_{30} is in domain D_1 while the C_{12}, C_{13}, C_{23} in domain D_2.

Since the electric field is represented by $E = -\nabla\Phi$, the capacitances are evaluated by the energy stored in the space between conductors:

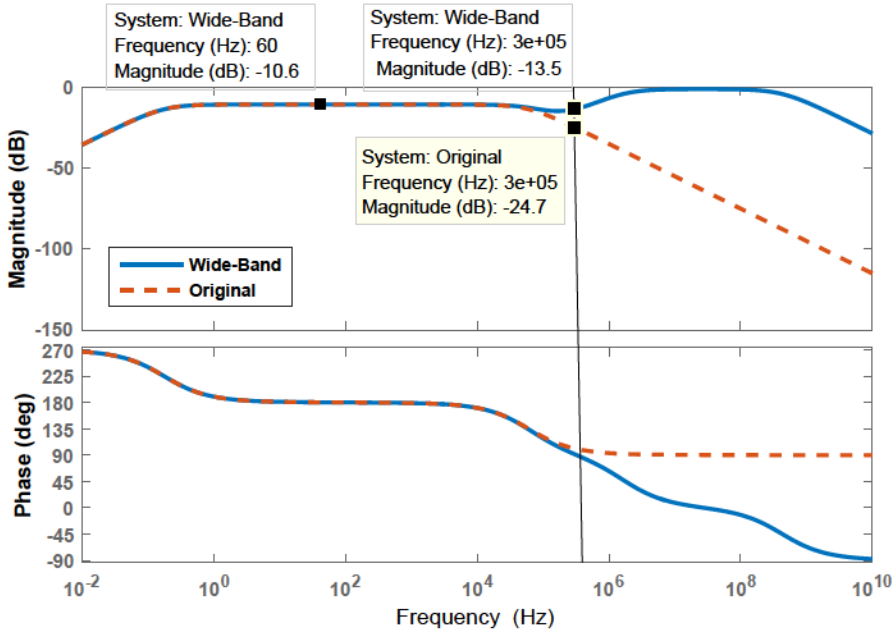


Figure 6.15: Frequency response comparison with and without the parasitic capacitances.

$$
\begin{aligned}
C_{30} &= \int_{D_1} \epsilon_0 E_3 \cdot E_3 \, dS, \; C_{12} = \int_{D_2} \epsilon_0 E_1 \cdot E_2 \, dS, \\
C_{13} &= \int_{D_2} \epsilon_0 E_1 \cdot E_1 \, dS, \; C_{23} = \int_{D_2} \epsilon_0 E_2 \cdot E_2 \, dS.
\end{aligned} \tag{6.17}
$$

The permittivity of air ϵ_0 in (6.17) is unchanged and the electrostatic fields are linear, implying the parasitic capacitances are constant regardless of the winding currents or voltage. Therefore, the electrostatic fields require to be computed once for all. On the contrary, the inductances extracted from the linear magnetic field depend on the external sources and should be updated at each time-step based on the winding currents, which is the main computational burden. The solution to the computational efficiency problem is illustrated in the next section.

As shown in Fig. 6.14, both the magnetic field and the electric field are converted to parameters that can couple with the electrical circuit, and the inductance extraction method described in the previous chapter remains unchanged. After some simplification, the parasitic capacitances are equivalent to the three capacitors: primary-winding to ground, secondary-winding to ground, and primary-winding to secondary-winding.

The frequency responses of the transformer with and without the parasitic capacitances are also different. Figure 6.15 shows the Bode diagram for both cases in one case study, and the detailed parameters are provided in the Appendix. It can be inferred that when the working frequency is under 20 kHz, the transformer can work normally and the magnitude -10.6dB and the voltage ratio match well: 20log(65/220)=-10.59dB. And at this stage, the parasitic capacitance causes little influence and does not seem important. However,

when the working frequency keeps increasing, the transformer ratio starts to fail and the impact of the parasitic capacitances becomes notable.

To further verify the performance of the transformer model with parasitic capacitances, the transient simulation is conducted.

Fig. 6.16 shows the simulation result under a normal working frequency of 60Hz, and the voltage ration matches the rating. The primary winding current I_p, secondary winding current I_s, primary winding voltage U_p, and secondary winding voltage U_s are provided in the figure. The results of the nodal domain decomposition method and Comsol^{TM} perfectly overlap.

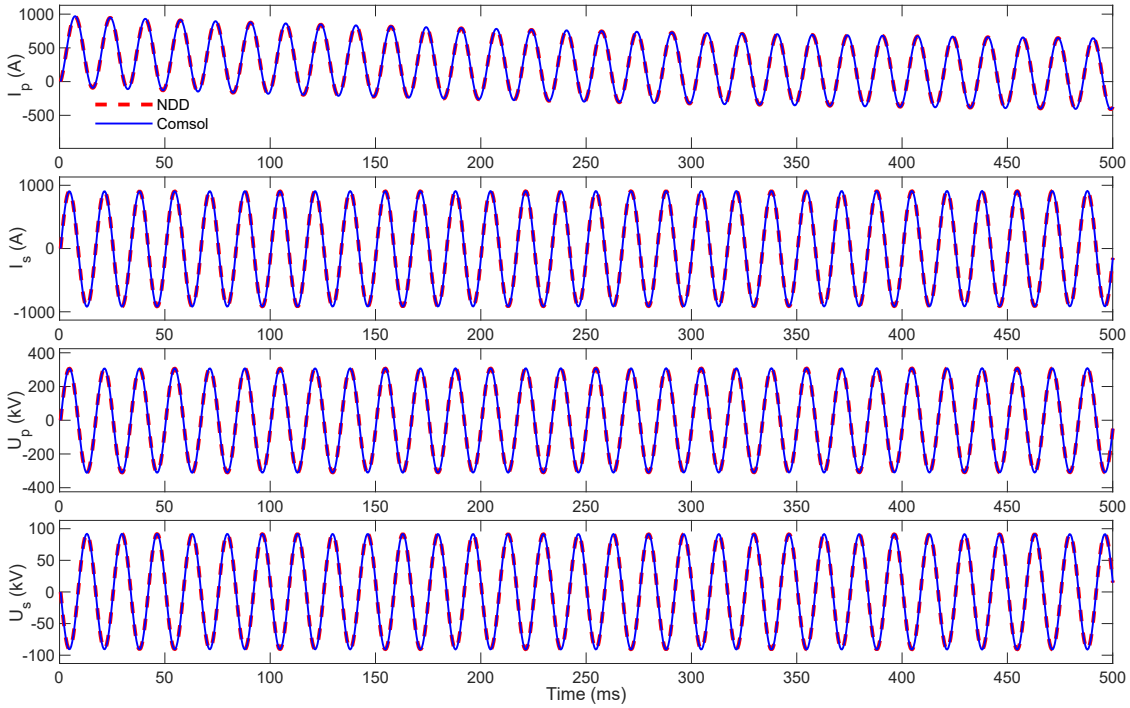


Figure 6.16: Transient simulation result comparison under 60 Hz with a time-step of 0.5ms.

Fig. 6.17 shows the simulation result under a high-frequency condition of 300kHz, and the secondary voltage drops to 12.3kV, and the magnitude ration is 20log(12.5/220)=-24.9dB, which verified the correctness of the Bode diagram.

From the Bode diagram, we can also infer that under 300kHz, the parasitic capacitances are important and will notably influence the performance, both magnitude, and phase, of the transformer. The secondary winding voltage can be predicted as $220*10^{-13.5/20}$=46.5kV. Figure 6.18 shows the simulation result comparison of the transformer with and without parasitic capacitances under 300kHz, and the secondary winding voltage is around 46kV as predicted.

With the extracted wide-band transformer model, the high-frequency conditions can be simulated more accurately. The Bode diagram can very help to analyze the performance

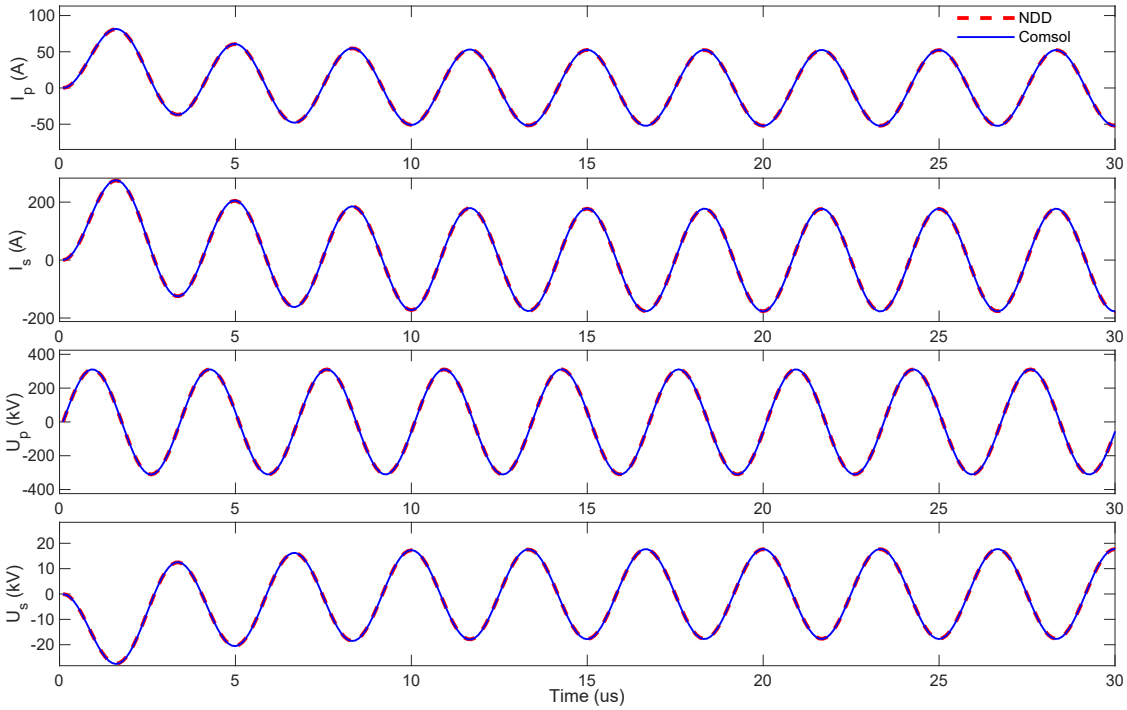


Figure 6.17: Transient simulation result comparison under 300 kHz with a time-step of 0.5ms..

of the transformer in the frequency domain.

6.7 Summary

This chapter proposed an integrated thermo-electromagnetic model to simulate the finite-element transformer transients interacted with MMC in multi-terminal DC grids. The comprehensive physical details enabled by detailed modeling, including the transient field distributions within the transformer and device-level information of the MMC, can aid engineers to make better decisions on practical transformers and MMC problems such as material B-H properties, loss reduction, saturation, and power converter design guide, especially under transient conditions. The thermal field, magnetic field, and the electrical networks are fully coupled, and the information exchanges were implemented by extracting the coupling coefficients. The field-circuit coupling scheme, the finite element transmission-line modeling solution, and the fine-grained MMC model all have perfect parallelism, and the codes were sufficiently parallelized and implemented on Tesla V100 GPU to improve the computational efficiency. Consequently, the execution time of the integrated model is more than 47 times faster than Ansys co-simulation while maintaining good accuracy, and the substantially reduced execution time enables more efficient design and test. It should be mentioned that the results of the proposed parallel method match well with the commercial software regarding the same boundary value problem. To repre-

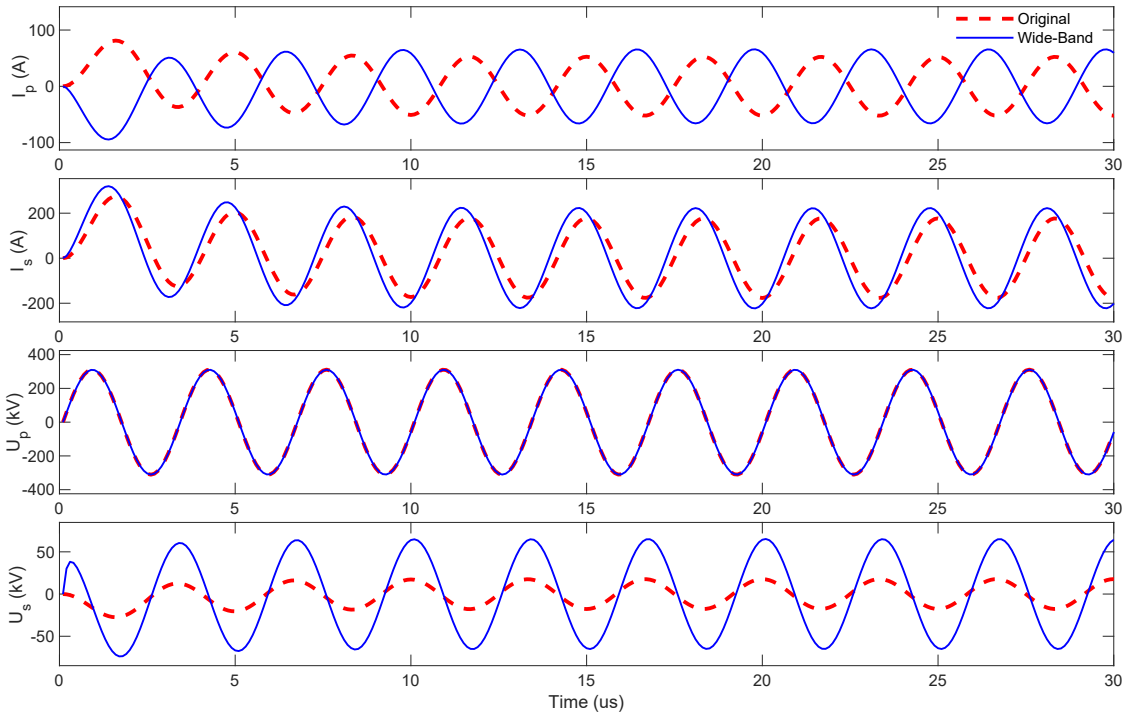


Figure 6.18: Transient simulation result comparison with and without parasitic capacitances under 300 kHz with a time-step of 0.1us.

sent the physical problem more accurately, future work will reduce the assumptions and simplifications that the finite element model utilized.

7 Conclusions and Future Work

This thesis explored the divide-and-conquer strategy for finite element computation at the node-level and element-level in a massively parallel manner and the proposed parallel algorithms can be easily vectorized and accommodated for parallel hardware such as GPU and FPGA. Both the node-level and element-level parallel algorithms can handle the nonlinear problem for time-domain studies. In this way, preferences can be given to the finite element models of transformers and transmission lines in engineering simulation to benefit from the high accuracy and comprehensive field information with substantially improved computational efficiency. The divide-and-conquer strategy was also applied for parallel finite-difference computation for the ionized field around the transmission line. For the application of a fast finite element model, the field-circuit interface was considered to couple the finite element model with the electrical network for electromagnetic transient studies. Other applications of finite element modeling such as the parasitic capacitance extraction from an electrostatic field and multi-physics simulation considering the thermal field were also discussed.

This thesis contributed to the prevalent trend of deploying finite element computation in high-performance computing environments and the trend of highly accurate device-level modeling in an electromagnetic transient simulation.

7.1 Conclusions of Thesis

The conclusions of the thesis are summarized as follows:

- A nodal domain decomposition scheme was proposed to solve the nonlinear finite element problem with node-level parallelism without having to assemble a global matrix. The sub-domain solver showed perfect modularity for single instruction multi-

ple data programming with the specifically defined data structure, and the memory required increased linearly with the problem size. The mixed boundary condition was incorporated to the sub-domain solver to accelerate the convergence. The accuracy and efficiency of the NDD scheme implemented on many-core GPU were discussed for different problem sizes, and comparison with the results from Comsol^{TM} showed a speedup of more than 30 times while maintaining high accuracy (error less than 0.85%). For time-domain finite element computation, the solution of each time-step could served as valuable information to contribute to the convergence of the next time-step.

- A transmission line decoupling technique was explored to solve the nonlinear finite element problem with element-level parallelism. The transmission line decoupling scheme successfully decoupled the nonlinear elements from the linear network so that the computation of the nonlinearities can be massively parallelized. A real-time finite-element transformer model was attempted on FPGA for the first time. The parallelism of the different phases of the proposed scheme was fully explored and implemented on an FPGA board with deep data pipelining. Tests conducted in comparison with a commercial FEM package proved excellent accuracy and computational efficiency of the real-time FEM approach, which provided unprecedented data detail of the simulated transformer.

- The nodal-level parallelism was also extended for the finite difference method used to solve the ionized field around high-voltage direct-current transmission line. The Poisson's equation and current continuity equation were iteratively solved with the parallel finite difference method with differentiated grid sizes on GPU, increasing the computational efficiency of more than 30 times.

- An indirect field-circuit coupling technique was proposed to extract the self and mutual inductances from the winding zones of the finite element transformer at each time-step, and these parameters can be used by the electromagnetic transient studies. The thermal effect of the winding current was also considered in a multi-physics case study and the electrical circuits, magnetic field, and thermal field were solved independently using coupling coefficients.

7.2 Future Research Topics

The proposed future research topics include but are not limited to the following:

- For linear case, the nodal domain decomposition scheme without mixed boundary conditions is equivalent to the Jacobi iterative method and the sub-domain solvers are essentially doing matrix-vector multiplication Ax. Thus the matrix-free scheme

can be also applied in the preconditioned conjugate gradient method where matrix-vector multiplication and inner product of vectors are the main operations.

- The nodal domain decomposition method can perfectly handle nonlinear finite element problems with moving parts like generators since the computation is decentralized, and the changing geometry only affects the coefficients of the nodes involved with no matrix being altered.
- The parallel finite element methods proposed will be also extended to the 3D problem and tested on a workstation with multiple GPUs connected with NVlink for larger problem size. Then it can be deployed on clusters of the workstation. Similarly, the algorithms can also be implemented on multiple connected FPGA boards to enable large problem sizes.
- Explore a parallel algorithm for dynamic mesh generation on GPU to solve finite element problems with moving parts and make the procedures of the finite element computation more complete.
- The parasitic capacitances of the transformer can be extracted from an electrostatic field for electromagnetic transient studies under some high-frequency conditions.

Bibliography

[1] H. W. Dommel, "Digital computer solution of electromagnetic transients in single- and multiphase networks," *IEEE Trans. Power App. Syst.*, vol. PAS-88, no. 4, pp. 388–399, 1969, ISSN: 0018-9510. DOI: 10.1109/TPAS.1969.292459.

[2] J.-M. Jin, *The finite element method in electromagnetics*. John Wiley & Sons, 2015.

[3] J. A. Martinez and B. A. Mork, "Transformer modeling for low- and mid-frequency transients - a review," *IEEE Trans. Power Del.*, vol. 20, no. 2, pp. 1625–1632, 2005, ISSN: 1937-4208. DOI: 10.1109/TPWRD.2004.833884.

[4] J. Liu and V. Dinavahi, "A real-time nonlinear hysteretic power transformer transient model on fpga," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 7, pp. 3587–3597, 2014, ISSN: 0278-0046. DOI: 10.1109/TIE.2013.2279377.

[5] W. Janischewskyj and G. Gela, "Finite element solution for electric fields of coronating dc transmission lines," *IEEE Trans. Power App. Sys.*, vol. PAS-98, no. 3, pp. 1000–1012, 1979, ISSN: 0018-9510. DOI: 10.1109/TPAS.1979.319258.

[6] C. Fu, "Parallel computing for finite element structural analysis on workstation cluster," in *2008 International Conference on Computer Science and Software Engineering*, vol. 3, 2008, pp. 291–294.

[7] K. Preis, O. Biro, G. Buchgraber, and I. Ticar, "Thermal-electromagnetic coupling in the finite-element simulation of power transformers," *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 999–1002, 2006, ISSN: 0018-9464. DOI: 10.1109/TMAG.2006.871439.

[8] W. Wang, O. Kolditz, and T. Nagel, "Parallel finite element modelling of multi-physical processes in thermochemical energy storage devices," *Applied Energy*, vol. 185, pp. 1954–1964, 2017.

[9] S. J. Salon, *Finite element analysis of electrical machines*. Kluwer academic publishers Boston USA, 1995, vol. 101.

[10] Summit. (2019). Supercomputer, [Online]. Available: https://www.olcf.ornl.gov/summit/.

[11] Xilinx. (2019). Vcu118, [Online]. Available: https://www.xilinx.com/support/documentation/boards_and_kits/vcu118/ug1224-vcu118-eval-bd.pdf.

[12] NVidia. (2019). Teslav100, [Online]. Available: https://images.nvidia.com/content/technologies/volta/pdf/tesla-volta-v100-datasheet-letter-fnl-web.pdf.

[13] G. M. Amdahl, "Computer architecture and amdahl's law," *Computer*, vol. 46, no. 12, pp. 38–46, 2013, ISSN: 0018-9162. DOI: 10.1109/MC.2013.418.

[14] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct methods for sparse matrices*. Oxford University Press, 2017.

[15] Y. Saad, *Iterative methods for sparse linear systems*. siam, 2003, vol. 82.

[16] X. S. Li, "An overview of superlu: Algorithms, implementation, and user interface," *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 3, pp. 302–325, 2005.

[17] O. Schenk, K Gärtner, G Karypis, S Röllin, and M Hagemann, "Pardiso solver project," *URL: http://www. pardiso-project. org*, 2010.

[18] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent, "A fully asynchronous multifrontal solver using distributed dynamic scheduling," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 1, pp. 15–41, 2001.

[19] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet, "Hybrid scheduling for the parallel solution of linear systems," *Parallel Computing*, vol. 32, no. 2, pp. 136–156, 2006.

[20] M. M. Dehnavi, D. M. Fernandez, and D. Giannacopoulos, "Enhancing the performance of conjugate gradient solvers on graphic processing units," *IEEE Trans. Magn.*, vol. 47, no. 5, pp. 1162–1165, 2011, ISSN: 0018-9464. DOI: 10.1109/TMAG.2010.2081662.

[21] ——, "Enhancing the performance of conjugate gradient solvers on graphic processing units," *IEEE Trans. Magn.*, vol. 47, no. 5, pp. 1162–1165, 2011, ISSN: 0018-9464. DOI: 10.1109/TMAG.2010.2081662.

[22] T. Okimura, T. Sasayama, N. Takahashi, and S. Ikuno, "Parallelization of finite element analysis of nonlinear magnetic fields using gpu," *IEEE Trans. Magn.*, vol. 49, no. 5, pp. 1557–1560, 2013, ISSN: 0018-9464. DOI: 10.1109/TMAG.2013.2244062.

[23] D. M. Fernandez, M. M. Dehnavi, W. J. Gross, and D. Giannacopoulos, "Alternate parallel processing approach for fem," *IEEE Trans. Magn.*, vol. 48, no. 2, pp. 399–402, 2012, ISSN: 0018-9464. DOI: 10.1109/TMAG.2011.2173304.

[24] J. P. A. Bastos and N. Sadowski, "A new method to solve 3-d magnetodynamic problems without assembling an $ax = b$ system," *IEEE Trans. Magn.*, vol. 46, no. 8, pp. 3365–3368, 2010, ISSN: 0018-9464. DOI: 10.1109/TMAG.2010.2044873.

[25] J. P. A. Bastos and N. Sadowski, *Magnetic materials and 3D finite element modeling*. CRC press, 2013.

[26] I. Kiss, S. Gyimothy, Z. Badics, and J. Pavo, "Parallel realization of the element-by-element fem technique by cuda," *IEEE Trans. Magn.*, vol. 48, no. 2, pp. 507–510, 2012, ISSN: 0018-9464. DOI: 10.1109/TMAG.2011.2175905.

[27] G. F. Carey, E. Barragy, R. McLay, and M. Sharma, "Element-by-element vector and parallel computations," *Commun. Appl. Numer. Methods*, vol. 4, no. 3, pp. 299–307, 1988.

[28] B. Smith, P. Bjorstad, and W. Gropp, *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge university press, 2004.

[29] Comsol. (2019). Acdc, [Online]. Available: https://www.comsol.com/acdc-module.

[30] Ansys. (2019). Maxwell, [Online]. Available: https://www.ansys.com/products/electronics/ansys-maxwell.

[31] W. F. Ames, *Numerical methods for partial differential equations*. Academic press, 2014.

[32] A. Z. Elsherbeni and V. Demir, *The finite-difference time-domain method for electromagnetics with MATLAB simulations*. The Institution of Engineering and Technology, 2016.

[33] A. C. Polycarpou, "Introduction to the finite element method in electromagnetics," *Synthesis Lectures on Computational Electromagnetics*, vol. 1, no. 1, pp. 1–126, 2005.

[34] M. J. Flynn, "Some computer organizations and their effectiveness," *IEEE Trans. on Computers*, vol. 100, no. 9, pp. 948–960, 1972.

[35] P. Pacheco, *An introduction to parallel programming*. Elsevier, 2011.

[36] A. López-Ortiz, A. Salinger, and R. Suderman, "Toward a generic hybrid cpu-gpu parallelization of divide-and-conquer algorithms," in *2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, 2013, pp. 601–610.

[37] X. Liu, H. A. Ounifi, A. Gherbi, Y. Lemieux, and W. Li, "A hybrid gpu-fpga-based computing platform for machine learning," *Procedia Computer Science*, vol. 141, pp. 104–111, 2018.

[38] T. Okimura, T. Sasayama, N. Takahashi, and S. Ikuno, "Parallelization of finite element analysis of nonlinear magnetic fields using gpu," *IEEE Trans. Magn.*, vol. 49, no. 5, pp. 1557–1560, 2013, ISSN: 0018-9464. DOI: 10.1109/TMAG.2013.2244062.

[39] A. F. P. de Camargos, V. C. Silva, J. M. Guichon, and G. Munier, "Efficient parallel preconditioned conjugate gradient solver on gpu for fe modeling of electromagnetic fields in highly dissipative media," *IEEE Trans. Magn.*, vol. 50, no. 2, pp. 569–572, 2014, ISSN: 0018-9464. DOI: 10.1109/TMAG.2013.2285091.

[40] Q. Dinh and Y. Marechal, "Toward real-time finite-element simulation on gpu," *IEEE Trans. Magn.*, vol. 52, no. 3, pp. 1–4, 2016, ISSN: 0018-9464. DOI: 10.1109/TMAG.2015.2477602.

[41] W. J. Wang, R. Xu, H. Y. Li, Y. Liu, X. Y. Guo, Y. Xu, H. L. Li, H. J. Zhou, and W. Y. Yin, "Massively parallel simulation of large-scale electromagnetic problems using one high-performance computing scheme and domain decomposition method," *IEEE Trans. Electromagn. Compat.*, vol. 59, no. 5, pp. 1523–1531, 2017, ISSN: 0018-9375. DOI: 10.1109/TEMC.2017.2656891.

[42] Y. Takahashi, K. Fujiwara, T. Iwashita, and H. Nakashima, "Parallel finite-element analysis of rotating machines based on domain decomposition considering nonconforming mesh connection," *IEEE Trans. Magn.*, vol. 52, no. 3, pp. 1–4, 2016, ISSN: 0018-9464. DOI: 10.1109/TMAG.2015.2477308.

[43] I. Kiss, S. Gyimothy, Z. Badics, and J. Pavo, "Parallel realization of the element-by-element fem technique by cuda," *IEEE Trans. Magn.*, vol. 48, no. 2, pp. 507–510, 2012, ISSN: 0018-9464. DOI: 10.1109/TMAG.2011.2175905.

[44] P. Liu and V. Dinavahi, "Finite-difference relaxation for parallel computation of ionized field of hvdc lines," *IEEE Trans. Power Del.*, vol. 33, no. 1, pp. 119–129, 2018, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2017.2661845.

[45] V. Jalili-Marandi, Z. Zhou, and V. Dinavahi, "Large-scale transient stability simulation of electrical power systems on parallel gpus," *IEEE Trans. Parallel Distrib. Syst*, vol. 23, no. 7, pp. 1255–1266, 2012, ISSN: 1045-9219. DOI: 10.1109/TPDS.2011.291.

[46] H. Karimipour and V. Dinavahi, "Extended kalman filter-based parallel dynamic state estimation," *IEEE Trans. Smart Grid*, vol. 6, no. 3, pp. 1539–1549, 2015, ISSN: 1949-3053. DOI: 10.1109/TSG.2014.2387169.

[47] O. Deblecker, J. Lobry, and C. Broche, "Use of transmission-line modelling method in fem for solution of nonlinear eddy-current problems," *IEE Proceedings - Science, Measurement and Technology*, vol. 145, no. 1, pp. 31–38, 1998, ISSN: 1350-2344. DOI: 10.1049/ip-smt:19981639.

[48] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu, *The finite element method*, 6th. Amsterdam; London: Elsevier Butterworth-Heinemann, 2005, p. 733, ISBN: 0750663200.

[49] M. Dryja and O. B. Widlund, *Towards a unified theory of domain decomposition algorithms for elliptic problems*. New York University, Courant Institute of Mathematical Sciences, Division of Computer Science, 1989.

[50] MagWeb. (2019). Bh curves, [Online]. Available: http://magweb.us/free-bh-curves/.

[51] J. A. Martinez and B. A. Mork, "Transformer modeling for low- and mid-frequency transients - a review," *IEEE Trans. Power Del.*, vol. 22, no. 2, pp. 1235–1246, 2007, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2008.924192.

[52] V. Brandwajn, H. W. Dommel, and I. I. Dommel, "Matrix representation of three-phase n-winding transformers for steady-state and transient studies," *IEEE Trans. Power App. Syst.*, vol. PAS-101, no. 6, pp. 1369–1378, 1982, ISSN: 0018-9510. DOI: 10.1109/TPAS.1982.317184.

[53] F. de Leon and A. Semlyen, "Complete transformer model for electromagnetic transients," *IEEE Trans. Power Del.*, vol. 9, no. 1, pp. 231–239, 1994, ISSN: 0885-8977. DOI: 10.1109/61.277694.

[54] N. D. Hatziargyriou, J. M. Prousalidis, and B. C. Papadias, "Generalised transformer model based on the analysis of its magnetic core circuit," *IET Gen., Transm. Distrib*, vol. 140, no. 4, pp. 269–278, 1993, ISSN: 0143-7046. DOI: 10.1049/ip-c.1993.0040.

[55] A. Narang and R. H. Brierley, "Topology based magnetic model for steady-state and transient studies for three-phase core type transformers," *IEEE Trans. on Power Syst.*, vol. 9, no. 3, pp. 1337–1349, 1994, ISSN: 0885-8950. DOI: 10.1109/59.336132.

[56] J. Arrillaga, W. Enright, N. R. Watson, and A. R. Wood, "Improved simulation of hvdc converter transformers in electromagnetic transient programs," *IET Gen., Transm. Distrib.*, vol. 144, no. 2, pp. 100–106, 1997, ISSN: 1350-2360. DOI: 10.1049/ip-gtd:19970849.

[57] J. Liu and V. Dinavahi, "A real-time nonlinear hysteretic power transformer transient model on fpga," *IEEE Trans. Ind. Electron.*, vol. 61, no. 7, pp. 3587–3597, 2014, ISSN: 0278-0046. DOI: 10.1109/TIE.2013.2279377.

[58] ——, "Nonlinear magnetic equivalent circuit-based real-time sen transformer electromagnetic transient model on fpga for hil emulation," *IEEE Trans. Power Del.*, vol. 31, no. 6, pp. 2483–2493, 2016, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2016.2518676.

[59] P. B. Johns and R. L. Beurle, "Numerical solution of 2-dimensional scattering problems using a transmission-line matrix," *Proceedings of the Institution of Electrical Engineers*, vol. 118, no. 9, pp. 1203–1208, 1971, ISSN: 0020-3270. DOI: 10.1049/piee.1971.0217.

[60] P. B. Johns and M. O'Brien, "Use of the transmission-line modelling (t.l.m.) method to solve non-linear lumped networks," *Radio and Electronic Engineer*, vol. 50, no. 1.2, pp. 59–70, 1980, ISSN: 0033-7722. DOI: 10.1049/ree.1980.0006.

[61] W. J. R. Hoefer, "The transmission-line matrix method - theory and applications," *IEEE Trans. Microw. Theory Techn.*, vol. 33, no. 10, pp. 882–893, 1985, ISSN: 0018-9480. DOI: 10.1109/TMTT.1985.1133146.

[62] S. Y. R. Hui and C. Christopoulos, "Physical science, measurement and instrumentation, management and education, iee proceedings a," *IEE Proceedings A - Physical Science, Measurement and Instrumentation, Management and Education*, vol. 137, no. 6, pp. 379–384, 1990, ISSN: 0143-702X.

[63] V. Dinavahi, "Transient analysis of systems with multiple nonlinear elements using tlm," *IEEE Trans. Power Syst.*, vol. 19, no. 4, pp. 2102–2103, 2004, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2004.836165.

[64] B. Asghari and V. Dinavahi, "Real-time nonlinear transient simulation based on optimized transmission line modeling," *IEEE Trans. Power Syst.*, vol. 26, no. 2, pp. 699–709, 2011, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2010.2066992.

[65] J. Lobry, J. Trecat, and C. Broche, "The transmission line modeling (tlm) method as a new iterative technique in nonlinear 2-d magnetostatics," *IEEE Trans. Magn.*, vol. 32, no. 2, pp. 559–566, 1996, ISSN: 0018-9464. DOI: 10.1109/20.486548.

[66] O. Deblecker, J. Lobry, and C. Broche, "Novel algorithm based on transmission-line modeling in the finite-element method for nonlinear quasi-static field analysis," *IEEE Trans. Magn.*, vol. 39, no. 1, pp. 529–538, 2003, ISSN: 0018-9464. DOI: 10.1109/TMAG.2002.806334.

[67] I. D. Mayergoyz, *Mathematical models of hysteresis*. Berlin, Germany: Springer-Verlag, 1991.

[68] J. Fuzi, "Analytical approximation of preisach distribution functions," *IEEE Trans. Magn.*, vol. 39, no. 3, pp. 1357–1360, 2003, ISSN: 0018-9464. DOI: 10.1109/TMAG.2003.810536.

[69] C. Christopoulos, "The transmission-line modeling (tlm) method in electromagnetics," *Synthesis Lectures on Computational Electromagnetics*, vol. 1, no. 1, pp. 1–132, 2005.

[70] B. Asghari, V. Dinavahi, M. Rioual, J. A. Martinez, and R. Iravani, "Interfacing techniques for electromagnetic field and circuit simulation programs ieee task force on interfacing techniques for simulation tools," *IEEE Trans. Power Del.*, vol. 24, no. 2, pp. 939–950, 2009, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2008.2002699.

[71] P. Lombard and G. Meunier, "A general method for electric and magnetic coupled problem in 2d and magnetodynamic domain," *IEEE Trans. Magn.*, vol. 28, no. 2, pp. 1291–1294, 1992, ISSN: 0018-9464. DOI: 10.1109/20.123926.

[72] J. Li, P. Liu, and V. Dinavahi, "Massively parallel computation for 3-d nonlinear finite edge element problem with transmission line decoupling technique," *IEEE Trans. on Magn.*, vol. 55, no. 10, pp. 1–8, 2019.

[73] Y. Zhen, X. Cui, T. Lu, Y. Liu, X. Li, L. Li, and W. Zhang, "Ion flow field analysis considering the finite conductivity of the building near hvdc transmission lines," *IEEE Trans. Magn.*, vol. 51, no. 3, pp. 1–4, 2015, ISSN: 0018-9464. DOI: 10.1109/TMAG.2014.2357834.

[74] P. S. Maruvada and W. Janischewskyj, "Analysis of corona losses on dc transmission lines: I - unipolar lines," *IEEE Trans. Power App. Sys.*, vol. PAS-88, no. 5, pp. 718–731, 1969, ISSN: 0018-9510. DOI: 10.1109/TPAS.1969.292362.

[75] J. S. Townsend, "The potentials required to maintain currents between coaxial cylinders," *Philosophical Magazine Series 6*, vol. 28, no. 163, pp. 83–90, 1914.

[76] W. Deutsch, "Über die dichteverteilung unipolarer ionenströme.," *Annalen der Physik*, vol. 408, no. 5, pp. 588–612, 1933, ISSN: 00033804.

[77] Z. Luo, X. Cui, W. Zhang, and J. Lu, "Calculation of the 3-d ionized field under hvdc transmission lines," *IEEE Trans. Magn.*, vol. 47, no. 5, pp. 1406–1409, 2011, ISSN: 0018-9464. DOI: 10.1109/TMAG.2010.2090514.

[78] P. S. Maruvada, "Electric field and ion current environment of hvdc transmission lines: Comparison of calculations and measurements," *IEEE Trans. Power Del.*, vol. 27, no. 1, pp. 401–410, 2012, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2011.2172003.

[79] J. Qiao, J. Zou, and B. Li, "Calculation of the ionised field and the corona losses of high-voltage direct current transmission lines using a finite-difference-based flux tracing method," *IET Gen., Trans. Distrib.*, vol. 9, no. 4, pp. 348–357, 2015, ISSN: 1751-8687. DOI: 10.1049/iet-gtd.2014.0333.

[80] T. Takuma, T. Ikeda, and T. Kawamoto, "Calculation of ion flow fields of hvdc transmission lines by the finite element method," *IEEE Power Eng. Rev.*, vol. PER-1, no. 12, pp. 28–28, 1981, ISSN: 0272-1724. DOI: 10.1109/MPER.1981.5511970.

[81] T. Takuma and T. Kawamoto, "A very stable calculation method for ion flow field of hvdc transmission line," *IEEE Power Eng. Rev.*, vol. PER-7, no. 1, pp. 51–52, 1987, ISSN: 0272-1724. DOI: 10.1109/MPER.1987.5527318.

[82] B. Qin, J. Sheng, Z. Yan, and G. Gela, "Accurate calculation of ion flow field under hvdc bipolar transmission lines," *IEEE Trans. Power Del.*, vol. 3, no. 1, pp. 368–376, 1988, ISSN: 0885-8977. DOI: 10.1109/61.4266.

[83] M. Yu and E. Kuffel, "A new algorithm for evaluating hvdc ionized fields in the presence of strong wind," in *Electromagnetic Field Computation, 1992. Digest of the Fifth Biennial IEEE Conference on*, 1992, TOD7–TOD7. DOI: 10.1109/CEFC.1992.720703.

[84] Z. Al-Hamouz, "Corona power loss computation in bundled bipolar conductors," in *Industry Applications Conference, 2000. Conference Record of the 2000 IEEE*, vol. 2, 2000, 719–724 vol.2. DOI: 10.1109/IAS.2000.881909.

[85] ——, "Corona power loss, electric field, and current density profiles in bundled horizontal and vertical bipolar conductors," *IEEE Trans. Ind. Appl.*, vol. 38, no. 5, pp. 1182–1189, 2002, ISSN: 0093-9994. DOI: 10.1109/TIA.2002.802931.

[86] T. Lu, H. Feng, X. Cui, Z. Zhao, and L. Li, "Analysis of the ionized field under hvdc transmission lines in the presence of wind based on upstream finite element method," *IEEE Trans. Magn.*, vol. 46, no. 8, pp. 2939–2942, 2010, ISSN: 0018-9464. DOI: 10.1109/TMAG.2010.2044149.

[87] G. Huang, J. Ruan, Z. Du, and C. Zhao, "Highly stable upwind fem for solving ionized field of hvdc transmission line," *IEEE Trans. Magn.*, vol. 48, no. 2, pp. 719–722, 2012, ISSN: 0018-9464. DOI: 10.1109/TMAG.2011.2174203.

[88] J. Liu, J. Zou, J. Tian, and J. Yuan, "Analysis of electric field, ion flow density, and corona loss of same-tower double-circuit hvdc lines using improved fem," *IEEE Trans. Power Del.*, vol. 24, no. 1, pp. 482–483, 2009, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2008.2007009.

[89] Y. Zhen, X. Cui, T. Lu, X. Zhou, and Z. Luo, "High efficiency fem calculation of the ionized field under hvdc transmission lines," *IEEE Trans. Magn.*, vol. 48, no. 2, pp. 743–746, 2012, ISSN: 0018-9464. DOI: 10.1109/TMAG.2011.2172195.

[90] J. Davis and J. Hoburg, "Hvdc transmission line computations using finite element and characteristics method.," *Journal of Electrostatics*, vol. 18, pp. 1 –22, 1986, ISSN: 0304-3886.

[91] X. Zhou, X. Cui, T. Lu, Y. Zhen, and Z. Luo, "A time-efficient method for the simulation of ion flow field of the ac-dc hybrid transmission lines," *IEEE Trans. Magn.*, vol. 48, no. 2, pp. 731–734, 2012, ISSN: 0018-9464. DOI: 10.1109/TMAG.2011.2171924.

[92] X. Zhou, T. Lu, X. Cui, Y. Zhen, and G. Liu, "Simulation of ion-flow field using fully coupled upwind finite-element method," *IEEE Trans. Power Del.*, vol. 27, no. 3, pp. 1574–1582, 2012, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2012.2197226.

[93] Z. Du, G. Huang, J. Ruan, G. Wang, Y. Yao, C. Liao, J. Yuan, and W. Wen, "Calculation of the ionized field around the dc voltage divider," *IEEE Trans. Magn.*, vol. 49, no. 5, pp. 1933–1936, 2013, ISSN: 0018-9464. DOI: 10.1109/TMAG.2013.2241405.

[94] X. Zhou, T. Lu, X. Cui, Y. Liu, and X. Li, "Simulation of ion-flow field at the crossing of hvdc and hvac transmission lines," *IEEE Trans. Power Del.*, vol. 27, no. 4, pp. 2382–2389, 2012, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2012.2207409.

[95] B. Zhang, W. Li, J. He, and R. Zeng, "2d/3d hybrid computation of ion flow field around house near hvdc bipolar transmission lines," in *14th Biennial IEEE Conference on Electromagnetic Field Computation (CEFC), 2010*, 2010, pp. 1–1. DOI: 10.1109/CEFC.2010.5481520.

[96] Y. Zhen, X. Cui, T. Lu, Z. Luo, and X. Zhou, "Total electric field calculation of body model under hvdc transmission lines," in *Power and Energy Engineering Conference (APPEEC), 2011 Asia-Pacific*, 2011, pp. 1–4. DOI: 10.1109/APPEEC.2011.5748583.

[97] M. F. Kacamarga, B. Pardamean, and J. Baurley, "Comparison of conjugate gradient method and jacobi method algorithm on mapreduce framework," *Applied Mathematical Sciences*, vol. 8, no. 17, pp. 837–849, 2014.

[98] V. Jalili-Marandi and V. Dinavahi, "Simd-based large-scale transient stability simulation on the graphics processing unit," *IEEE Trans. Power Syst.*, vol. 25, no. 3, pp. 1589–1599, 2010, ISSN: 0885-8950. DOI: 10.1109/TPWRS.2010.2042084.

[99] Z. Zhou and V. Dinavahi, "Parallel massive-thread electromagnetic transient simulation on gpu," *IEEE Trans. Power Del.*, vol. 29, no. 3, pp. 1045–1053, 2014, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2013.2297119.

[100] H. Karimipour and V. Dinavahi, "Extended kalman filter-based parallel dynamic state estimation," *IEEE Trans. Smart Grid*, vol. 6, no. 3, pp. 1539–1549, 2015, ISSN: 1949-3053. DOI: 10.1109/TSG.2014.2387169.

[101] ——, "Parallel relaxation-based joint dynamic state estimation of large-scale power systems," *IET Generation, Transmission Distribution*, vol. 10, no. 2, pp. 452–459, 2016, ISSN: 1751-8687. DOI: 10.1049/iet-gtd.2015.0808.

[102] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3. The Johns Hopkins University Press, 1989.

[103] ATCOElectric. (2016). Structures, [Online]. Available: http://www.atcoelectric.com/Projects/HVDC/Project-Details/Structures.

[104] X. Song, V. Pickert, B. Ji, R. T. Naayagi, C. Wang, and Y. Yerasimou, "Questionnaire-based discussion of finite element multi-physics simulation software in power electronics," *IEEE Trans. Power Electron.*, pp. 1–1, 2017, ISSN: 0885-8993. DOI: 10.1109/TPEL.2017.2756449.

[105] Y. Huangfu, S. Wang, L. D. Rienzo, and J. Zhu, "Radiated emi modeling and performance analysis of a pwm pmsm drive system based on field-circuit coupled fem," *IEEE Trans. Magn.*, vol. 53, no. 11, pp. 1–4, 2017, ISSN: 0018-9464. DOI: 10.1109/TMAG.2017.2705079.

[106] E. Melgoza, R. Escarela-Perez, J. L. Guardado, and M. A. Arjona-López, "Strong coupling of an electromagnetic transients program and a finite element magnetic field solver including eddy currents," *IEEE Trans. Power Del.*, vol. 32, no. 3, pp. 1414–1421, 2017, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2016.2604225.

[107] W. Liang, J. Wang, T. Lu, and W. Fang, "A new method for multiple finite-element models in cosimulation with electrical circuit using machine multiloop modeling scheme," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 6583–6590, 2014, ISSN: 0278-0046. DOI: 10.1109/TIE.2014.2314053.

[108] J. Wu, J. Wang, C. Gan, Q. Sun, and W. Kong, "Efficiency optimization of pmsm drives using field-circuit coupled fem for ev/hev applications," *IEEE Access*, vol. 6, pp. 15 192–15 201, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2813987.

[109] B. S. Umesh and K. Sivakumar, "Pole-phase modulated multiphase induction motor drive with reduced torque ripple and improved dc link utilization," *IEEE Trans. Power Electron.*, vol. 32, no. 10, pp. 7862–7869, 2017, ISSN: 0885-8993. DOI: 10.1109/TPEL.2016.2634092.

[110] F. Xu, V. Dinavahi, and X. Xu, "Parallel computation of wrench model for commutated magnetically levitated planar actuator," *IEEE Trans. Ind. Electron.*, vol. 63, no. 12, pp. 7621–7631, 2016, ISSN: 0278-0046. DOI: 10.1109/TIE.2016.2592866.

[111] J.-M. Jin, *The finite element method in electromagnetics, third edition*. John Wiley & Sons, 2015.

[112] W. Wang, R. Xu, H. Li, Y. Liu, X. Guo, Y. Xu, H. Li, H. Zhou, and W. Yin, "Massively parallel simulation of large-scale electromagnetic problems using one high-performance computing scheme and domain decomposition method," *IEEE Trans. Electromagn. Compat.*, vol. 59, no. 5, pp. 1523–1531, 2017, ISSN: 0018-9375. DOI: 10.1109/TEMC.2017.2656891.

[113] P. Liu and V. Dinavahi, "Matrix-free nodal domain decomposition with relaxation for massively parallel finite-element computation of em apparatus," *IEEE Transactions on Magnetics*, vol. 54, no. 9, pp. 1–7, 2018, ISSN: 0018-9464. DOI: 10.1109/TMAG.2018.2848622.

[114] M. Unno, S. Aono, and H. Asai, "Gpu-based massively parallel 3-d hie-fdtd method for high-speed electromagnetic field simulation," *IEEE Trans. Electromagn. Compat.*, vol. 54, no. 4, pp. 912–921, 2012, ISSN: 0018-9375. DOI: 10.1109/TEMC.2011.2173938.

[115] M. Namdari, M. Khosravi-Farsani, R. Moini, and S. H. H. Sadeghi, "An efficient parallel 3-d fdtd method for calculating lightning-induced disturbances on overhead lines in the presence of surge arresters," *IEEE Trans. Electromagn. Compat.*, vol. 57, no. 6, pp. 1593–1600, 2015, ISSN: 0018-9375. DOI: 10.1109/TEMC.2015.2457451.

[116] Z. Zhou and V. Dinavahi, "Fine-grained network decomposition for massively parallel electromagnetic transient simulation of large power systems," *IEEE Power Energy Technol. Syst. J.*, vol. 4, no. 3, pp. 51–64, 2017, ISSN: 2332-7707. DOI: 10.1109/JPETS.2017.2732360.

[117] N. Lin and V. Dinavahi, "Exact nonlinear micro-modeling for fine-grained parallel emt simulation of mtdc grid interaction with wind farm," *IEEE Trans. Ind. Electron.*, pp. 1–1, 2018, ISSN: 0278-0046. DOI: 10.1109/TIE.2018.2860566.

[118] P. Liu and V. Dinavahi, "Real-time finite-element simulation of electromagnetic transients of transformer on fpga," *IEEE Transactions on Power Delivery*, vol. 33, no. 4, pp. 1991–2001, 2018, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2018.2812753.

[119] R. W. Lewis, K. Morgan, H. Thomas, and K. N. Seetharamu, *The finite element method in heat transfer analysis*. John Wiley & Sons, 1996.

[120] N. Lin and V. Dinavahi, "Dynamic electro-magnetic-thermal modeling of mmc-based dc–dc converter for real-time simulation of mtdc grid," *IEEE Transactions on Power Delivery*, vol. 33, no. 3, pp. 1337–1347, 2018, ISSN: 0885-8977. DOI: 10.1109/TPWRD.2017.2774806.

[121] M. Zhang, A Courtay, and Z. Yang, "An improved behavioral igbt model and its characterization tool," in *Proceedings 2000 IEEE Hong Kong Electron Devices Meeting (Cat. No. 00TH8503)*, IEEE, 2000, pp. 142–145.

[122] R Wachal, A Jindal, S Dennetiere, H Saad, O Rui, S Cole, M Barnes, L Zhang, Z Song, J Jardini, *et al.*, "Guide for the development of models for hvdc converters in a hvdc grid," *Cigré TB604 (WG B4. 57), Paris, Tech. Rep*, 2014.

[123] ABB. (2019). Pak, [Online]. Available: https://new.abb.com/semiconductors/stakpak.

[124] M. Hagiwara and H. Akagi, "Control and experiment of pulsewidth-modulated modular multilevel converters," *IEEE Trans. Power Electron.*, vol. 24, no. 7, pp. 1737–1746, 2009, ISSN: 0885-8993. DOI: 10.1109/TPEL.2009.2014236.

A

A.1 Simulation Parameters in Chapter 4

- Preisach model: $a_0 = a_1 = 0$, $b_0 = b_1 = 3162$, $c = 0.6$.
- Transformer parameters: The transformer size is 1.3m×1.8m for the outer rectangle and 0.7m×1.2m for the inner. The coil size is 0.3m×0.5m, and the number of coil turns is 200 for the primary side and 1076 for the secondary. $\mathrm{R_p} = 0.667\Omega, \mathrm{R_s} = 3.588\Omega$ and $\sigma^e = 1000$.
- Case study parameters: $\mathrm{V_{ac}} = 37.5\sqrt{2}\sin(120\pi t)$kV, $\mathrm{R_1}$=5Ω, $\mathrm{L_1}$=2mH, $\mathrm{R_{L1}}$=200Ω, $\mathrm{R_{L2}}$=100Ω, $\mathrm{L_{L1}}$=10mH and $\mathrm{L_{L2}}$=5mH. The length of the transmission line TL is 15km, travelling time is 50μs and characteristic impedance Z_{TL} is 200Ω. The magnitude of the injected second and fourth harmonics are 9.38kV and 4.69kV respectively.
- Comsol™ simulation parameters: (1) For the finite element solution of the magnetic field with nonlinear B-H curve, the winding zone is modeled as the multi-turn coil, the nonlinear method is set to the automatic Newton method, the termination technique utilizes a tolerance factor of 0.001 or a maximum iteration number of 25, and other settings such as damping factor remain default. (2) For the external circuit, the external I-VS-U is used to interface with the magnetic field, the transmission line is modeled as a series of equivalent Π models, the switches are modeled as two-state time-varying resistors defined by some piecewise functions, and other events like harmonics injections are also implemented with time-varying functions.

B

B.1 Simulation Parameters in Chapter 5

- Computation Resources: The GPU version is GeForce GTX Titan Black, with 2880 cores, 889MHz clock frequency, and 4GB memory. The CPU version is Intel E5-2620, with 16 cores, 2.1GHz clock frequency, and 32GB memory.

C

C.1 Multi-Physics Simulation Parameters in Chapter 6

- Material parameters: for copper, $\alpha = 3.8 \times 10^{-11}\Omega m/K$ and $\sigma_0 = 5.8 \times 10^7 S/m$; and for steel, $\alpha = 5 \times 10^{-11}\Omega m/K$ and $\sigma_0 = 9.6 \times 10^6 S/m$.
- MMC parameters: voltage level 5-513, DC voltage V_{dc}=$\pm$100kV, rated power 500MW, AC voltage V_{ac}=280kV; Submodule capacitor 3mF, arm inductance 50mH; DC line parameters: length 100km, l=0.05mH/km, r=0.012Ω/km, c=0.015μF/km.
- Transformer parameters: The transformer size is $5.2m \times 3.6m$ for the outer rectangle and $1.85m \times 2.6m$ for the two inner rectangles. The coil size is $0.25m \times 2m$, and the number of coil turns is 418 for the primary side and 200 for the secondary.

C.2 Wide-Band Transformer Parameters in Chapter 6

In Fig. 6.14, R_p=1.2mΩ, R_s=0.35mΩ, L_p=4.6850H, L_s=1.3836H, L_m=-1.0680H, $C_p = C_s$=104pF, and C_{ps}=812pF.