UNIVERSITY OF CALIFORNIA, SAN DIEGO

Fast Algorithms and Solvers in Computational Electromagnetics and Micromagnetics on GPUs

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in
Electrical Engineering (Photonics)

by

Shaojing Li

Committee in charge:

Professor Vitaliy Lomakin, Chair
Professor Yeshaiahu Fainman
Professor Eric E. Fullerton
Professor Carl H. Gibson
Professor Ross C. Walker

2012

The dissertation of Shaojing Li is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2012

DEDICATION

献给一直支持我的

爸爸李长真，妈妈邵燕琼

和

我亲爱的梁文琴

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

It is my great pleasure to have this chance, after more than twenty years of formal education, to thank the people who have helped me all the way. Whilst I owe great gratitude to many people, I especially want to thank my advisor, Prof. Vitaliy Lomakin, for his continued guidance and encouragements during my five years of graduate studies. From the time I first came to the United States, I have received countless help and support from him in academic research as well as in everyday life. It is my privilege to have met such an advisor that has exceptional intellectual ability while at the same time talk to his students just like their peers. He has been involved in all the work presented in this thesis, from the early brainstorming stage, to the final analysis and discussions. Without his tremendous help, I would have never been able to reach the current stage of my career and the thesis could have never been completed.

I would also like to thank Prof. Eric Fullerton who taught most of my knowledge in magnetic recording systems and magnetic materials. I can still remember the day he came into the classroom, smiling, being proud to tell us that the 2008 Nobel Physics Prize was awarded to discoverers of the GMR effect. His interesting and funny classes have always been my favorite. My gratitude also goes to Prof. H. Neal Bertram, who I am fortunate to meet before I knew anything in the field of micromagnetics. I am also grateful for the encouragements on numerous occasions from Prof. Thomas Schrefl

(Austria), Dr. Ganping Ju (Seagate) and Dr. Jan van Ek (Western Digital Corporation).

I treasure my friendship with all my labmates I have during my stay in Room 3507 of the Jacobs Hall. Dr. Boris Livshitz gave me much help in my early days and a part of the data in this thesis is obtained using the micromagnetic solver he wrote. Many thanks to Dr. Derek van Orden, who I enjoyed talking to about anything from complex Green's function to the great time you had in France. My greate appreciation goes to Ruinan Chang, Marko Lubarda, Marco Escobar, Javier Martin and Sidi Fu. Many times we learned together and fought through obstacles and bottlenecks in various projects. It my great pleasure to have known and worked with you all.

Finally, my deepest gratitude goes to my parents, Changzhen Li and Yanqiong Shao, and my beloved Wenqin Liang. The love and support you have been giving me is unconditional.

Part of the materials presented in this thesis have been published or is accepted for future publication on refereed journals.

Chapter 4 contains materials from the following papers:

- S. Li, B. Livshitz, and V. Lomakin, "Fast evaluation of Helmholtz potential on graphics processing units (GPUs)," Journal of Computational Physics, vol. 229, no. 22, pp. 8463-8483, 2010.

- S. Li, R. Chang, and V. Lomakin, "Chapter 19 - Fast Electromagnetic Integral Equation Solvers on Graphics Processing Units," GPU Computing Gems Jade Edition, W. H. Wen-mei, ed., pp. 243-266, Boston: Morgan Kaufmann, 2012.
- S. Li, R. Chang, and V. Lomakin, "Fast integral equation solvers on Graphics Processing Units for Electromagnetics," IEEE Antennas and Propagation Magazine, to appear in 2013.

Chapter 5, in part, is reprint, with some minor modifications for the clarity, of the materials as it appears in

- S. Li, D. A. Van Orden, and V. Lomakin, "Fast periodic interpolation method for periodic unit cell problems," Antennas and Propagation, IEEE Transactions on, vol. 58, no. 12, pp. 4005-4014, 2010.

Chapter 6 contains results from:

- M. A. Escobar, M. V. Lubarda, S. Li, R. Chang, B. Livshitz, and V. Lomakin, "Advanced Micromagnetic Analysis of Write Head Dynamics Using Fastmag," Magnetics, IEEE Transactions on, no. 99, pp. 1-1, 2012.
- R. Chang, S. Li, M. Lubarda, B. Livshitz, and V. Lomakin, "FastMag: Fast micromagnetic simulator for complex magnetic structures," Journal of Applied Physics, vol. 109, no. 7, pp. 07D358-07D358-6, 2011.

Chapter 7 consists of results and discussion from papers:

- S. Li, B. Livshitz, H. N. Bertram, E. E. Fullerton, and V. Lomakin, "Microwave-assisted magnetization reversal and multilevel recording in composite media," Journal of Applied Physics, vol. 105, no. 7, pp. 07B909-07B909-3, 2009.

- S. Li, B. Livshitz, H. N. Bertram, A. Inomata, E. E. Fullerton, and V. Lomakin, "Capped bit patterned media for high density magnetic recording," Journal of Applied Physics, vol. 105, no. 7, pp. 07C121-07C121-3, 2009.

VITA

2009 – 2012 Ph. D. in Electrical Engineering (Photonics), University of California, San Diego

2007 – 2009 M. Sc. in Electrical Engineering (Photonics), University of California, San Diego

2003 – 2007 B. Sc. in Electrical Engineering, Wuhan University

PUBLICATIONS

[1] S. Li, R. Chang, V. Lomakin, "Fast Electromagnetic Integral Equation Solvers on Graphics Processing Units", *IEEE Antennas and Propagation Magazine*, accepted for publication, 2013.

[2] M. V. Lubarda, M. A. Escobar, S. Li, R. Chang, E. E. Fullerton, and V. Lomakin, "Domain wall motion in magnetically frustrated nanorings", *Physics Review B*, vol. 85, no. 21, p. 214428, 2012.

[3] M. A. Escobar, M. V. Lubarda, S. Li, R. Chang, B. Livshitz, and V. Lomakin, "Advanced Micromagnetic Analysis of Write Head Dynamics Using FastMag," *IEEE Transactions on Magnetic*, vol. 48, no. 5, pp. 1731-1737, 2012 (invited)

[4] R. Chang, S. Li, M. A. Escobar, M. V. Lubarda, and V. Lomakin, "Accurate Evaluation of Exchange Fields in Finite Element micromagnetic solvers," *Journal of Applied Physics*, vol. 111, p. 07D129, 2012

[5] S. Li, R. Chang, V. Lomakin, “Fast Electromagnetic Integral Equation Solvers on Graphics Processing Units”, *GPU Computing Gems Jade Edition Chapter 19*, Elsevier, 2011

[6] M. Lubarda, S. Li, B. Livshitz, E. E. Fullerton, and V. Lomakin, "Antiferromagnetically-coupled capped bit patterned media for high-density magnetic recording", *Applied Physics Letters*, vol. 98, p. 012513, 2011.

[7] R. Chang, S. Li, M. V. Lubarda, B. Livshitz, and V. Lomakin, "FastMag: Fast micromagnetic solver for large-scale simulations", vol. 109, p. 07D358, *Journal of Applied Physics*, 2011 (invited).

[8] M. Lubarda, S. Li, B. Livshitz, E. E. Fullerton, and V. Lomakin, "Reversal in Bit Patterned Media With Vertical and Lateral Exchange", *IEEE Transcations on Magnetics*, vol. 47, no. 1, pp. 18-25, 2011 (invited).

[9] S. Li, B. Livshitz, and V. Lomakin, "Fast evaluation of Helmholtz potential on graphics processing units (GPUs)", *Journal of Computational Physics*, vol. 229, no. 22, pp. 8463-8483, 2010.

[10] S. Li. D. A. Van Orden, and V. Lomakin, "Fast periodic interpolation method for periodic unit cell problems", *IEEE Transactions on Antennas and Propagation*, vol. 58, no. 12, pp. 4005- 4014, 2010.

[11] S. Li, B. Livshitz, and V. Lomakin, "Graphics Processing Unit accelerated O(N) micromagnetics solver", *IEEE Transactions on Magnetics*, vol. 46, no. 6, pp. 2373 – 2375, 2010.

[12] S. Li, B. Livshitz, H. N. Bertram, M. Schabes, T. Schrefl, E. Fullerton, and V. Lomakin, "Microwave assisted magnetization reversal in composite media", *Applied Physics Letters*, vol. 94, p. 202509, 2009.

[13] S. Li, B. Livshitz, H. N. Bertram, A. Inomata, E. E. Fullerton, and V. Lomakin, "Capped bit patterned media for high density magnetic recording", *Journal of Applied Physics*, vol. 105, p. 07C121, 2009.

[14] S. Li, Boris Livshitz, H. N. Bertram, E. E. Fullerton, and V. Lomakin, "Microwave assisted magnetization reversal and multilevel recording in composite media", *Journal of Applied Physics*, vol. 105, p. 07B909, 2009.

[15] V. Lomakin, S. Li, B. Livshitz, A. Inomata, and H. N. Bertram, "Patterned media for 10 Tbit/in2 utilizing dual-section "ledge" elements," *IEEE Transactions on Magnetics*, vol. 44, no. 11, pp. 3454-3459, 2008 (invited).

[16] V. Lomakin, R. Choi, B. Livshitz, S. Li, A. Inomata, and H. N. Bertram, "Dual-layer patterned media ledge design for ultra-high density magnetic recording", *Applied Physics Letters*, vol. 92, p. 022502, 2008.

ABSTRACT OF THE DISSERTATION

Fast Algorithms and Solvers in Computational Electromagnetics and Micromagnetics on GPUs

by

Shaojing Li

Doctor of Philosophy in Electrical Engineering (Photonics)

University of California, San Diego, 2012

Professor Vitaliy Lomakin, Chair

In this thesis, fast algorithms for solving fields defined by the Helmholtz equation using integral equation methods are developed and implemented on Graphics Processing Units (GPUs). GPUs are massively parallel processors that offer tens or even hundreds of times of floating point computing capability to current generation CPUs. A short history of the GPUs is given and their unique architecture is described in details. On this new hardware architecture, algorithms like the hierarchical Non-uniform Grid Interpolation Method (NGIM) and the FFT-based Adaptive Integral

Method (AIM) have to be significant changed from their original sequential forms to achieve high performances. Specifically, the computational domains of the problems are divided into boxes, homogenizing the computing burdens across the wide SIMD-style stream multiprocessors. Computing operations are reformed and reorganized to exploit the enormous floating point computing power and while at the same time to minimize the data transfer latencies. The achieved computing performance on commercial GPUs is generally two orders of magnitude higher than that on state-of-the-art CPUs and with much lower memory consumption.

Based on these fast algorithms, an ultra-fast micromagnetic solver with linear or $O\left(N \log N\right)$ computational complexity is built. This solver, named FastMag, runs on desktop workstations with one or several GPU cards and is able to simulate magnetic systems with over one hundred million degrees of freedom. Electromagnetic solvers that use slightly different algorithms are also implemented and provide impressive performance on general electromagnetic problems such as wave scattering. This electromagnetic solver is also capable of handling periodic boundary problems using a new algorithm called the Fast Periodic Interpolation Method (FPIM). This algorithm significantly uses spatial interpolations as well as the FFT to reduce the time of evaluating fields generated by infinitely periodic structures.

Using previously developed micromagnetic solvers, the author investigated two novel magnetic recording systems that might be useful in the next generation ultra-high

density magnetic recording. The capped bit-patterned media (CBPM) are proposed to have lower reversal fields, lower switching field distribution as well as better readback signals. The reversal mechanisms of bit-patterned media under the influence of microwaves are also investigated. This leads to the proposed multi-layer recording system using the microwave-assisted magnetic recording (MAMR) technology.

1 Introduction

This chapter describes the motivations of the work presented in this thesis, the necessary background of the work, the current status of the field and author's contributions to the area.

1.1 Importance of numerical simulations to science and engineering

Numerical simulation is one the most important tools for the human beings to understand the physical world. Contrary to other approaches, such as experiments and analytical reasoning, simulations as a general approach for solving scientific problems appeared much later. It became widely used only after the emergence of modern electronic computers. Though it has a relatively short history, it has some unique features that make it irreplaceable.

Many real world phenomena can be modeled with fundamental physics laws, expressed mathematically using one or a system of ordinary differential equations (ODEs) or partial differential equations (PDEs). In particular, in the fields of computational micromagnetics and electromagnetics, the Landau-Lifshitz-Gilbert (LLG) equation and the Helmholtz equation are PDEs and numerically can be cast as ODEs. With proper initial or boundary conditions, the solutions of these differential equations could predict the behavior of actual physical systems.

However, for practical systems, few of these ODEs or PDEs can be solved analytically, so modeling realistic systems requires using numerical methods to solve the underlying equations. Furthermore, in many situations, performing numerical simulations has advantages over experiments as a power predictive tool. There are also cases that experiment in which experiments cannot be a viable option, e.g. when destructive or hazardous experiments are required.

1.2 Acceleration of numerical simulation

Scientists and engineers in all disciplines have sought ways to reduce the computing resources, such as the processor time and memory, used by their numerical simulators. Faster simulator can reduce the overall cost of research, produce more results within a fixed period of time or simulate larger or more accurate models of a system. A complete list of techniques to accelerate the simulation of a problem is far beyond the scope of this thesis, but for a certain types of problems that will be discussed in the field of computational electromagnetics and micromagnetics, general guidelines can be made on where and how to explorer the opportunities for acceleration. Similar approaches can also be used for a set of models in many other fields of computational physics.

For all the problems that will be discussed in this thesis, physical systems are modeled by a set of coupled differential equations, which can be often further cast into an integral equation form. Solutions of these equations follow a general procedure. First, the computational domain of interests is discretized both temporally and spatially, and the continuous differential or integral operators are emulated by their discrete counterparts. There are a number of ways for such discretization, such as Finite Difference methods (FD) [85, 175], finite element method (FEM) [5, 81] and the method of moments (MoMs) for the integral equation (IE) methods [68, 135, 140].

These methods usually express the original problem as a system of linear equations that can be represented by either sparse or dense matrix equations. Then, in a second stage, solution, stage, these matrix equations are solved through either direct matrix inversion methods [137] or iterative methods [60]. Finally, the obtained solutions can either be output to in proper visualized form or be plugged into the simulator again for other calculations or optimizations.

To accelerate the aforementioned process, multiple options are available and most efficient simulators most likely use many of them aiming at accelerating the most time consuming, *bottlenecks*, stages.

One obvious way to accelerate the computation is to reduce the number of independent variables or parameters to be solved. This can be achieved by modeling the original physical problems with coarser discretization, everywhere in the domain or

in appropriate areas, leading to fewer *degrees of freedom* in the matrix-vector equations. In this thesis, the term *"degrees of freedom"* is also referred to as *"unknowns"*, *"problem size"* or sometimes *"number of equations"*.

Another approach of acceleration is to solve the presented linear equation system using *"fast algorithms"* that use less number of operations than the straightforward direct inversion or iterative solution techniques would need. Following this approach usually requires identifying and utilizing certain characteristics of the corresponding matrices generated by a specific modeling method. The Adaptive Integral Methods (AIM) [11, 136], Fast Multipole Methods (FMMs) [34, 61], Non-uniform Grid Interpolation Methods (NGIMs) [16, 103]and H2-Matrix Approximation Methods [64, 65] are a few examples among many other. These fast algorithms can often be expressed as a series of complex matrix transform operations applied to the matrix equations.

Computational methods are developed for using digital computers and they must account for the hardware features. For several decades the computational power increase largely relied on the increase of the processor speed. However, the speed of single core systems has saturated due to fundamental physical limitations [23, 74, 78]. Further improvement of performance of computers should rely on the adoption of parallel computer systems, such as multi-core CPUs [55, 75], alternative many-core architectures [4, 79, 101, 130, 150] and heterogeneous computing architectures [3, 21,

23]. These hardware architecture developments have a major impact on the design of computational algorithms.

Considering all these aspects, we can see that accelerating numerical solvers usually involves extensive domain knowledge of modeling physics phenomena in sets of mathematical equations, capabilities to design and implement fast algorithms efficiently as computer programs and in-depth knowledge of computer architectures for the algorithms being adapted to state-of-the-art hardware and software. In this thesis, we will focus primarily on the latter two areas and describe the designing, optimizing and benchmarking processes of several fast algorithms on the General Purpose Graphics Processing Units (GPGPUs).

1.3 Many-core and heterogeneous computing architectures

1.3.1 Massive parallelization architectures

Many-core computing architectures emerged as an important technology for scientific computing less than ten years ago and it is becoming an increasing plausible platform of choice for numerical computing. With the instruction level parallelism (ILP) provided by compilers or hardware control logic being exhausted, chip designers decided to replicate multiple cores inside one chip and leave the higher level logic to utilize the thread-level or data-level parallelism [74, 78].

Historically, massive parallelization was the strategy of choice to scale the computation performance of scientific simulations to multiple tera- or peta-FLOPS range. The research area "high-performance computing" (HPC) deals mostly with computer clusters that consist of multiple computer nodes. In this regime, homogeneous high-performance computing nodes are built on processors responsible for all computing tasks, regardless of their nature. These nodes are connected through relatively slow network connections and interact with each other via certain message passing mechanism such as the Message Passing Interfaces (MPIs). Contrary to the traditional HPC approaches, newly emerged many-core processor like Graphics Processing Units (GPUs), IBM's Cell Broadband Engine (CBE) and Field Programmable Gate Arrays (FPGAs) allow for massive parallelization happens even within a single computing device. In particular, a single GPU card can contain hundreds of cores, e.g. 1536 cores (or stream processors) on NVIDIA GeForce GTX 680 card [130] and 2048 processors on AMD Radeon 7970 HD card [4]. Each of the processor cores can run several permitting for tens of thousands individual threads in parallel [127]. Multiple GPUs can be installed on a single workstation or a single cluster node.

1.3.2 Unique memory architecture

In addition to the unique processor architecture, the aforementioned many-core systems also have complex memory architectures to deal with the so-called *Memory Wall Problem* [116].

The cause of memory wall problem is closely related to the concept called arithmetic intensity, which is defined as the average number of arithmetic operations per memory access. For algorithms that have very low arithmetic intensity, the performance does not scale with the number of processor, because it is the latency of memory access that limits the overall arithmetic throughput. With the emergence of multi-TFLOPS single chip processors like the GPUs, this effect is magnified as the speed of GPU memory is only slightly faster than that of CPUs with a much fewer cores.

Therefore, GPUs employ more complex multi-level memory hierarchies including the shared memory, constant memory, L1/L2 cache and texture memory [101, 125, 127]. Utilizing these different types of memory against different types of tasks solely relies on the programmer and is critical for achieving high performance from any numerical algorithm to run on GPUs. To keep these high-speed caches and shared memory efficiently used, data reuse has to be maximized by improving the spatial and temporal data access locality. There are also other features provided by specific vendors, like the coalesced memory access and arithmetic/memory instruction overlapping to further accelerate the data access throughput.

1.4 Hardware-adapted algorithm design

All the aforementioned hardware architectures require much more efforts from the programmers than just expressing their formulas in high level languages and letting the compilers to do the rest. Architects of a simulator should consider not only the theoretical computational complexity of the algorithm but also how the algorithms can be matched to these computing architectures. Programmers have to utilize task or data parallelism at a much finer scale and try to increase the data locality as much as possible. Sometimes, they may face a trade-off between parallel efficiency and computational complexity and one might need to adopt unconventional techniques such as trading memory access with extra numerical computations to break the memory wall.

In this work, the hypothesis is that for various algorithms in the fields of computational electromagnetics and micromagnetics, high performance can only be achieved by simultaneously reducing the computational complexity and adapting the algorithms to the hardware. We believe researcher in all the numerical computing areas must pay close attention to the technology trends in all layers of the computing platform, from the very low-level hardware processor arrangement to mid-level runtime library and to high-level task-level parallelization.

1.5 Summary of contributions

The main contribution of the thesis is to demonstrate that GPUs and other similar emerging many-core accelerator architectures are promising platforms for scientific computing applications. This is especially true for areas that involve high floating point operation intensity such as computational electromagnetics and micromagnetics.

The thesis also gives important tips and hints on how existing algorithms should be modified to accommodate GPUs. The algorithms described, implemented and benchmarked show orders of magnitude improvements in speed and memory consumption over compiler-optimized single thread sequential code. It is also shown that GPUs, as a typical and widely used many-core computing architecture, are very effective for computational electromagnetics and micromagnetic applications. However, in order to obtain significant speed-ups, developers of numerical simulation software have to keep the hardware architecture in mind, in order to write code ready to scale on to the large number of cores with heterogeneous processor and memory configurations.

The thesis also shows several simulators utilizing fast algorithms on GPUs. The effectiveness of GPU-accelerated fast algorithms is demonstrated by complex magnetic recording simulations, perpendicular magnetic write head simulations and the electromagnetic wave scattering simulations.

Employing the ultra-efficient micromagnetic solver built on those GPU-accelerated fast algorithms, the author proposed several possible recording media designs for the next generation magnetic recording systems and verifies them by

computer simulations. The first set of simulations deal with a novel design of bit patterned media (BPM). The other use case is the simulation and verification of the microwave-assisted magnetic recording (MAMR) system. The simulation of the BPM and the MAMR applied on multilayer patterned media arrays shows interesting physical phenomena might help the design of the next generation hard-disk drive..

1.6 List of publications

During the past five years, the author has published several peer-review papers on several journal and made many presentations on conferences. Since the author started to do researches since the relatively early stage of GPUs emerged as a disruptive technology, many results shown in this are slightly outdated. Therefore, the readers should be careful when comparing the absolute numbers listed in the paper with those published much later and on more advanced hardware platforms.

The author of this thesis and those publications conducted his research with many collaborators. Consequently, all of the papers listed described below as well as this thesis involve extremely large amount of cooperative work. The author would like to acknowledge all the co-authors listed on all the publications, in various academic institutions and industrial partners.

1.6.1 Book chapters

In the book titled *"GPU Computing Gems: Jade Edition"*, published by Elsevier in 2011, the chapter 19 named *"Fast electromagnetic integral equation solvers on graphics processing units"* described the electromagnetic solvers based on Non-uniform Interpolation Method (NGIM) on GPU [93]. The NGIM is a hierarchical multilevel fast algorithm that reduces the quadratic computational complexity of general iterative integral equation solvers to linear or O(NlogN) complexity. The section 4.2 will discuss this NGIM algorithm in more details with updated results and detailed analysis.

1.6.2 Journal articles

Currently, 13 journal articles have been published or accepted for publication under the author's name. Among them, 8 papers are on the topic of fast algorithms and numerical techniques for high performance simulation solvers on GPUs and the rest are in the area of design and analysis of the magnetic recording system.

The first two papers published in 2008 discussed a novel design for ultrahigh density magnetic recording system [106, 107]. The papers discussed a dual-layer patterned element magnetic recording medium design that has two layers of different sizes stacked in the vertical directions. The magnetic behavior of this design has been investigated by simulations. From the simulations, various interesting behaviors induced by the prolonged soft upper layer and the ferromagnetic coupling between layers have been observed and it was found that the reversal field of such magnetic medium can be much lower than other single and multilayer design that have the same thermal stabil-

ity. Following these papers, the proposed media have been fabricated and tested experimentally [20, 25, 58].

In 2009, the author published two papers on the topic of the microwave-assisted magnetic recording [95, 97]. In these papers, the author revealed various interesting physical phenomena when a double-layer ferromagnetically coupled pattern media element is excited by applied microwave field. In the second paper, the possibility of multi-layer recording was discussed [95]. The content of these papers will be presented and discussed again in the Chapter 7 of this thesis. In addition, the author also published a paper proposing another magnetic recording media design, called the "capped media" in 2009 [96]. This paper also directly inspired another paper published in 2011 on Applied Physics Letters [113]. Capped media can be seen as an extension and improvement of the "ledge" media and the AFC-capped media is the design with further optimized performance. Following these papers, capped bit patterned media have been fabricated and their properties have been experimentally tested [84, 138, 139].

In 2010, the author started publishing extensively in the field of numerical computing and GPU algorithms and computing. Three papers were out. The first one in the IEEE Transactions on Magnetics describes micromagnetic solver on GPU with linear computational complexity [99], which is the early prototype of the FastMag solver that later published one year later in 2011 and is being used internally and externally in academia and industry. The next paper titled *"Fast periodic interpolation*

method for periodic unit cell problems" [100] describes a fast algorithm that can make and integral equation based electromagnetic solvers several orders of magnitude faster for problems with periodic boundary conditions. The new algorithm uses extensive spatial interpolations to accelerate the computation of fields and it can be efficiently parallelized on many-core architectures. The third paper in 2011, published in the Journal of Computational Physics describes the non-uniform interpolation methods (NGIMs) [15, 16, 102] on GPU in details [98]. The paper illustrates several key points for the NGIM to have high efficiency running on a single GPU card. This is the first published efficient implementation of a hierarchical fast method for integral solvers of Helmholtz equation. The key contributions of the paper are the unique approach of domain subdivision, the task distribution across stream processors and the *on-the-fly* interpolation calculation.

In 2011, the author published three papers in collaboration with other students in the lab. The paper titled *"FastMag: Fast micromagnetic solver for large-scale simulations"* summarized features of the FastMag micromagnetic solver [31]. The FastMag simulator is extremely fast and has the capability of handling very large and complex magnetic systems. The solver can efficiently solve ultra-complex problems on a desktop workstation utilizing the fast algorithms on GPUs designed by the author. Several other papers showed a few simulations running on the FastMag simulator and interesting phenomena have been observed leading to new and improved designs for various

magnetic systems. The papers on AFC-capped BPM extend the study of the "capped bit pattern media" and introduced an antiferromagnetically coupled cap layer that provides better cancellation of magnetostatic interactions between patterned islands [112, 113].

At the time of writing this thesis, two other papers have been published. One paper in the Journal of Applied Physics discussed accuracy of the evaluation of exchange field which is a critical component during the solving of the LLG equation [30]. The other paper on the IEEE Transactions on Magnetics discussed several aspects in the micromagnetic simulations while designing and analyzing magnetic recording systems [49]. The paper reveals that improper discretization would significantly affect the accuracy of the simulations via investigating the effect of wrap around shield (WAS). By using sufficiently dense meshes model the recording head, it is found that WAS improves the head field gradient in both down- and off-track directions but reduces the magnitude of the field. This study shows the capability of the FastMag simulatr by running ultra-complex magnetic systems. This work has further proven the practicality of the GPU acceleration for scientific simulators.

The most recently published paper *"Domain wall motion in magnetically frustrated nanorings"* [111], discussed interesting physical phenomena observed in frustrated magnetic nanorings which might be useful as an alternative storage device in the future.

Another accepted and yet to be published paper on the IEEE Antennas and Propagation Magazines introduces a new fast algorithm called box adaptive integral method (B-AIM) on GPUs. This algorithm also accelerates the field evaluation process in many iterative solvers for Helmholtz equations like NGIM but with slightly different philosophy. B-AIM is a newly designed FFT-based fast algorithm that also runs over 100x faster on a single GPU card versus the sequential implementation for CPUs. This algorithm will be discussed in details in the Section 4.3.

1.6.3 Conference presentations

The author is a regular attendant of various conferences organized by IEEE, APS and ACM, where he presented the new algorithms outlined above in the conferences.

1.7 Outline of the thesis

This thesis presents the scientific findings and engineering innovations the author has done during his PhD studies. Each chapter is mostly self-contained, including a motivation, introduction, and summary. However, the chapters also have many cross-relations between each other.

The document is organized in eight chapters. The first (current) chapter contains general introduction that describes the background of the fields and the

motivations of the author's researches. It also summarizes the contributions of the thesis that will be presented later on and listed the publications and other academic activities the author has involved.

Chapter 2 presents problems the author has been concerned, in a concise but accurate way. This section begins with a description of the equations to be solved in computational micromagnetics and electromagnetics. Then several approaches to reach these equations numerically are described, analyzed and compared. Most of the numerical work is concentrated on the integral equation (IE) or integral superposition approach of solving partial differential equations (PDEs) in the form of either Poission equation or Helmholtz equation.

Chapter 3 introduces the hardware platform the author used, the graphics processing units (GPUs). In the first section of Chapter 3, the history of GPUs is briefly described, including its emergence as special purpose coprocessor, development driven by consumer electronics and the current state of being powerful accelerators that possesses more computing power than CPUs. Then in Section 3.2 and 3.3, the hardware and software architectures of GPUs are described and analyzed. In the final section, the author concludes the chapter with an outlook of future development trends of many-core and heterogeneous computing platforms, including possible new hardware designs, software models, and programming concepts.

Chapter 4 concerns the fast algorithms implemented by the author. Detailed descriptions of three different algorithms that are common and efficient for evaluating various fields encountered in simulations are described and analyzed. Numerical results of each algorithm are shown in each section. Section 4.1 presents a literature review surveying the state-of-the-art at the time of construction of this thesis. Section 4.2 deals with the Non-uniform Grid Integration Methods (NGIM), which belongs to the class of multi-level hierarchical fast algorithms. Section 4.3 describes the Box Adaptive Integral Methods (B-AIM) which is based on FFTs and near field corrections. In Section 4.4, general guidelines for designing algorithms for GPUs are listed and discussed.

Chapter 5 describes Fast Periodic Interpolation Methods (FPIM) which targets a set of problems with infinite periodic boundary conditions.

In Chapter 6, several use cases of our computational electromagnetic and micromagnetic solvers are shown. The solvers is a collaborative effort of several people, including the author's advisor Prof. Vitaliy Lomakin and the author's contributions are mostly in the numerical fast algorithms, parallelization, GPU implementations, performance optimizations, and integration. The use cases include several verification problems that show the capability and validity of developed solvers and two real world simulation projects done for industrial partners of the group.

In Chapter 7, presents the use of micromagnetic solvers for application in the design of magnetic recording system. The material in this chapter is mostly from the previously published papers by the author.

The last chapter summarizes the thesis with observations at a higher-level perspective and also lists several the possible future work directions to improve and extend the existing methods and codes.

2 Problem statement and mathematical outline

This chapter aims to lay the mathematical foundations of the problems to be solved in the field of computational micromagnetics and electromagnetics.

2.1 Numerical solutions of general micromagnetic problems

Micromagnetic simulation deals with various phenomena that happen in the magnetic materials. It was introduced by William F. Brown Jr. in 1963 via the Brown's equation, which is derived by obtaining stationary points from the free energy functional [22]. The magnetization dynamics is described through the Landau-Lifshitz-Gilbert equations [56, 86]. This equation describes the motion of magnetic moments as a damped gyromagnetic precession around the local magnetic field they observe. Equilibrium states can be found by minimizing the magnetic energy of the system. Moreover, minima energy paths between various magnetic states can be found either using minimization methods of the Nudged Elastic Band method [46, 73].

2.1.1 Gibbs free energy

The total Gibbs free energy of a magnetic system is given by [22]

$$E_{total} = \int_{\Omega} \Big(E_{exch} + E_{anis} + E_{zeeman} + E_{demag} \Big) dv \tag{2.1}$$

Within this formula, each energy component can be expressed as:

a) $E_{exch} = \int_{\Omega} A\left((\nabla m_x)^2 + (\nabla m_y)^2 + (\nabla m_z)^2\right)dv$ is the exchange energy, where A is the exchange constant and m_x, m_y, m_z are three components of normalized magnetization in any discretized volume;

b) $E_{anis} = \int_{\Omega} K_1\left(1 - \left(\mathbf{m}\cdot\mathbf{a}\right)^2\right)dv$, where K_1 is the magnetocrystalline anisotropy constant, $\mathbf{m}$ is the normalized magnetization defined as $\mathbf{m} = \mathbf{M} / M_s$ and $\mathbf{a}$ is the unit vector along the easy axis. This expression is for materials with simple uniaxial anisotropy and more complex forms of anisotropy do exist but are not discussed within this thesis;

c) $E_{zeeman} = -\int_{\Omega} \mathbf{M}\cdot\mathbf{H}_{ext}dv$, where $\mathbf{M}$ is the magnetization and $\mathbf{H}_{ext}$ is the external field;

d) $E_{demag} = \int_{\Omega} -\frac{1}{2}\mathbf{M}\cdot\mathbf{H}_{demag}dv$, where $\mathbf{H}_{magneto}$ is the magnetostatic field.

Calculating the equilibrium state of the magnetic system requires finding the minimal total energy. Brown proposed a variational method that calculates the variational derivative of the total energy with respect to the magnetization [22]. So in the equilibrium state,

$$\frac{\delta E_{total}}{\delta \mathbf{M}} = 0 \tag{2.2}$$

This leads to the Brown's equation:

$$\mathbf{M}\times\left(2A\nabla^2\mathbf{m} + 2K_1\mathbf{a}\left(\mathbf{a}\cdot\mathbf{m}\right) + M_s\mathbf{H}_{\mathbf{ext}} + M_s\mathbf{H}_{\mathbf{demag}}\right) = 0 \tag{2.3}$$

This means that at the equilibrium state, the magnetization polarization will align itself with an effective field that can be expressed as

$$\mathbf{H}_{\mathbf{eff}} = \frac{2A\nabla^2\mathbf{m}}{M_s} + \frac{2K_1\mathbf{a}\left(\mathbf{a}\bullet\mathbf{m}\right)}{M_s} + \mathbf{H}_{\mathbf{ext}} + \mathbf{H}_{\mathbf{demag}} \tag{2.4}$$

To solve a large magnetic system, one usually needs to discretize the whole computational domain into many small subdomains, within which all the magnetic properties are assumed to be uniform or have prescribed variations. The above energy calculation is valid for any of these subdomains and the equilibrium state of the entire system can be obtained by minimizing the total energy possessed by all the subdomains. As stated briefly in the Section 1.2, there are many ways of discretization, with the finite difference (FD) and finite element method (FEM) being two most widely used methods. Using any of these methods will generate a system of equations that can be expressed as a matrix equation and its solution will be discussed in later sections.

2.1.2 The Landau-Lifshitz-Gilbert equation

Often the dynamic behavior of the magnetization is of main interest. The dynamics of a single magnetic moment under the influence of a magnetic field is governed by its Larmor precession and the model to describe this motion was proposed by Landau and Lifshitz [86] and later modified by Gilbert [56, 57]. The Landau-Lifshitz equation for modeling the dynamic behavior of a single magnetic moment is as follows:

$$\frac{d\mathbf{M}}{dt} = -\gamma_L \mathbf{M} \times \mathbf{H}_{\mathbf{eff}} - \frac{\beta}{M_s} \mathbf{M} \times \left(\mathbf{M} \times \mathbf{H}_{\mathbf{eff}} \right) \tag{2.5}$$

, where the γ_L and β are precession and damping constant respectively.

The first term of the above equation is the precession term and the second term is an empirical damping term. Later in the 1955, Gilbert proposed a different approach to model the dissipation process and came out with another expression of the damping term. The magnetic dynamics equation with the Gilbert damping term is expressed as follows

$$\frac{d\mathbf{M}}{dt} = -\gamma \mathbf{M} \times \mathbf{H}_{\mathbf{eff}} + \frac{\alpha}{M_x} \mathbf{M} \times \frac{d\mathbf{M}}{dt} \tag{2.6}$$

These two equations are mathematically equivalent, though they are derived from different physical perception. The damping and precession constants in the above two equations can be related via

$$\gamma_L = \frac{\gamma}{1+\alpha^2}, \quad \beta = \frac{\gamma\alpha}{1+\alpha^2} \tag{2.7}$$

The commonly used as the "Landau-Lifshitz-Gilbert equation" (LLG equation) has the following form

$$\frac{d\mathbf{M}}{dt} = -\frac{\gamma}{1+\alpha^2} \mathbf{M} \times \mathbf{H}_{\mathbf{eff}} - \frac{\gamma\alpha}{\left(1+\alpha^2\right) M_s} \mathbf{M} \times \left(\mathbf{M} \times \mathbf{H}_{\mathbf{eff}} \right) \tag{2.8}$$

This is the equation that we are going to discuss and attempt to solve in this thesis for all the micromagnetic problems.

2.1.3 Solution of the dynamic equation

As discussed in the previous two sections, using either FEM or FD method to construct either the energy variational equation or the LLG equation for a disrectized magnetic domain will lead to a series of ODEs. The LLG equation can be expressed in a general form of

$$\frac{d\mathbf{y}}{dt} = f\left(t, \mathbf{y}\right) \tag{2.9}$$

This system of ODEs can be solved by various time integration methods, including explicit and implicit methods [37, Tenenbaum, 1985 #304].

In order to supply the right-hand-side (RHS) as shown in Eq. (2.9) to the time integrators, the simulator will have to evaluate the $\mathbf{H}_{\text{eff}}$ at every time step, which means that Eq. (2.4) needs to be calculated for multiple times (often tens of thousands and hundreds of thousands times). Therefore, the ability to rapidly compute the effective field is of a primary importance.

2.1.4 Evaluation of the magnetostatic field component

Among the four components of the effective field, the magnetostatic component differentiates itself from other field components as it is the only long range field. So to obtain the magnetostatic field in the entire computational domain, one has either to solve the Poisson equation or use the superposition principle to calculate the interactions between all sources and observers via the Green's function method. Both approaches have high computational complexity. The actual simulations confirm that

even for relative small problems with sizes up to several hundreds, the time used to obtain the magnetostatic field may be dominant as compared to other effective field components.

In this thesis, we are focusing on accelerating the Green's function method as will be called in the following texts. The magnetostatic field can be expressed as

$$H_{demag} = \nabla \iiint_V \frac{\nabla' \bullet \mathbf{M}}{\left|\mathbf{r} - \mathbf{r}'\right|} dV' - \nabla \oiint_S \frac{\mathbf{M} \bullet \mathbf{n}'}{\left|\mathbf{r} - \mathbf{r}'\right|} dS', \tag{2.10}$$

where the first and second terms corresponds to the field generated by the equivalent volumetric and surface magnetic charge distributions, respectively. Both of the above integration terms can be transformed into a discrete convolution followed by the gradient operator

$$\begin{aligned} \phi_j &= \sum_{\substack{i=1 \\ i \neq j}}^{N} \frac{1}{\left| \mathbf{r}_i - \mathbf{r}_j \right|} q\left(\mathbf{r}_i\right), \quad j = 1,...,N \\ H_{demag} &= \nabla \phi \end{aligned}, \tag{2.11}$$

where $q\left(\mathbf{r}_i\right)$ is discretized volumetric or surface magnetic charge and ϕ is the static magnetic scalar potential.

The evaluation of the magnetic scalar potential is a computationally complex task, which is of primary interest of Chapter 4.

2.2 Numerical solutions of general electromagnetic problems

Electromagnetics is a branch of science that deals with physical phenomena related to electromagnetic fields and interactions. The contemporary electromagnetics is built on the foundation laid by James Clerk Maxwell between 1861 and 1862 through the Maxwell's equations [114]:

$$\begin{cases} \nabla \times \mathbf{E} = -jw\mathbf{B} \\ \nabla \cdot \mathbf{B} = 0 \\ \nabla \times \mathbf{H} = jw\mathbf{D} + \mathbf{J} \\ \nabla \cdot \mathbf{D} = \rho \end{cases}, \tag{2.12}$$

Here the Maxwell's equation is shown in for time-harmonic fields, and **E, D, B, H, J** are electric field, electric displacement, magnetic flux density, magnetic field and electrical current density, respectively.

2.2.1 Helmholtz wave equations and its Green's function

From those four equations, electromagnetic wave equation can be deduced and the wave equation in the vacuum can be expressed as second-order partial differential equations:

$$\begin{aligned} \nabla^2\phi + k^2\phi &= -\frac{\rho}{\varepsilon_0} \\ \nabla^2\mathbf{A} + k^2\mathbf{A} &= -\mu_0\mathbf{J} \end{aligned} \tag{2.13}$$

where $\mathbf{A}$ and ϕ are the magnetic vector potential and the electric scalar potential defined as

$$\mathbf{B} = \nabla \times \mathbf{A}, \quad \mathbf{E} = -\nabla\phi - jw\mathbf{A}\,, \tag{2.14}$$

respectively and under Lorentz Gauge Condition

$$\nabla \bullet \mathbf{A} + \frac{jw}{c^2}\phi = 0 \tag{2.15}$$

To solve these wave equations, a number of integral equation formulations can be used. In this paper, we discuss the frequency domain volumetric/surface integral equation only and for time domain IEs, please refer to Ref. [118, 141] for more details.

Here a frequency surface integral equation formulation is shown as an example. Consider a surface problem comprising a structure made of a perfect electric conductor residing in free space. An electric field IE can be written for an unknown surface current $\mathbf{J}_s$ distributed on the surface S of the structure of the linear size D in the following mixed potential form [135]

$$\begin{aligned} &\mathbf{n} \times \mathbf{E}^i(\mathbf{r}) = \mathbf{n} \times (j\omega\mathbf{A}(\mathbf{r}) + \nabla\phi(\mathbf{r})); \ \mathbf{r} \in S \\ &\phi(\mathbf{r}) = \frac{1}{4\pi\varepsilon}\iint\limits_S \frac{\rho_s(\mathbf{r}')e^{-jk|\mathbf{r}-\mathbf{r}'|}}{|\ \mathbf{r}-\mathbf{r}'\ |}ds';\ \rho_s(\mathbf{r}') = -\frac{\nabla' \cdot \mathbf{J}_s(\mathbf{r}')}{j\omega} \\ &\mathbf{A}(\mathbf{r}) = \frac{\mu}{4\pi}\iint\limits_S \frac{\mathbf{J}_s(\mathbf{r}')e^{-jk|\mathbf{r}-\mathbf{r}'|}}{|\ \mathbf{r}-\mathbf{r}'\ |}ds' \end{aligned} \tag{2.16}$$

Here, $\mathbf{E}^i$ is the incident field, is the magnetic vector potential generated by the surface current $\mathbf{J}_s$, and ϕ is the scalar potential generated by the surface charge ρ_s related to the surface current via the divergence. In addition, k is the wavenumber corresponding to the wavelength $\lambda = 2\pi/k$ and frequency $f = kc/2\pi$ with velocity c.

2.2.2 Solution of the Helmholtz equation

Careful readers can easily see that Eq. (2.16) is very similar to the Eq. (2.10) except for different unknowns to be solved. To solve Eq. (2.16), the Method of Moments (MoM) is usually used to discretized the surface, which is meshed into surface elements, typically triangles [68]. The unknown current is expanded via $\mathbf{J_s}(\mathbf{r}) \simeq \sum_{n=1}^{N_s} j_n \mathbf{f}_n(\mathbf{r})$, where j_n are unknown coefficients, $\mathbf{f}_n(\mathbf{r})$ are basis functions, and N_s is the number of the degrees of freedom in the expansion. Often Rao-Wilton-Glisson functions are chosen as the basis functions due to its versatility in handing various geometries [140]. The current expansion is substituted in the IE and subsequently tested (i.e. integrated) with testing functions (which may be chosen the same as the basis functions). Integrating by parts the scalar potential component the IE is transformed in a set of algebraic equations

$$\sum_{n=1}^{N_s} Z_{mn} j_n = V_m \,, \tag{2.17}$$

where Z_{mn} are elements of the impedance matrix and V_m is the tested (known) incident field

$$\begin{aligned} Z_{\mathrm{mn}} &= k \iint\limits_{Sm} d\mathbf{r} \mathbf{f}_{\mathrm{m}}(\mathbf{r}) \bullet \iint\limits_{S_n} \frac{\mathbf{f}_{\mathrm{n}}(\mathbf{r}') e^{-jk|\mathbf{r}-\mathbf{r}'|}}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}' - \frac{1}{k} \iint\limits_{Sm} d\mathbf{r} \nabla \bullet \mathbf{f}_{\mathrm{m}}(\mathbf{r}) \iint\limits_{S_n} \frac{\nabla' \bullet \mathbf{f}_{\mathrm{n}}(\mathbf{r}') e^{-jk|\mathbf{r}-\mathbf{r}'|}}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}' \\ V_m &= j \frac{4\pi}{\eta} \iint\limits_{Sm} \mathbf{f}_{\mathrm{m}}(\mathbf{r}) \bullet \mathbf{E}^{inc} d\mathbf{r} \end{aligned} \tag{2.18}$$

For $m \neq n$ the integrals in Eq. (2.18) are computed using quadrature rules of a certain order while for $m = n$ the singular behavior of the integral kernel is taken into account via an analytical integration. The ways to solve this system of equations by iterative methods will be described in the Section 2.4, but usually the solver will need to evaluate the following equation:

$$u(\mathbf{r}_m) = \sum_{n=1;n\neq m}^{N} \frac{e^{-jk\left|\mathbf{r}_m - \mathbf{r}_n\right|}}{\left|\mathbf{r}_m - \mathbf{r}_n\right|} Q_n\,,\ m = 1,2,...,N \tag{2.19}$$

Here, the potential $u(\mathbf{r}_m)$ at the observation locations $\mathbf{r}_m$ is evaluated by a discrete convolution of the Green's function $G(\mathbf{r}_m, \mathbf{r}_n) = e^{-jk\left|\mathbf{r}_m - \mathbf{r}_n\right|} \Big/ \left|\mathbf{r}_m - \mathbf{r}_n\right|$ and sources Q_n co-located with the observers.

The advantage of the IE method is that it generally does not require the empty spaces in a computational domain to be discretized thus are very effective for complex geometries with empty spaces. It is also good for 3D surface problems such as the electromagnetic wave scattering from conductors.

2.3 Numerical solutions of electromagnetic problems with periodic boundary conditions

Periodic programs are often met in electromagnetics and micromagnetics, e.g. antenna arrays, meta-materials, arrays of magnetic elements, etc. [135]. Using the integral equation methods, the only difference between solving a free-space problem and

periodic problem is the form of their respective Green's functions. To account the infinite cells in the computational domain, the free space Green's function need to be replaced by a periodic Green's function (PGFs) which is usually much more complex than the one shown in the Eq. (2.19) [13, 83, 164]. Therefore, to accelerate the solution of periodic problems, a solver must overcome two major obstacles: 1) Evaluation of PGFs and 2) Convolution as in Eq. (2.19).

2.4 Integral equation solvers

From the description in the previous sections, we could see that in computational micromagnetics and electromagnetics, one of the most critical issues for accelerating the solution of problems are the rapid evaluation of the fields. The fields can either be static fields such as the magnetostatic field defined by the Poisson's equation or the electromagnetic field defined by the Helmholtz equation. Using the Green's function method to solve this two equations will result in integral equations in the form of Eq. (2.17) and Eq. (2.18). Combining the equations for the whole computational domain, we could obtain a matrix equation expressed as

$$\mathbf{U} = \mathbf{GQ} \tag{2.20}$$

As mentioned earlier, in the micromagnetic simulations as described in the Section 2.1, we need to evaluate the magnetostatic field from given known distribution of magnetic charges. This leads to "forward" evaluation of the matrix equation Eq.

(2.20) with $O\left(N^2\right)$ operations. (For the definition of the big-O notations used throughout the thesis, please refer to the Appendix B) In other applications, such as those commonly seen in the electromagnetic simulations, we need to solve for the unknown source distribution from a known field distribution. This can be accomplished via direct inversion methods, which have the cost of $O\left(N^3\right)$ if implemented naïvely. Another class of methods named *iterative methods* solves the above equation without inverting the matrix $\mathbf{G}$ and replaces this process with multiple passes of "forward" evaluation. A general iterative method would have the following steps:

1. Guess the initial solution of $\mathbf{Q}$ as $\mathbf{Q}_0$

2. Evaluate the matrix-vector multiplication and obtain $\mathbf{U}_0 = \mathbf{G}\mathbf{Q}_0$

3. Calculate some measure of the residual $\mathbf{R_0} = \mathbf{U} - \mathbf{U_o}$ whether it is smaller than a prescribed error ε_0. If yes, return $\mathbf{Q}_0$ as the solution. If no, calculate $\mathbf{Q}_1$ from $\mathbf{Q}_0$ using a formula or with the information from $\mathbf{R}_0$.

4. Repeat the stage 2 and 3 until the residual meets the exit condition.

There are many iterative methods such as Jacobi method, Gauss-Seidel method, conjugate gradient (CG) method or the generalized minimal residual (GMRES) method, etc. [60, 147, 148]. The cost of these methods if implemented directly is $O\left(N_{it}N^2\right)$, where N_{it} is the number of iterations. Therefore, for both the micromagnetic simulations and the electromagnetic simulations, the most critical issue of a high performance

solver is the rapid evaluation of the matrix-vector multiplication as shown in the Eq.(2.20), which can be seen as a convolution between two functions. This thesis is concerned with iterative methods.

2.5 Fast methods for integral equations

Reducing the computational complexity of the Eq. (2.11) or the Eq. (2.19) has been pursued by many researchers in the various disciplines, e.g. Ref. [36] This thesis presents two algorithms that represent two categories of fast algorithms.

The first category of methods realizes that the Eq. (2.11) and Eq. (2.19) can be seen as a convolution between two functions, which can be accomplished through multiplication in the frequency domain after Fast Fourier Transforms [39]. The resulting computational complexity is $O\left(N \log N\right)$, which is a significant reduction over the original $O\left(N^2\right)$ operations. However, the utilization of FFT requires the sources to present uniformly and in a periodic pattern. Thus, to meet this requirement, the Conjugate-Gradient FFT (CG-FFT) method [26] assumes a uniform source discretization scheme, resulting in regularly distributed discrete sources. On the contrary, the precorrected-FFT (pFFT) method [136] and Adaptive Integral Method (AIM) [11] do not impose any restrictions on the discretization by introducing uniform auxiliary grids for transferring the interaction between the actual discretized sources and observers. These

auxiliary grids interact with their primary sources/observers through various interpolation and anterpolation/projection schemes. Both of these methods make the evaluation of the convolution to be done in $O\left(N \log N\right)$ operations for general volumetric problems and $O\left(N^{3/2} \log N\right)$ for general surface problems [36]. In Section 4.3, the author presents a modified AIM method called, Box AIM (B-AIM) that achieves the same asymptotic complexity but with much higher parallel efficiency on GPU hardware platforms.

The second category of method is the hierarchical multi-level method, often called "tree-code" due to the way they divide the computational domain. Algorithms fall in this category include the Barnes-Hut Method [8], Particle-Particle Particle-Mesh (P3M) [48, 161], and Multi-Level Fast Multipole Methods (MLFMMs) [34, 61, 110, 143], etc, with MLFMMs being the most well-known one. The FMMs are originally proposed by Greengard and Rokhlin in their famous paper published in 1987 [61]. For statics In FMMs use the fact that the field far from a group of sources can be represented in terms of a small number of multipoles. For dynamics, the field can be represented in terms of plane waves or related expansions and the translation properties of plane waves can be utilized to achieve a similar complexity reduction as in the static case, albeit with a higher complexity. This property reduces the computational complexity contrary to treating sources one by one as in direct methods. To correctly and efficiently represent the field generated by sources both far and near, the computational

domain is divided into far-field region, where the error of the multipole expansion is below a prescribed value, and near-field region, where the field can only be calculated directly through superposition. The total complexity of FMM-like methods depends on the two competing factors, the near-field operations and the far-field operations which are dictated by the choice of the near-far field criteria. The asymptotic computational complexity of the MLFMM is $O\big(N\big)$ for the static field calculation as in Eq. (2.11) and $O(N \log N)$ for dynamic field calculation as in Eq. (2.19) [36, 143].

In Section 4.2, a spatial multi-level "tree-code" called the Non-uniform Interpolation Method (NGIM) is described. The NGIM is theoretically proposed by A. Boag in [16] and the computational complexity is proven to be the same as the MLFMMs [15, 16, 102]. However, this algorithm has several advantages while dealing with specific types of problems and the author will present detailed results and analysis in that section.

3 Introductions to the Graphics Processing Units (GPUs)

Graphics Processing Units (GPUs) are one of the many processors in modern-day desktop, laptop and mobile PCs and tablets. GPUs were originally designed for handling display-, image- video- and graphics-related applications. However, the impacts of GPUs have stepped outside traditional definitions and a new term, General Purpose Graphics Processing Units (GPGPUs), is becoming increasingly popular in a number of computational communities. To help understand why computational scientists and researchers put much effort in designing or modifying their data or compute intensive algorithms to run on GPUs, and to understand what distinctive features GPUs have a brief history of GPUs is presented next.

3.1 A short history of GPUs

Graphic processors appeared long time ago ever since the invention of the computer itself. Early graphic processors were used for drawing texts and pictures on the screen and later, after the three-dimensional (3-D) graphics contents became popular, the modern graphics pipeline emerged. We begin our survey on the history of GPUs from the late 1990s, when the 3D graphics appeared. The earlier history of GPUs can be found in Ref. [12].

3.1.1 3D graphics pipeline

A typical scene in 3D computer graphics consists of many objects that a viewer might be able to see. The geometrical shapes of these objects can be very complicated with thousands of surfaces. These objects are formed through basic elements such as points, line segments, and polygons. Often the surface of a 3D object model is represented by a bundle of triangles, each of which is made of three vertices.

The major tasks of GPUs are to process the information related to these objects and to draw them on the screen in the way that looks natural to the viewer. To that end, the GPUs need to perform various transforms on primitive geometrical data related to the objects and this process is described in the "graphics pipeline".

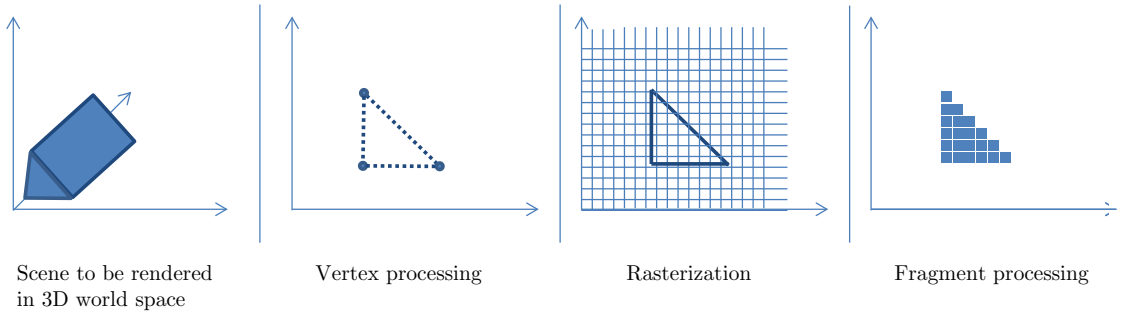


Figure 1 Stages of a common 3D graphics pipeline

(a) Vertex transformation:

After geometrical data of a 3D object is passed to the GPUs, one of the jobs they need to perform is to transform the data into a two-dimensional space that

matches the actual display region. This is done through transforming all vertices of the objects and reconnects them in the new coordinate space. There are many coordinate systems associated with this process. An object has a position and orientation in the 3D world space, a global coordinate system that relates the object with other objects in the same scene. There is also camera space where the z axis is parallel to the viewing direction and x and y axis are aligned with the display boundaries. Before showing it onto the screen, the object also needs to undergo a transformation that takes the perspective into consideration, which make the far objects smaller to the viewer. Finally, the object is mapped onto the screen taking the position of the active window into consideration.

The operations in this stage consist of large amount floating point number operations. Transformations done on different vertices are completely independent, which allows for massive parallelization. Therefore, it is not surprising that modern GPUs have thousands of parallel processors and a very high parallel performance (currently in the tera-FLOPS range).

(b) Rasterization:

Once a graphic object has been transformed into the window space, the GPU must determine which pixels are covered by the object. The process of converting a triangle to a collection of pixel fragments is called rasterization. Each triangle sampled

uniformly at the center of each pixel consists of a group of points. The depth, interpolated color, and textures of these points are called a fragment.

(c) Fragment operations:

Fragments generated by different primitive objects are well possible to overlap with each other and some fragments might even be facing back. The GPUs may eliminate these unnecessary fragments at beginning of this stage and this is called "fragment shading". Each pixel is then subsequently processed to compute a final color value, via simple superposition of interpolated color values from vertices or very complicated formulas that emulates complex environmental lightings. This is called "pixel shading". The need for complex shading processes led to programmable compute units that later made the GPU capable of doing general purpose calculations.

(d) Frame buffer operations:

Finally, the computed shaded pixel fragments are written to a frame buffer that are ready to be displayed onto a windowed application.

3.1.2 Fixed-function GPUs

Before the year 2000, GPUs were mostly fix-function engines. One notable generation of GPUs, the NVIDIA's Geforce 256 and ATI's Radeon 7200, introduced in Oct. 1999 were the first set of devices that took the entire graphic pipeline onto its shoulder, freeing the CPUs for other tasks. In fact, the name "GPU" itself was intro-

duced by NVIDIA after the launch of GeForce 256. Nevertheless, scientists did try to map non-graphics operations to those GPUs. They found many limitations on then-state-of-the-art GPUs such as fixed-precision numbers and very slow frame-buffer readback [146].

Two generations later, the NVIDIA Geforce 3 and ATI Radeon 8500 cards implemented the programmable shaders that significantly changed the way developers thinking about graphics pipelines. The programmable shaders allowed developers to write codes without thinking about setting the pipeline states. This change intrigued more researches to explore the possibility of using GPUs for general-purpose computations. In Ref. [69, 70], PDEs were implemented on this generation of GPUs and the term General Purpose GPU (GPGPU) was first introduced by Mark Harris.

Two generations later, in the NVIDIA Geforce FX 5800 and ATI Radeon 9700 cards, a new shader model was implemented and the GPUs started to support the floating point precisions. The floating point precision calculation is a critical feature for the computer graphics developers to render many physical phenomena such as fire, smoke, fog, clouds to the realistic quality. It opened the door to new possibilities of employing GPUs in general computations. 32-bit floating point format was supported by NVIDIA right at the launch of those GPUs while ATI chose to implement a 24bit scheme as they believed that it was sufficient for most graphic applications. Later, both

vendors started supporting support 32-bit single precision floating points, though they were not entirely IEEE-754 compliant.

Another important feature that appeared in this generation of GPUs is the high level shading languages, replacing the assembly-level languages used for programming previous generation shaders. NVIDIA proposed Cg, Microsoft integrated a High Level Shading Languages (HLSL) into its DirectX API and loyalty-agonistic OpenGL standard introduced GL Shading Languages (GLSL), which made coding GPU shaders much easier than before.

3.1.3 The Emergence of GPGPUs

From the previous description of the graphics pipeline, the reader could see that the two programmable stages, the vertex processing and the fragment processing, perform quite different operations. The vertex processing stage, in which coordinate transformations are involved, is abundant of arithmetic operations. The fragment processing stage, on the other hand, is filled with high-throughput memory transfer due to texture filtering. Typical graphics workloads requires more computational power in the fragment processing stage than in the vertex processing stage and the GPUs listed in the last section usually handling this by providing 2 or 3 time more fragment shaders than vertex shaders. However this static solution is obviously suboptimal for tasks under extreme conditions such as rendering scenes with many large triangles.

In addition to the load balancing problem between vertex and fragment shaders, the demand of rendering increasingly complex scenes made it necessary to have a new shader, called the "geometric shader", appearing in between the vertex and fragment processing stage. Allocating and balancing hardware resources across these three types of shaders became an impossible job and the GPU architects began considering what was a revolutionary "unified shader" that could handle all the tasks involved in these three stages. These "common" processors had powerful floating point operation capabilities as well as high bandwidth for accessing the graphics memory. As a result they were becoming more and more suitable for general- purpose computing tasks.

3.2 The Architecture of GPGPUs

The first generation of GPUs with unified processor architecture were NVIDIA's Geforce 8800GTX (released in November 2006) and ATI's Radeon HD 2900XT (released in May 2007).

3.2.1 NVIDIA G80 architecture

Figure 2 illustrates the architecture of the Geforce 8800 GTX chip, codename G80, the first CUDA-capable GPUs. The device was built from an array of stream processors (SP), which are sometimes called ALUs in other vendors GPUs. Eight of these SPs were bundled together to form streaming multiprocessors (SM). Various

other functional units such as texture memory caches, instruction dispatcher and special function unit (SFU), etc., were attached to SM to assist dealing with the pipeline operations. In Geforce 8800 GTX, there were 128 SPs, grouped into 16 SMs, and in each SM there were 8 ALUs, 2 special function units, 1 instruction dispatcher unit, 16 KB of fast "scratch pad" memory. There were also 64 KB read-only constant memory shared by all SMs.

Figure 2 The Geforce 8800GTX architecture (G80) with unified shaders. The figure on the bottom shows the internal structure of an SM.

Since different shaders might use different function units within an SM, each SM was hardware multithreaded, which meant it was able to manage much more threads than actual SP it contained with almost no scheduling overhead. For G80 architecture, each SM could handle up to 768 concurrent threads and the entire GPU would handle up to 12K threads simultaneously. Later architectures from NVIDIA such

as the GT200, GF104 and current generation GK104 are all similar to G80 with some changes in processor arrangement and cache levels [125, 130].

For performance reasons, the GPU architect invented a concept called "warp" [127] that is made of 32 threads. Threads within each warp execute same instruction at the any given time, which brings down the scheduling burden of the dispatcher unit. However, if the code branches within a single warp, resulting in different instruction paths for different threads, the warp will go through all the instruction paths sequentially and disabling threads not at that path one at a time. This will significantly reduce the execution efficiency and should be avoided when possible.

Another unique feature of G80 architecture is its various types of memories. The latencies of each types of memory are benchmarked in Ref. [6, 172]. The registers are on-the-chip temporary storage that has almost no latency to be accessed, similar to those inside CPU. One level higher, there is "shared memory". These extremely fast memories are designed for the threads within the same "thread block" to access simultaneously. Thread block consist of several warps that may or may not do similar tasks. The threads within the same block can communicate with each other without going back to the off-chip DRAMs and can be synchronized with specific instructions. They are always mapped to the same SP. Communication between threads of different block has to be done via off-chip DRAMs, called the "global memory". Global memories is relatively large in sizes (768 MB for Geforce 8800GTX) and several giga-bytes for more

recent generations (e.g. 6 GB for Tesla M2090). GPUs' global memory has very high bandwidth compared to CPUs' main RAM but it can only be accessed at a rate close to the theoretical limit through the "coalesced memory access". The "coalesced memory access" occurs only when multiple contiguous threads within the same warp attempts to access contiguous memory addresses that are aligned to a certain number. If these conditions are met, a burst of data can be transferred through global memory bus and the non-negligible latency of global memory can be hidden across multiple threads.

3.2.2 AMD Radeon R600 architecture

Though AMD Inc. acquired ATI Technologies in 2006, it kept ATI's product line and branding. Radeon R600 architecture was the first generation GPUs featuring a unified shader architecture from AMD.

The R600 architecture is very similar to G80 and only different in the way that it bundles its stream processors. The Radeon HD 2900XT GPU contains 320 scalar stream processors (SP) arranged into for groups. For each group, the 80 SPs are again divided into 20 units and each 4-SP unit is attached a special function unit to form a 5-way superscalar shader. Each of this 5-way shader has its own branch control unit and general-purpose registers. This architecture has changed in the latest generation of AMD GPUs that will be mentioned in Section 3.4, so the hardware architecture is described here only for completeness. R600 architecture also used the same types of memories that match to those in the G80 architecture.

3.3 GPU programming model and its impact in scientific computing

A mature and easy-to-use development environment usually hides the hardware details from developers as the hardware implementations are different from different vendors. A well-defined set of APIs sits between the application and driver layer is necessary for software developers to unleash the power of GPUs. Several famous APIs are listed here.

3.3.1 Graphics APIs

The two most widely used graphics APIs are DirectX (Direct3D, specifically) and OpenGL. These two APIs were proposed in 1995 and 1992, respectively and they have evolved significantly. The OpenGL specification is moderated by an industrial consortium called Khronos Group and is implemented as a multi-platform loyalty-free open standard. The DirectX is a product of Microsoft for exclusive use on Windows and its XBOX gaming console. There were other APIs being used at some point in the history, such as the Glide3D proprietary API developed by 3dfx, but it was later abandoned.

Both of these APIs serve the same purpose - abstracting the hardware details and letting the programmer to focus on the actual operations in the graphics pipeline. Graphics APIs are designed specifically for computer graphics applications and some-

times influence the hardware design as well. New features and functionalities sometimes appears in definition of graphics APIs before being actually realized in the hardware. This is especially true for DirectX.

3.3.2 General purpose programming APIs for GPGPUs

As have described in the previous chapters, even before the GPGPU had appeared, researchers and scientists have tried to leverage the computing power provided by GPUs to accelerate the scientific simulation codes. However, GPUs were not a popular choice among the majority of researchers until much more user-friendly high-level languages and APIs appeared.

(a) CUDA C

The CUDA C programming model was the first and most widely used computing model for GPGPUs, proposed by NVIDIA. It is a multi-platform API, supporting Windows, Linux and MacOS, but runs only on NVIDIA GPUs after Geforce GTX 8800. CUDA C is designed for developers to directly control the stream processors and multiple levels of on- and off-chip memory as well as the communication between the CPU (the host) and GPU (the device).

On the host side, CUDA deals with necessary initialization, data passing between host as well as device and kernel launching. On the device side, it is an extended version of traditional C programming language with modifications to address the

unique needs of massive parallelization and multiple levels of memories. Each function written to be running on the device is called a "kernel" and kernels are usually designed to be executed thousands or even millions of times simultaneously. Each of the launched instances is called a "thread" and they are usually identified via a distinct "thread ID". Several threads are bundled into warps and several warps form a "thread block". A "thread block" is mapped to a SM. Each block has its own shared memory and threads can communicate with each other using this "shared memory". It is usually a good practice to use the shared memory as it is on-chip and is at least one order of magnitude faster than the global memory. Correct utilization of the shared memory is one of the most important aspects to make many GPU codes efficient. There is no way to determine the sequence of executions among threads from different thread blocks and there is no synchronization mechanism for them except at the end of a kernel launch. There is a size limit on the number of threads that a thread block can contain, and this number is 512 for older CUDA compute architectures like G80 and 1024 or 2048 for newer ones.

Since the launch of the first version of CUDA in 2007, NVIDIA has continuously added many new features to this programming model to accommodate the change of hardware architecture and to increase its usability. The latest beta release of CUDA is version 5, which introduces many new features that will be discussed in the Section 3.4.

All the GPU algorithms that will be presented in Chapter 4 and all the GPU-accelerated solvers presented in Chapter 5 are written for NVIDIA GPUs using CUDA. During the writing of this thesis, the current stable version of CUDA is 4.2 but many timing results shown in Chapter 4 are still obtained using older version as far back as 3.0.

(b) OpenCL

OpenCL is an API proposed by Apple Inc. in 2008 and maintained by Khronos Group. It aims at creating portable, vendor, and device independent programs that are capable of being accelerated on many different hardware platforms and it is the first API that explicitly aims at utilizing all computing resources in a computer system, including CPUs and GPUs [82, 121]. However, as the GPGPUs remain as the most widely used many-core processors, the OpenCL programming model turns out to be very similar to the NVIDIA's CUDA model. It also contains two different sections, the host side APIs and the device side kernel language based on C. Despite small differences in names, the OpenCL APIs can almost be one-on-one matched with corresponding CUDA Driver APIs. The OpenCL extensions used in the kernel code can also be understood by many experienced CUDA developer with ease. Interested readers can refer to Ref. [2, 82, 121] for further details of OpenCL.

(c) DirectCompute

DirectCompute is a component of DirectX 11, published by Microsoft and supported by both NVIDIA and AMD GPUs. DirectCompute differs itself from the CUDA and OpenCL programming environment as it runs only on Windows operating systems, specifically speaking, Windows 7 and above. Naturally, the host side APIs of DirectCompute follows the Direct3D fashion and the device side kernel code is similar to HLSL, both of which are maintained by Microsoft.

(d) Microsoft C++ AMP

Microsoft C++ Accelerated Massive Parallelism (AMP) is a library developed by Microsoft and bundled as a run-time library with its Visual Studio 2012 compiler suite [117]. C++ AMP accelerates the execution of C++ code by automatically identify pieces of code that can be executed on GPUs with high efficiency. C++ AMP exploits the data parallelism opportunities that are often met in loops, multidimensional arrays and memory transfer etc. It is built up on DirectCompute and if no compatible GPGPUs are presented, it will fall back onto CPUs. The syntax of C++ AMP is similar to the newly proposed and standardized C++11, and should be relatively easy for C++ programmers. However, same as the DirectCompute, the lack of multi-platform support limits its applicability in many scientific computing applications.

(e) OpenACC

OpenACC is an open standard proposed by NVIDIA that intends to make the GPGPUs more accessible for scientists who focuses less on the optimization process. OpenACC adopts a similar approach as the OpenMP and is designed as set of directives and pragmas for the programmers to hint the compiler to do certain parallelization and optimizations on certain sections of codes. OpenACC is still in its very early stage of development and very limited information is available [131].

3.4 Future architectures and potential impact to scientific computing

Ever since their emergence in 2006, GPGPUs have gained a tremendous momentum in compute intensive applications such as computational photography and physics simulations. This wide adoption of GPGPUs has encouraged the vendors of GPUs to improve their architecture further for compute tasks. Other many-core architectures, too, have begun entering the high performance computing area. In this section, we briefly introduce these new architectures and interested readers should follow to documentations from each vendor to see their latest development.

3.4.1 NVIDIA's Kepler GK110 architecture [130]

The Kepler GK110 GPGPU architecture is the latest generation CUDA capable compute architecture. It pushes the GPUs a step further towards the HPC and empha-

sizes not only the raw computing power but also energy efficiency. The key features of GK110 GPGPUs include:

(a) Significant boost in double precision computing capability

One of the most critical areas that GPU has lacked so far is the double precision computing capability, partly due to the fact that graphics applications do not require such high precision. In fact, many operations, especially in the fragment shading stage, require only 16-bit floating point accuracy to be visually satisfying. However, for many scientific computations, such as iterative solvers and other sensitive systems, a little truncation error of floating point numbers may lead to high numerical inaccuracies. Therefore, increasing double precision performance should increase the range of GPGPU use in high-performance scientific computing.

(b) Increased ability handling complex flow

NVIDIA implemented two new features, thread launching from within GPU kernels and handling multiple kernels simultaneously. This would make the programming of GPUs similar to multithreaded applications on CPUs.

(c) Changed SM composition.

The new architecture also changed the way SPs are organized into an SM and a rebalance of compute resources for better accommodating the HPC computing needs. This includes but not limited to, significant increase in the register resources, quad

warp scheduler for each SM and new shuffle instructions useful for non-sequential memory operations, etc.

3.4.2 Intel's Many Integrated Cores (MIC) [79, 150]

Intel's approach towards heterogeneous computing architecture is different from graphics vendors like NVIDIA or AMD. In 2009, Intel demonstrated a GPU that is completely different from their widely adopted Intel GMA graphics chip. This GPU was at that time known as *"Larrabee"* but its development was terminated in 2010. Later, Intel proposed a new architecture, called "Many Integrated Cores" that inherited many features of *Larrabee* and the first device named "Knights Corner" with 50 cores integrated into a single chip, was released in 2012. There are very few benchmarks available for this new device at the moment and the technical design is not exposed to public either. However, Intel claims that MIC is fully compatible with application compiled for x86 CPUs but it does include special graphic function units, such as the texture sampling unit.

3.4.3 Reconfigurable Computing (RC) architectures [18, 38]

Reconfigurable computing has long been a hot topic in scientific computing. It takes the advantage of reconfigurable hardware such as field programmable gate arrays (FPGAs) as accelerators to handle highly parallel computing tasks. FPGAs can change their hardware design during the runtime and adapts themselves to different algorithms

at the hardware level. More importantly, RC takes advantages of data parallelism in the algorithm too. Comparing with multi-core CPUs and GPUs, FPGAs utilizes everything on the chipset to do the specific operations and waste nothing in non-task-related functions like interrupt handling or even instruction branch prediction, etc. Another advantage of RC is its low FLOPS-Watt ratio, which might be a critical limiting factor for the current supercomputers to reach the Exa-FLOP scale. Benefitted from its low clock frequency, the power efficiency of FPGA is usually higher than that of the traditional CPUs and GPUs. However, since FPGAs are expansive and relatively hard to program, the growth rate of their usage in HPC community is much lower comparing with GPGPUs.

3.4.4 Merge of traditional CPU and GPUs

From the above discussion one can see that there is a tendency that GPUs become increasingly general-purpose and CPUs become increasingly parallel. This tendency has been well identified by many chip vendors. Companies like AMD even started developing unified computing unit called Accelerated Processing Units (APUs) that aims at ultimately unified the CPUs and GPUs. This parallelization process actually fits the scientific computing application very well as many physical phenomena are intrinsically parallel. Many researchers share the same conclusion as the ultimate physical limit for any forms of processors should be the same [32, 78, 176].

4 Fast algorithms for integral equation solvers on GPUs

4.1 Current status and literature review

In Section 2.5, we have briefly discussed the ways that integral equation solvers can be accelerated. However, even IE solvers using those fast methods easily take too long for desktop workstations. GPUs are very promising to further accelerate those solvers.

4.1.1 MoM on GPUs

Accelerating the direct implementation of iterative MoMs approaches is straightforward. There is actually a tech demo in the NVIDIA GPU Computing SDK that shows simple gravitational simulations, within which the gravity *n-body* problem is calculated using Eq. (2.11) [129]. In this demo, the algorithm to obtain the aggregated gravitational forces on each particle is the direct superposition as shown in Eq. (2.11). Though simple in implementation, it is very effective for accelerating simulations that require solving this type of n-body problem. When the number of degrees of freedom is relatively small the small overhead of direct method makes it the fastest option. In Section 4.2.3, we could see that GPU direct method outperforms many CPU fast methods on problems even up to several millions.

In the field of computational electromagnetics specifically, accelerating the method-of-moments (MoMs) without fast methods has also been the topic of many

research literatures. Most of the implementations of MoMs on CPUs lead to a system of linear equations represented in the form of matrix equations, in order to take advantage of existing well-optimized high-performance linear algebra libraries, such as BLAS.

Speaking specifically about the computational electromagnetics, Ref. [44, 89, 90, 92, 160], the impedance matrices are filled in the preprocessing stage and solved by LU decomposition. In Ref. [45, 132], the synthesized matrix equations are solved by the conjugate gradient (CG) iterative method. The speed-ups achieved following these approaches are in the range of 10-50 depending on the optimization details and the actual hardware platforms the comparisons are made on. Nevertheless, due to significant memory used by the uncompressed impedance matrix, the largest problem sizes these solvers can handle are limited to tens of thousands, even on a dedicated compute-only NVIDIA TESLA card with several giga-bytes of memory. To the best of the author's knowledge, there is no literature on *"on-the-fly"* implementation of MoM on GPUs. *"On-the-fly"* means filling the impedance matrices right before they are used and discarding them after. However, either the LU decomposition or the iterative solvers involves many interactions between elements so most of them require the whole matrices being existed on GPUs' global memory at a certain point. This would make the previous *"on-the-fly"* approach meaningless from the perspective of memory saving.

4.1.2 FFT-based fast algorithms on GPUs

As mentioned in Section 2.5, the quadratic computational complexity of the direct MoMs limits its applicability to realistic simulations, so porting MoM with fast methods to GPUs draws much attention from the researchers in the computational electromagnetic field. Due to relatively simple data structures and high parallelizability, the FFT-based fast methods are popular among solvers designed for multi-node computer clusters and are naturally being considered for porting to GPUs. In any of the FFT-accelerated algorithms, one of the essential building blocks is the FFT itself. Not being a perfect algorithm for GPUs but widely used in many areas, the FFT has been ported to GPUs long before the GPGPU era. The Ref. [119] might be among many earliest attempts to use OpenGL APIs for the general computing needs. Since CUDA was introduced, various implementations were published for several generations of GPUs. The CUFFT library that comes along with the CUDA release might be the most widely used one due to its similarity with the popular FFTW libraries on CPUs and adequate, constantly improving performance [128]. Other researchers have claim that they achieved higher performance [52, 59, 63, 109, 124, 144]. Meanwhile, Ref. [42, 47, 123] focus on other aspects, such as the memory transfer between CPU and GPU, mixed-radix performance improvement, and automatic performance tuning. There are also efforts on accelerating FFTs across multiple GPUs, e. g. in [33], but to the best of the author's knowledge, the performance achieved are not satisfactory for many applications due to significantly data exchange latencies on multi-GPU systems.

Being built based on aforementioned FFT libraries, researchers in computational electromagnetics tried to implement AIM and pFFT algorithms on GPU. Only a few attempts are recorded. In Ref. [50, 51, 133], FFT-based fast algorithms are implemented and tested up to 80k unknowns. These implementations try to express the fast methods in the form of matrix-vector multiplication explicitly, just like for direct MoMs. Thus all of them can go up to only few tens of thousands of degrees of freedom.

4.1.3 Hierarchical fast methods

Hierarchical fast methods usually divide the computational domain into multiple levels of boxes and treat the far- and near-field separately. In Ref. [67, 155], multilevel direct Coulomb summation is shown to achieve over 100 GFLOPS on NVIDIA Geforce 8800GTX card and approximately 50x speed-ups comparing with a core of a Intel Core2 QX6700 CPU. More complex tree codes like the Barnes-Hut and Particle-to-particle-particle-to-mesh (P3M) methods [9, 24, 66, 80, 154] are implemented on desktop CPU- and GPU- clusters, achieving astounding tens to hundreds of TeraFLOPS of peak performance on several hundreds of nodes. FMMs are also a quite popular choice for evaluating long range forces such as the electrostatics. Ref. [62] is an early attempt to porting static FMMs onto GPUs though most of the speed-ups are obtained by changing the ratio of near- and far-field computational burdens as the near-field is easier to be accelerated on GPUs. Later attempts obtained much better performance due to improvement in both the algorithm designs and hardware architec-

tures [29, 43, 87, 108, 178, 179]. The largest problem sizes these FMMs implementations can handle on a single GPU card is in the order of 10^7 [178] and on multiple GPU-clusters has exceeded one billion level [77, 179].

Although researches on FMMs for static fields are quite fruitful recently, using FMMs to calculate Helmholtz-type of dynamic fields is quite hard as the series representation and translation of dynamic fields are more complex. In Ref [41] , the researchers did a satisfactory job for accelerating FMMs in the low frequency and achieved good speed-ups.

4.1.4 Solution of linear systems

With or without those fast methods, the MoMs usually produces a system of linear equations that need to be solved either directly by matrix inversion or by various iterative methods. Direct matrix solvers on GPUs have been shown in Ref. [54, 91, 168] and researches on iterative solvers can be found in Ref. [7, 17, 28, 71, 165]. The performance obtained on widely used iterative methods like the GMRES are around 5-10x, comparing with sequential codes on CPUs.

4.2 Non-uniform Grid Interpolation Method (NGIM)

In this section, we describe a highly efficient GPU implementation of a modification of the non-uniform grid interpolation method (NGIM) [16, 98, 102], for fast evaluation of the potential $u(\mathbf{r}_m)$ in Eq. (2.11) or Eq. (2.19) . Acceleration of NGIM is

built on the fact that the field potential far from a source is a function with a known asymptotic behavior. The slowing varying fields far from the sources can be sampled and represented pretty accurately by a few sample points. Then the field values on many other points can easily be interpolated from these samples points instead of doing costly Green's function evaluation. The algorithm is implemented using a hierarchical domain decomposition method, similar to Multi-level FMMs (MLFMMs) [36, 152]. The domain is subdivided into several levels comprising subdomains of different sizes. Near- and far-field spaces are identified and the interpolation procedures are implemented for the sufficiently separated subdomains. This algorithm achieves the computational cost of $O(N)$ in the low-frequency regime, $O(N \log N)$ in the high-frequency regime, and somewhere in between in the mixed-frequency regime.

Similar to the MLFMMs, NGIM can also handle non-uniform geometries and has the same asymptotic cost for both volumetric and surface problems. The NGIM differs from the MLFMMs in that it relies on direct spatial interpolations, which are operations the GPUs destined to do. Moreover, the same NGIM can be applied to static $(k = 0)$ and dynamic $(k \neq 0)$ problems, as well as to problems with other kernels, without major changes in its structure and mathematical operations, as opposed to MLFMMs. Moreover, NGIM does not require any special function evaluations, which increases it execution efficiency on GPU systems, as will be shown in the Section 4.2.3.

4.2.1 Algorithm description

In this section we present the description of the NGIM and its implementation scheme. We discuss how the algorithm is coupled with general electromagnetic IE solvers.

The NGIM divides the computational domain into a hierarchy of boxes containing sources and observers. At any level, each box is treated as a "parent" box, which is divided into eight "child" boxes at lower level. This process goes on recursively and stops until boxes at finest level contain less than a prescribed number of sources. These boxes form an oct-tree. Then, for a certain box, near-field and far-field boxes are identified by distances larger or smaller than a predefined value (e.g., twice of the box diameter). The fields contributed by sources in the near-field boxes will later be evaluated via direct superposition. The fields generated by sources in far field boxes are aggregated to their respective box center and have them interpolated on observers through several stages of operations, including complex upward and downward traversing of the oct-tree. This procedure, illustrated in Figure 4 - Figure 7, is similar to other multi-level algorithms such as MLFMMs.

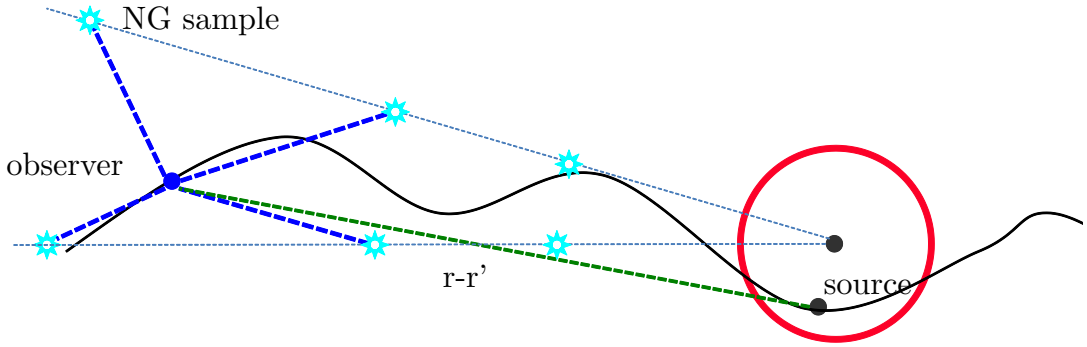


Figure 3 Calculating the field from far away sources by interpolation through NG samples. The direct evaluation of the interaction between the source and the observer is shown as the green dashed line and the indirect evaluation through NGIM is done through grid samples (cyan points) and interpolations (blue dashed lines).

The field outside a group of sources is amplitude- and phase-compensated with respect to the common distance from the group center. The resulting slowly varying "compensated field" can then be sampled at a sparse non-uniform grid (NG) and calculated at all desirable observation locations via inexpensive local (e.g. Lagrange) interpolation. The density and specific position of NG samples are determined mathematically [14] in the preprocessing stage and remain unchanged during the entire simulation. For low-frequency evaluations, a total number of $O\left(N\right)$ NG samples are required to sample the field across all levels of the boxes. But, in high-frequency regime, since its oscillation does not diminish to zero at infinity, this number is $O\left(N\log N\right)$. Generally speaking, in both regimes, the higher the accuracy requirement, the denser the NG grid.

The frequency of fields also affects another aspect of the algorithm. For low frequency problems, the field transition from NG samples to the final observation points are not done directly but via another set of intermediate Cartesian Grid (CG). The CG saves the computational cost because in low frequency applications the density of observers is determined by the geometry of objects thus might be very high. In this case, a small number of sampling points is sufficient for calculating fields on a large number of observers. In high-frequency regime, the density of discretized sources is determined by the wavenumber of the field, so the number of CG samples would be the same as the observers and would not save operations. Therefore, CG is not used in the high-frequency calculations. In the mixed-frequency regime, CG is built for lower levels with boxes much smaller than wavelengths (low-frequency levels) and is omitted for higher levels otherwise (high-frequency levels).

After the boxes and grids are built, fields at actual observers are calculated from sources via a sequence of interpolations. In the low-frequency regime, for example, the field at observers is interpolated via CG samples, which is in turn obtain from NG samples of certain boxes and CG samples of their parent boxes. The fields at NG samples are obtained, via interpolation, from their child boxes, except on the lowest level, on which they are calculated directly from the sources. This process can be described as a stage-by-stage procedure shown below: [93, 98]

Stage 0 (near-field evaluation):

All near-field interactions between the sources in the near-field boxes at the finest level are computed via direct superposition. This step is completely independent of the rest of the algorithm and can be separately implemented and executed in parallel or in sequence of other stages, depending on the available hardware.

Stage 1 (finest level NG field calculation):

The field values on NG samples are computed directly from the sources. This operation is only valid for the boxes at the finest level.

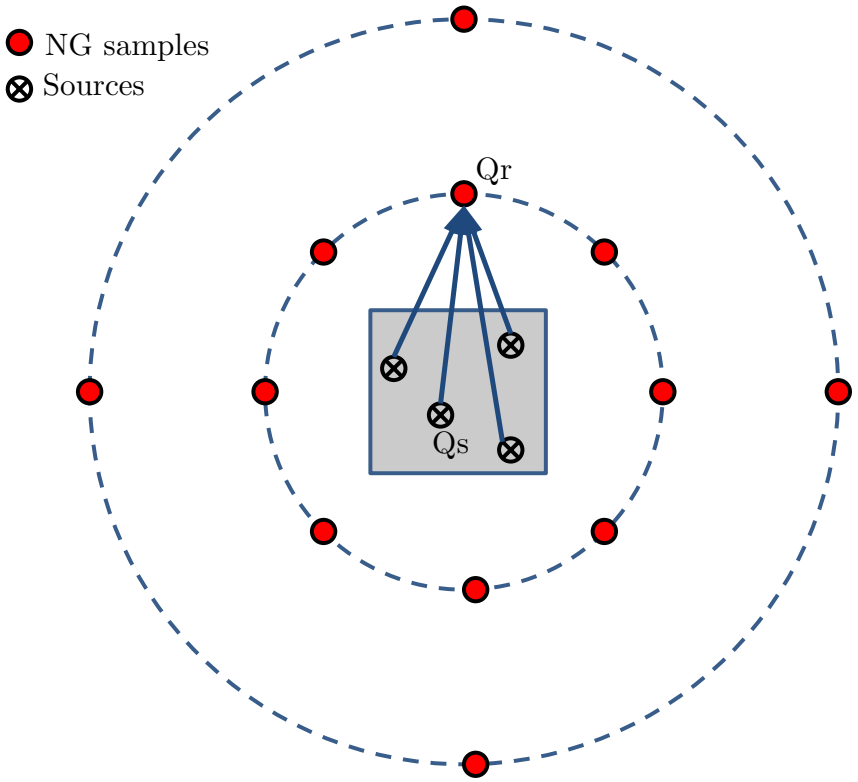


Figure 4 The illustration of the Stage 1: field calculation on NG samples on the finest level. The fields on the NG samples (red dots) from sources (cross dots) are calculated directly.

Stage 2 (aggregation of NGs/upward pass):

The field values on NG samples of the boxes at coarser levels are computed by accumulating fields at NG samples of their eight child boxes. Such aggregation involves local interpolations and common distance compensation in the amplitude and phase between the corresponding NG samples.

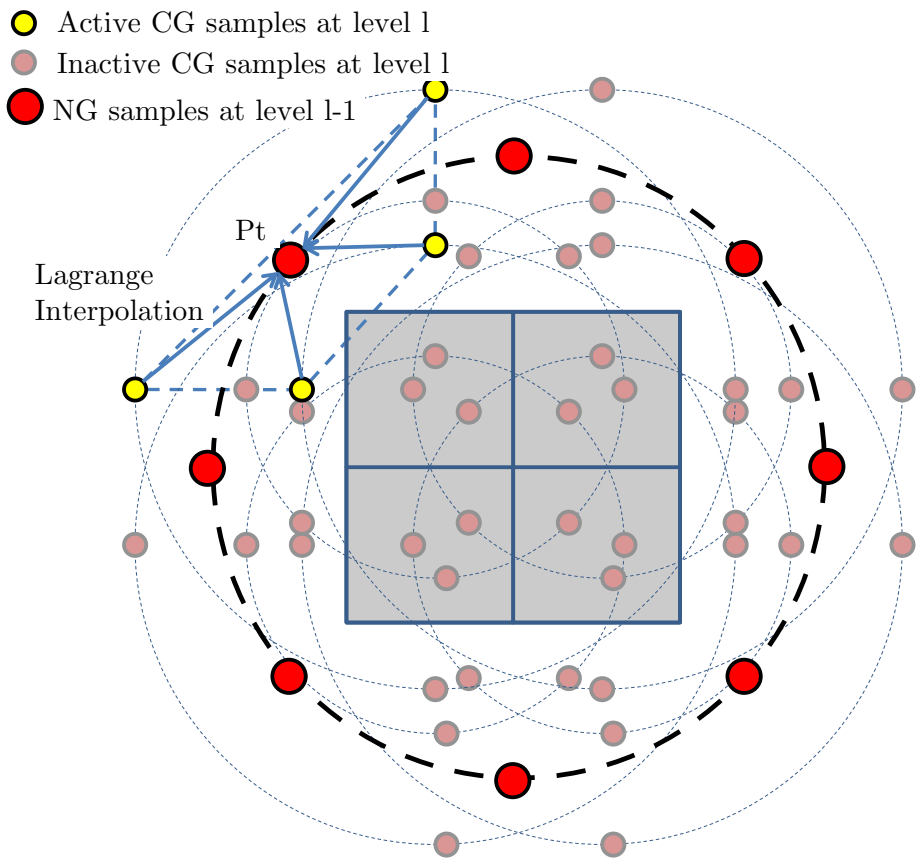


Figure 5 The illustration of Stage 2, aggregating fields on NG samples on coarser levels. The fields on NG samples on the higher levels (red dots) are calculated through interpolations from lower levels (yellow dots).

Stage 3 (NG to CG transitions and CG decomposition/downward pass):

Field values on CG samples are calculated for boxes at all levels. Field values at CG samples of an observer box on a specific level come from two origins. The first is from the interaction-list boxes on the same level. The interaction-list boxes have their parent box as a neighbor of the observer box, except for those having already been taken into considerations in the near-field stage (corresponding to influences of the

source of "medium distance"). The second contribution of field at CG samples comes from their parent box. This contribution is valid only for boxes below the interface level if the calculation is done in mixed-frequency. CG samples of an observer box obtain fields from these two sources through interpolations, from NG samples in the former case, and from CG samples in the latter case.

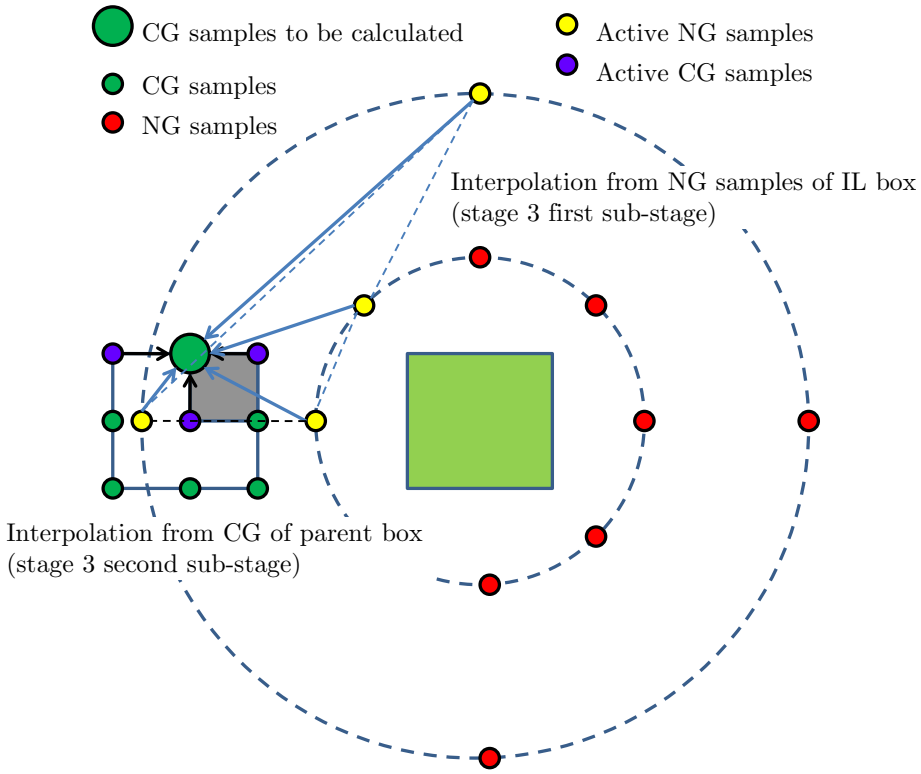


Figure 6 The illustration of Stage 3, calculating fields on CG samples (large green dots) from NG samples of IL boxes (yellow dots) and CG samples of higher levels (green dots).

Stage 4 (CG to observation point):

The field values at actual observation points are obtained by local interpolations from the CG samples of the finest level boxes. The whole process has a computational cost of $O(N)$. Using local interpolations guarantees the adaptivity of the algorithm to non-uniform geometries since the NG samples and CG samples are built and processed only around locations where sources and observers exist.

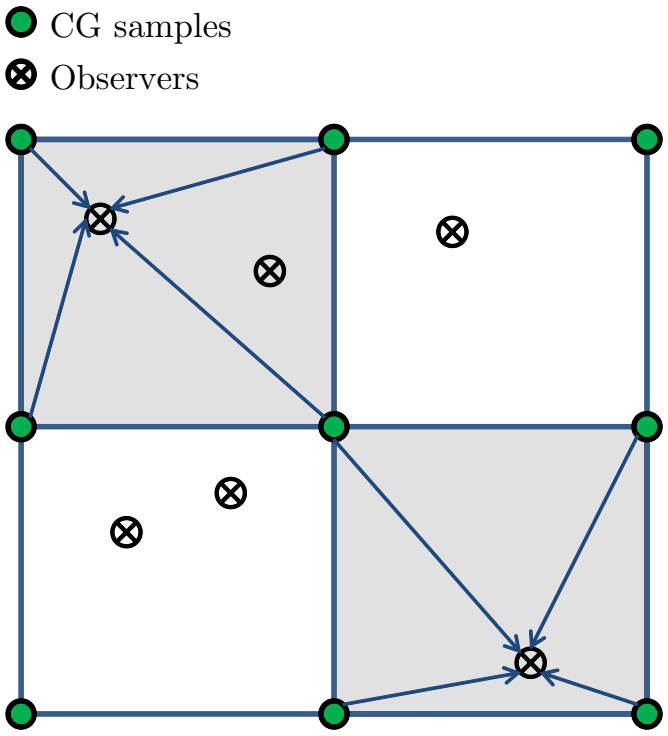


Figure 7 The illustration of Stage 4, calculating field values on observers (cross dots) from CG samples (green dots)

One thing worth mentioning is that, as discussed in previous paragraph, there are no CGs constructed for computations in high-frequency regime, so the CG-CG interpolations and CG-observer interpolation do not exist and the whole downward

pass disappears. In this regime, fields are directly interpolated from the NGs at each level to the observers. In the mixed-frequency regime, the downward pass partially exists, for the *"low-frequency"* levels of boxes. So the downward pass of mixed-frequency calculation is a hybrid scheme of the *"direct NG-observer interpolation"* and *"NG-CG-observer interpolation"*.

4.2.2 GPU NGIM

To implement NGIM efficiently on GPUs, special care should be taken concerning the unique programming model and hardware architecture of GPUs. Coalesced global memory accessing and utilization of shared memory are two critical features, among many others, that should be taken advantage as much as possible. All these concepts and mechanisms have been discussed extensively in Ref. [127], as well as many other works related to scientific computing on GPUs. The implementation of the NGIM on GPUs follows the same "stage-by-stage" protocol as that on CPUs, but contains significant changes.

(a) Preprocessing and initialization stages

In the preprocessing stage, all necessary data structures used by the NGIM are allocated and initiated. The initialization operations include copying coordinates of sources and observers to the allocated matrices on GPUs and reshaping and copying

various auxiliary data, such as the interaction-list. The operations here are executed only once in standard IE solvers for one specific problem.

In addition to the memory transfer operations, another crucial task done in the preprocessing stage specifically for GPUs is rearranging the source amplitude and coordinate arrays so that sources belong to the same box can be found at contiguously places in the memory. This is critical for GPUs to adopt the *coalesced accessing* for accelerated the memory read and write, which will be described in details later. For NGIM, the hierarchical relationship between boxes matches this requirement as the sources that belong to the same box at the finer level belong to the same box at the coarser level too.

In the GPU version of NGIM, the grids are not built in the preprocessing stage. Instead, the position of NG or CG samples and the interpolation coefficients are computed *on-the-fly* when needed. This *on-the-fly* approach reduces the memory consumption and the total memory access operations in the later stages and eventually leads to much better overall performance. As a result, the preprocessing time of the GPU code is reduced by a few orders of magnitude compared with the CPU code, making the NGIM more effective for large problems. It is noted that using a similar approach for FMM-type methods is also possible but it may be somewhat less efficient due to more complex operations in the field evaluation stages, as these methods usually require computing special functions, e.g. the Hankel functions.

(b) Near-field computation

In this stage, the fields at the observers are evaluated directly by accumulating the field contributions from sources belonging to the level L boxes in the near-field region of the observer box. Methods to parallelize this stage also apply to direct evaluations of the classical *"n-body"* problems [129]. Since the computational domain has already been divided into boxes with sources and observers, the traversing the list of observers and sources can be done in a box-by-box fashion.

Mathematically, the near-field computation is a sparse matrix-vector multiplication. However, for many problems, the sparse impedance matrix is too large for GPUs if pre-filled and stored. Therefore, instead of trying to accelerate sparse matrix-vector products directly, we compute all the fields *"on-the-fly"*. Figure 8 shows the flowchart diagram of CPU and GPU implementation of near-field stage, respectively, and Figure 9 shows the thread arrangement and assignment in the same stage. The key points are summarized as follows:

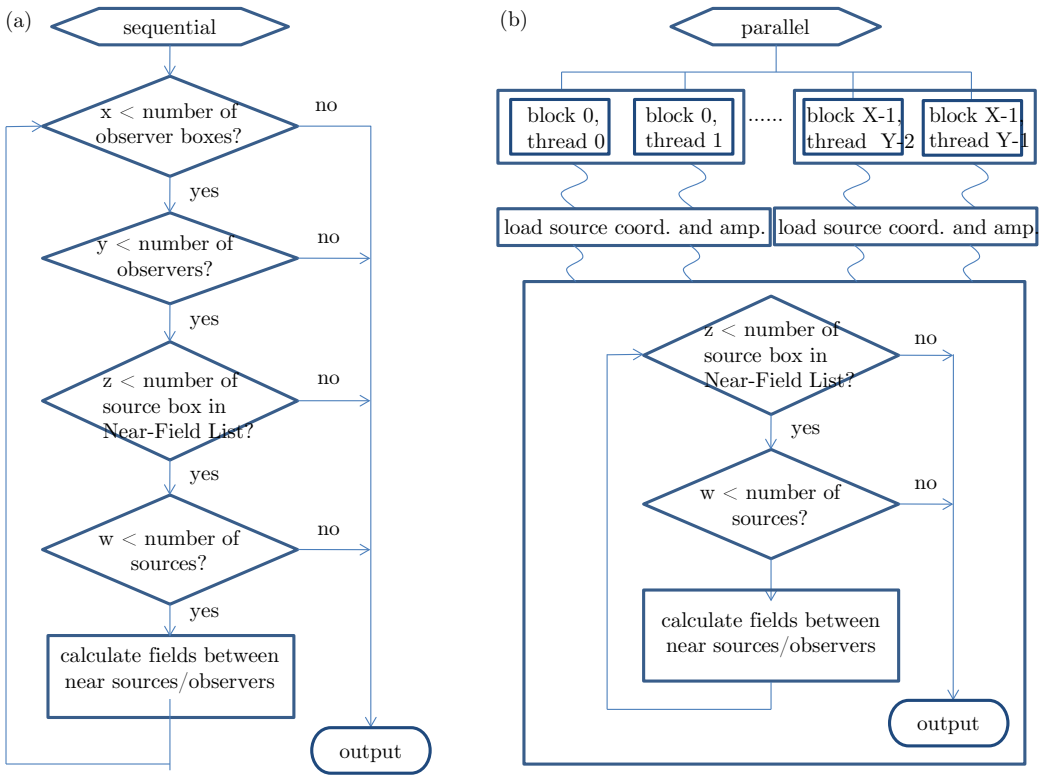


Figure 8 The flow chart of CPU and GPU NGIM. (a) The sequential version of the near-field stage of the NGIM involves a four-level loop that takes into account each source-observer pair satisfies the Near-Field criterion. (b) In the corresponding parallel version of the near-field stage of the NGIM, two levels of loop are spread onto parallel stream processors of GPU. X and Y are number of observer box and number of observers in each box. Coalesced memory loading is utilized and shown in details in Figure 9.

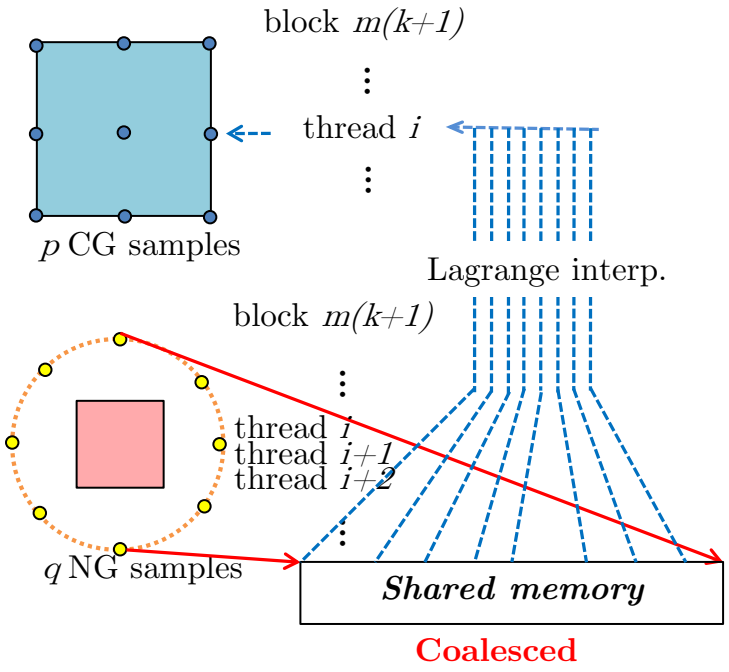


Figure 9 Memory access patterns of threads within the same block. Coalesced global memory is utilized to accelerate the memory loading.

(1) We adopted *"one-thread-per-observer"* type of parallelization, in which one thread is responsible for calculating the field value at one observer.

(2) The number of threads per block can be chosen by the user or determined by the computing hardware, the number of unknowns of the problem, or the source/observer distribution. However, only threads processing observers belonging to the same box are bundled together to form a block. One or several thread blocks might be needed to handle certain boxes when the number of observers in that box is too large. To achieve a better performance, the number of

threads per block should be multiples of 32 on NVIDIA GPUs as this is the size of a warp.

(3) Since observers in the same box have the same set of near sources, those sources are cooperatively loaded from global memory once and in later stages they can be accessed through ultrafast shared memory. This cooperative memory loading is coalesced as the sources belongs to the same box is stored in contiguous locations.

(4) Some intrinsic mathematical functions are used to accelerate the computations. Though not as accurate as their CPU C version counterparts [127], these trigonometric and exponential functions are adequate for NGIM in terms of accuracy and are much faster than standard-binding double precision functions.

Table 1 Computational times and speed-up ratios of the near-field stage

N_p	L	CPU (Xeon X5482)	GPU (GTX 480)	ratio
16	3	2.11e-1	1.02e-3	206.9
32	3	8.63e-1	2.22e-3	388.7
64	3	3.42e0	5.90e-3	579.7
N_p	L	CPU	GPU	ratio
16	4	1.97e0	7.49e-3	263.0
32	4	7.84e0	1.74e-2	450.6
64	4	3.13e1	4.84e-2	646.7
N_p	L	CPU	GPU	ratio
16	5	1.85e1	7.76e-2	238.4
32	5	6.75e1	1.45e-1	465.5
64	5	2.66e2	5.30e-1	501.9
N_p	L	CPU	GPU	ratio
16	6	1.43e2	6.38e-1	224.1
32	6	5.66e2	1.19e0	475.6
64	6	2.22e3	4.37e0	508.0

All timing results shown in this section are in seconds. N_p is the average number of sources per box on level L. The relation between l and N is L.

Table 1 shows the computational time of the near-field stage on CPUs and GPUs. The CPU timing results were obtained on a single core of an Intel Xeon X5248 CPU at 3.2GHz using Intel Fortran Compiler v10 with –O3 optimization. On the GPU

side, an NVIDIA GTX480 at 700MHz with 1.5 GB of memory was used. The GPU implementation was written and compiled using CUDA Toolkit v3.0 from NVIDIA. Both CPU and GPU versions of the code use *"on-the-fly"* approach and the positions of sources and observers are distributed randomly in a cubic computational domain with a uniform probability distribution function.

It is evident that the speed-up ratios between the GPUs and the CPUs are very high, varying between 200 and 650. The speed-ups are higher for larger N_p when the enormous number of parallel processors are fully exploited. Taking into account the fact that the number of the GPU cores in the considered case is 480 and they are run at the clock rate around 4.5 times lower than that of the CPUs, achieving the acceleration ratio above 600 is impressive. Such high rates are obtained not only due to massive floating point computing powers and memory bandwidth of GPUs but also faster manually handled shared memory on GPUs.

A comment should be made on the timing results in Table 1. For fixed number of levels, the complexity of the near-field calculation stage scales quadratically since the number of near-field evaluations is proportional to ${N_p}^2$. The linear complexity of the near-field stage is achieved by the increase of number of levels L with the increase of N. Indeed, the computational time behaves as $O\left(N\right)$ when the number of level L is properly chosen, balancing the near- and far-field computation time.

(c) Outward computation from sources to NG samples (Stage 1)

The NG construction stage computes the field values on the NG grid points, which is the first step of the upward pass of the algorithm. The core operations in this stage are the construction of NGs for each non-empty box at the finest level L and the direct calculation of the field values at these NG samples. The sequential version of this stage consists of three nested loops dealing with all pairs of sources and NG samples for individual boxes and another loop to account for all boxes at level L.

Since the relative positions of NG samples and sources do not change during the whole process of IE solver, their interactions coefficients can be calculated beforehand in the pre-processing stage and stored for later use (i.e. the interacting matrix filling). However, this is only done in our CPU version of code. For GPU, no coefficient matrices are used for the reasons mentioned in previous sections.

In the parallel version of this stage, the "one-thread-per-observer" approach described in the preprocessing stage is also used, but here the "observers" are in their broader definition, referring to NG samples. One or several blocks of threads are allocated for each observation box. For calculation in the low-frequency regime, there are generally a moderate number of NG samples per box, so one block of threads would be enough to achieve the maximal parallel efficiency. In the high- and mixed-frequency regimes, however, the boxes at high-frequency levels may contain many NG samples. This requires assigning multiple blocks to each box. Regardless the number of blocks

assigned to each box, the "coalesced" memory reading technique is always triggered and the global memory accessing is of high efficiency.

Computational times of Stage 1 are presented in Table 2 (these results are frequency independent for the same N, L, and the number of NG samples per smallest box). It is evident that the speed-up ratio increases significantly with the increase of the number of sources per box.

It should be mentioned that, generally, the computational time of stage 1 is about 1-5% of the total time.

Table 2 Computational times and speed-up ratios of the source-to-NG stage (stage 1)

Accuracy requirement	N_p	L	CPU (Xeon X5482)	GPU (GTX 480)	Ratio
L_1 error $= 1 \times 10^{-3}$ for domain size $D = \lambda / 2$	16	3	5.89e-3	1.33e-4	44
	32	3	1.55e-2	1.00e-4	155
	64	3	3.16e-2	1.33e-4	238
	N_p	L	CPU	GPU	Ratio
	16	4	5.01e-2	6.31e-4	79
	32	4	1.21e-1	9.28e-4	130
	64	4	2.45e-1	1.66e-3	148
	N_p	L	CPU	GPU	ratio
	16	5	4.74e-1	3.99e-3	119
	32	5	1.16e0	6.73e-3	172
	64	5	2.44e0	1.24e-2	197
	N_p	L	CPU	GPU	ratio
	16	6	5.49e0	2.99e-2	184
	32	6	1.07e1	5.23e-2	205
	64	6	2.14e1	9.78e-2	219

All timing results shown in this paper are in seconds. N_p is the average number of sources per box on the level L. The relation between N_p and N is $N_p = N / 8^L$.

(d) NG upward aggregation (Stage 2)

In this stage, the field values at the NG samples of the parent boxes at levels from $L-1$ to 2 are computed by interpolation from the NG samples of the corresponding non-empty child boxes. In this paper, we use Lagrange interpolations for phase- and amplitude-compensate fields.

Similar to other stages, the GPU implementation follows the "one-thread-per-observer" parallelization, in which one thread handles one observer and threads handling observers in the same box are bundled to form one or more blocks. The interpolation process includes calculating coordinates of the NG samples of parent boxes, transforming them into the coordinate system of their child boxes, extracting coordinates and amplitudes of the nearest grid samples around the observers, calculating the interpolation coefficients, and finally evaluating the fields. All these operations are are implemented in a single kernel to minimize the stress on global memory. This seems to be stressful for registers, shared memory and ALUs on the GPUs but the actual test runs reveals that the GPUs can handle such computational tasks with ease.

Table 3 shows the computational time results of Stage 2 of the low-frequency case. Note that the results are not shown for different N since this stage depends solely on grids but not on N for a fixed L. (We assume all boxes are active, which means at least one source/observer is presenting in any boxes.) The speed-up ratios are in the same range as those of stage 1 for most problem sizes. It is noted that in the low-frequency regime, stage 2 takes only less than 2% of the total computational time.

Table 3 Computational times and speed-up ratios of the NG aggregation stage (stage 2) in the low-frequency regime

Accuracy requirement	L	CPU (Xeon X5482)	GPU (GTX 480)	Ratio
L_1 error $= 1 \times 10^{-3}$ for domain size $D = \lambda$ / 2	3	2.70e-3	1.33e-4	20
	4	2.74e-2	3.96e-4	69
	5	2.25e-1	1.77e-3	127
	6	1.81e0	8.33e-3	217

All timing results shown in this paper are in seconds. The number of NG samples per box is chosen as 64 due to the accuracy requirement.

Table 4 shows the computational time of the GPU code in the high-frequency regime. The absolute computational time is noticeably larger than that of the low-frequency case in Table 3 due to larger number of necessary operations. It is noted that no CPU results are shown for this case because the increased number of grid points for sampling high-frequency fields make the current CPU code consumes way too much memory. For the same reason, in this thesis, high-frequency NGIM results for other stages are shown for the GPU code only.

Table 4 Computational times and speed-up ratios of the NG aggregation stage (stage 2) in high-frequency regime

Accuracy requirement	L	GPU (GTX 480)
$L_1 = 5 \times 10^{-2}$	3	4.14e-4
	4	2.92e-3
	5	3.59e-2
	6	3.89e-1

* All timing results shown in this paper are in seconds. No CGs are constructed. The number of NG samples per box at level l is $64 \times 8^{L-l}$, and the frequency for a given L is chosen as $D = \lambda\big(L-2\big)$.

(e) Evaluation of field values on CG samples (Stage 3)

In Stage 3, the field values at the samples of CGs are calculated. This stage can conceptually be treated as a two-step process in low-frequency regime as shown in Figure 10 but as a one-step process in high-frequency regime. In the two sub-stages in low-frequency regime, each of the two origins of fields, as mentioned in Section 4.2.1, are accounted. In the high-frequency regime, fields on observers are obtained directly via interpolations. In the mixed-frequency regime, the process is a combination of those two as described in Ref. [98]. The "one-thread-per-observer" approach is used as the "observers" now are CG samples for low-frequency calculations and actual observers for high-frequency calculations.

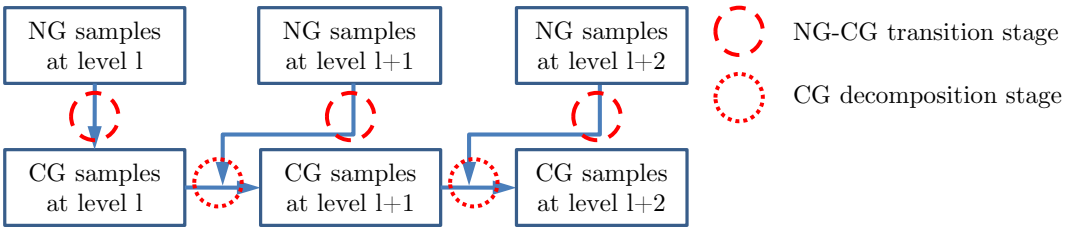


Figure 10 The relationship of two sub-stages: NG-CG transition stage and CG decomposition stage in calculating the field values on CG samples of boxes at each computational level in the low-frequency regime.

The sub-stage that evaluates fields at CG samples from NG samples in the interacting far-field boxes is the most time consuming part of the entire far-field calculation. The major reason is the enormous number of operations involved. For each box, there are at most 189 boxes in its interaction-list, if the near-field range is defined at the closest neighbors of a certain box. This number could go much higher for an *increased* near-field range. However, there are several other reasons to make this sub-stage the bottleneck of the calculation as well. To name a few, for a certain observer box, the interaction-list boxes are not necessarily situated contiguously in the memory which lowers the global memory cache hit-rate.

The CG decomposition sub-stage is executed for CG samples in all observer boxes at the low-frequency levels after the NG-CG transition sub-stage. On each such level, CG samples in the child boxes are obtained from the CGs samples in the parent boxes via Lagrange interpolations. This step is similar to the NG-NG aggregation stage (Stage 2) but in the opposite direction. Computationally, it is much simpler as no

spherical-Cartesian coordinate transformations are needed. The total contribution of this sub-stage to the overall computational time is low, only about 1-2%.

Table 5 shows the computational time of Stage 3 (NG-CG transition stage and CG decomposition combined) for different L in the low-frequency regime. Similar to the results in Table 3 (Stage 2), for a fixed number of levels L, the speed of Stage 3 is independent of the problem size N and hence no dependence of N is shown. The obtained computational times depend on the number NG samples and CG samples constructed for each box. The speed-up ratio is at least 100, which is in the same order to other far-field stages (comparing with significantly lower speed-up for similar stages in GPU implementation of MLFMMs from Ref. [41, 62]). The speed-up ratio increases for increasing oversampling rates as GPUs use those extra operations to fully saturate their thousands of stream processor. In addition, from our tests on different generations of GPUs, we found that the Fermi GPUs (GeForce GTX 480) handle kernels with access to relatively "random" sets of data much better. This is due to the improved architecture with L1 cache and better organized SMs. The result is three-fold computational time reduction on GeForce GTX 480 as compared to Tesla C1060. This time reduction is interesting taking into account that the number of stream processors in GTX 480 is only twice as large (480 vs. 240) and with very similar core frequency.

Table 5 Computational times and speed-up ratios of field to CG stage (Stage 3) in the low-frequency regime

Accuracy requirement	L	CPU (Xeon X5482)	GPU (GTX 480)	Ratio
L_1 error $= 1 \times 10^{-3}$ for domain size $D = \lambda$ / 2	3	2.68e-1	2.44e-3	110
	4	3.02e0	2.32e-2	130
	5	2.95e1	2.11e-1	140
	6	2.43e2	1.82e0	134
	L	CPU	GPU	Ratio
L_1 error $= 2.5 \times 10^{-4}$ for domain size $D = \lambda$ / 2	3	3.23e0	2.10e-2	154
	4	4.16e1	2.32e-1	179
	5	3.38e2	2.18e0	155

Times are shown in seconds.

Table 6 Computational times and speed-up ratios of field to CG stage (Stage 3) in the high-frequency regime

Accuracy requirement	L	GPU (GTX 480)
L_1 error $= 5 \times 10^{-2}$	3	3.33e-3
	4	4.79e-2
	5	6.16e-1
	6	6.97e0
	L	GPU
L_1 error $= 1.5 \times 10^{-2}$	3	2.84e-2
	4	4.39e-1
	5	5.33e0

The times shown are in seconds. The number of NG samples per box at level l

Table 6 shows the computational times of stage 3 in the high-frequency regime. The time increases compared to the low-frequency case in Table 5 but this increase is relatively insignificant (on the order of the number of levels L), which demonstrates the efficiency of the code in the high-frequency regime.

(f) CG grids to observers (Stage 4)

In this stage, the fields at actual observers are interpolated from the CG samples of the finest level L boxes. This stage is only valid in low- and mixed-frequency regime. This stage is conceptually reciprocal to the Stage 1. All critical designing strategies in Stage 1 are followed, including the "one-thread-per-observer", coalesced loading of sources, etc. The computational time of this stage is usually smaller than that of stage 1, because the interpolation operations are less demanding than the direct field calculations through the Green's functions.

Timing results of Stage 4 are presented in Table 7. The general trend of computational times is similar to that of Stage 1 (Table 2). The GPU computational times are constant for smaller problem sizes and grow up to a saturation point of around 150 when the problems size increases. We have also tested more cases with increased CG oversampling rates. We found that the increase of the CG oversampling rates barely affects the computational times on either CPUs or GPUs, even though with more CG samples per box, more data has to be loaded before doing the interpolations. For the CPUs, the reason might be that the time spent memory loading time is negligible

compared with that of calculations, while for the GPUs the memory loading time is small due to coalescent access.

It should be mentioned that, generally, the computational time of stage 4 is below 1% of the total time. Therefore, the influence of this stage is insignificant provided other stages are implemented efficiently.

Table 7 Computational times and speed-up ratios of CG-to-receiver stage (stage 4)

N_p	L	CPU (Xeon X5482)	GPU (GTX 480)	ratio
16	3	1.18e-3	1.50e-4	8
32	3	1.99e-3	1.50e-4	13
64	3	3.62e-3	1.70e-4	21
N_p	L	CPU	GPU	ratio
16	4	9.77e-3	3.30e-4	30
32	4	1.68e-2	3.30e-4	51
64	4	3.53e-2	5.06e-4	70
N_p	L	CPU	GPU	ratio
16	5	1.03e-1	1.50e-3	69
32	5	1.84e-1	1.50e-3	123
64	5	3.51e-1	2.60e-3	135
N_p	L	CPU	GPU	ratio
16	6	9.01e-1	9.90e-3	91
32	6	1.58e0	1.01e-2	156
64	6	2.95e0	1.79e-2	165

All time shown in this table is in seconds. There are 64 CG samples per box. N_p is the average number of sources per box on the level L. The relation between N_p and N is $N_p = N / 8^L$.

4.2.3 Overall results

In this section, we present and analyze the overall performance of the NGIM algorithm on CPUs and GPUs for low-, high- and mix-frequency problems. The asymptotic time and space complexity of the algorithm are shown to meet the theoretical hypothesis. The speed-ups between GPUs and CPUs are astonishingly high and we listed several reasons that might contribute.

(a) Computational time for low frequency problems

We present the computational times in the low-frequency regime in this section. The overall performance of CPU and GPU implementations of NGIM is shown in Figure 11 and Table 8. The GPU implementations have been tested on two generations of NVIDIA GPUs: the Tesla C1060 with 4 GB memory (GT200 architecture, with CUDA compute capability 1.3) and new generation Geforce GTX 480 with 1.5 GB memory (Fermi architecture, CUDA compute capability 2.0). Since the accelerations brought by GPUs vary across different stages and are closely related to the problem sizes, optimal performances of respective CPU and GPU version are achieved under different parameters. As a result, similar to [62], we define "effective" speed-up ratios as the ratios between optimal computational times achieved on CPUs and GPUs.

In Figure 11, the computational time of the direct method, i.e. the evaluation of each source-observer pair on CPUs and GPUs, are provided as a reference. The direct method has $O\left(N^2\right)$ complexity. Computational times of the NGIM shown as solid and dashed lines, scale as $O\left(N\right)$ for both CPU and GPU NGIM when the optimal number

of levels is chosen respectively. It can be observed that each curve actually consists of several pieces of curves, each of which corresponds to computational times achieved under a certain L. At some points, L has to be increased in order to balance the near- and far-field time. This leads to the overall linear increase of computational time with respect to N. We also see that the "cross points" that the optimal L changes are different for GPU and CPU cases. This is due to different acceleration ratios of near- and far-field components.

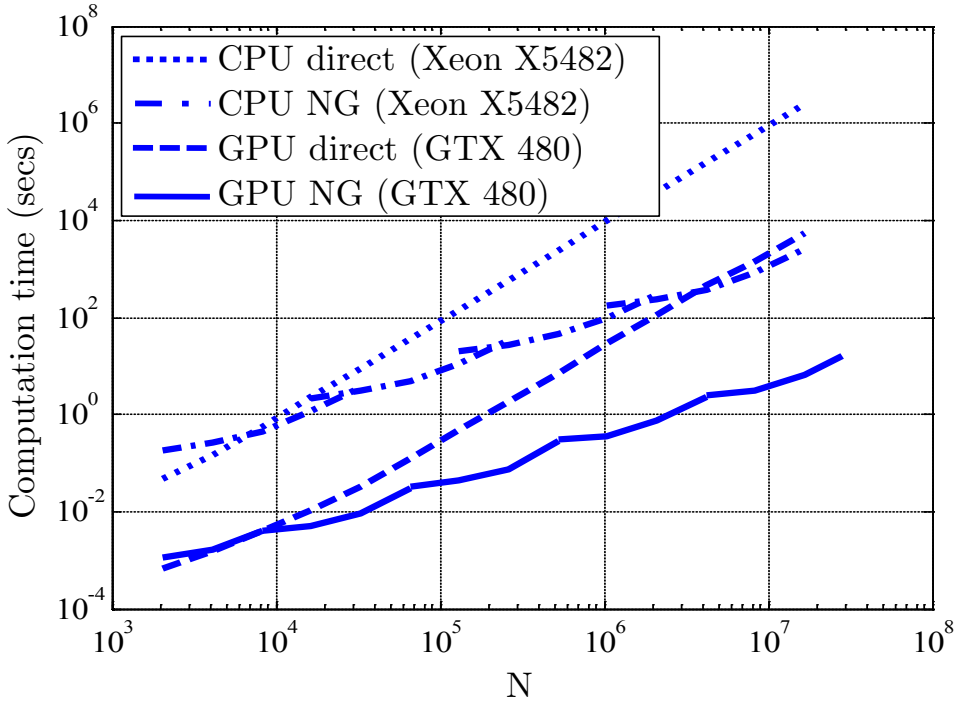


Figure 11 Computational times of the direct method and multi-level NGIM on CPU and GPU as a function of N in the low frequency regime. The time of all necessary memory transfer between the hosts and the GPU devices are included, as will be the case for all other timing results in this section. The size of the computational domain is $D = \lambda / 2$. The relative L_1 error is approximately 5×10^{-3}.

The GPU code is significantly faster in the entire test range. The largest problem size N that GeForce GTX 480 can handle is 28 million (1.5 GB memory) while Tesla C1060 can handle $N = 64\,M$ (4 GB memory). As a comparison, the CPU code can run up to 16 million with 32 GB of memory. Since the GPUs accelerate the near-field calculations better than the far-field calculations, the cross points of curves between neighboring levels all shift towards larger N on GPUs. It is remarkable that the breakeven point of the computational time between the GPU direct code and the NGIM CPU code is around $N = 4\,\mathrm{M}$. The breakeven point between the GPU direct code and the NGIM GPU code is approximately $N = 4\ \mathrm{K}$.

Table 8 shows a detailed list of the computational times. For a problem with $N = 2^{24}$, the computational time is only 6.36 seconds, which is 392 times faster than the CPU version of NGIM, 862 times faster than the GPU direct version, and 7 million times faster than the CPU direct version (estimated). The comparison between the GPU NGIM code running on Tesla C1060 and GeForce GTX 480 shows around two-fold speed increase of the latter, which is consistent with the two-fold increase of the number of stream processor in GeForce GTX 480 (480 vs. 240). As having been discussed in the previous paragraphs, the speed-up of GTX 480 compared to Tesla C1060 of the far-field regime is around 3 times. We attribute this fact to the improved architecture of Fermi cards, allowing easier handling more complex memory loading and thread arrangement required for the far-field calculation. However, for near-field calcu-

lations, the computation is almost brute force so it is natural to obtain approximately 2 times of speed-ups between the two generations of cards.

Table 8 Computational times and speed-up ratios of the CPU and GPU NGIM

# of Unknowns N	2^{14}	2^{16}	2^{18}	2^{20}	2^{22}	2^{24}	2^{26}
CPU Time	1.15e0	4.84e0	2.69e1	9.66e1	3.67e2	2.49e3	N/A
GPU (GTX480)	5.19e-3	3.09e-2	7.49e-2	3.69e-1	2.33e0	6.36e0	N/A
Speed-up	222	157	359	262	152	392	N/A
GPU (C1060)	1.15e-2	5.29e-2	1.53e-1	8.14e-1	3.85e0	1.18e1	8.49e1
Speed-up	100	90	176	119	95	211	N/A

The GPU and CPU implementations of the NGIM and the direct method for $\Omega_r = \Omega_a = \Omega_x = 2$. The size of the computational domain is $D = \lambda/2$. The relative L_1 error is approximately 5×10^{-3}.

Table 9 lists the computational time when the NG and CG grids are over-sampled to improve the accuracy of the calculation. Obviously, the computational time increases due to more NG and CG grid samples to be processed in the far-field stages. Qualitatively, the performance of the CPU and GPU versions is similar to the low oversampling case but GPU code handles the increased computational burdens much better. In fact, the "L_1 error $= 2 \times 10^{-2}$" case runs at the same speed as "L_1 error $= 5 \times 10^{-3}$" case. This is because our current version of the code uses at least one warp to handle one observer box, as explained in Section 4.2.2. In these two cases, NGIM uses 8

and 27 CG samples per box, respectively, both less than the warp size 32. Thus the computational times are the same.

Table 9 Computational times and speed-up ratios of the GPU and CPU NGIM with oversampled grids

# of Unknowns N			2^{20}	2^{23}
L_1 error $= 2\times10^{-2}$	CPU NG	Time (sec)	5.29e1	4.46e2
	GPU NG	Time (sec)	3.69e-1	3.11e0
		Speed-up	143	143
L_1 error $= 5\times10^{-3}$	CPU NG	Time (sec)	9.66e1	8.14e2
	GPU NG	Time (sec)	3.69e-1	3.11e0
		Speed-up	262	262
L_1 error$=1\times10^{-3}$	CPU NG	Time (sec)	3.43e2	N/A**
	GPU NG	Time (sec)	1.02e0	8.87e0
		Speed-up	336	N/A

** The CPU version of code requires more than 32 GB of RAM. The accuracy of each simulation is shown in the first column. The size of the computational domain is $D = \lambda/2$.

Finally, the performances of NGIM for surface problems are shown in Table 10. The simulation is done in the low-frequency regime and all sources are placed on the surface of an “inverse T-structure". For this problem, FFT-based methods that will be described in Section 4.3 would need to build a grid enclosing the whole computational

domain, including the vast empty spaces, resulting in excessive computational time and memory consumption. As evident from Table 10, the CPU and GPU implementations of the NGIM have performance similar to (and even better than) that obtained for the source/observer distribution in a box. The GPU-CPU speed-up ratios are high as well.

Table 10 Computational times and speed-up ratios of the GPU and CPU NGIM for the surface source-observer distribution of the "inverse-T" structure in Figure 12

# of Unknowns N		8,192	32,768	131,072	524,288
CPU NG (Xeon X5482)	Time (sec)	4.35e-1	2.03e0	8.34e0	4.41e1
GPU NG (GTX 480)	Time (sec)	2.22e-3	6.33e-3	2.16e-2	1.12e-1
	Speed-up	196	321	386	394

* The size of the computational domain is $D = \lambda / 2$

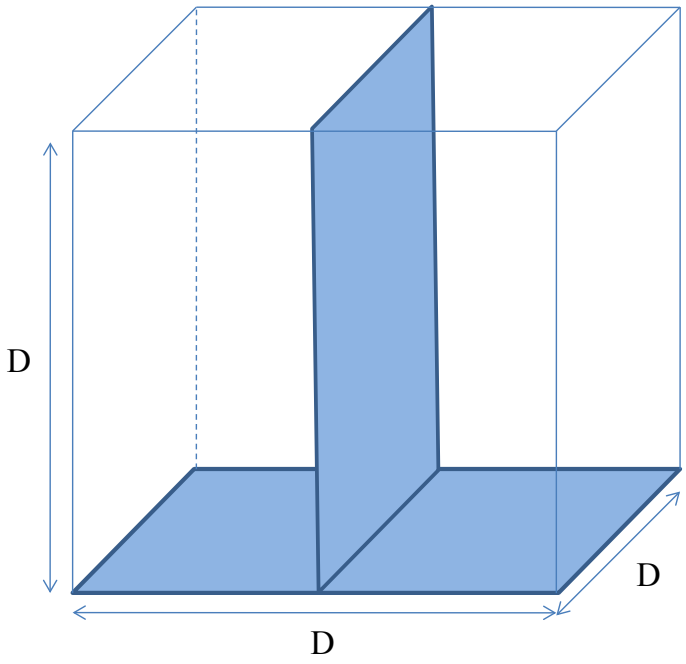


Figure 12 Sources distributed on two surfaces forming an "inverted T" structure with the lateral length equals D.

(b) Computational time for high- and mixed-frequency problems

Next we show the computational time results of the GPU NGIM code in the high- and mixed-frequency regimes. The trend of the computational times is qualitatively similar to that in the low-frequency regime and hence many observations and conclusions in the low-frequency regime apply here as well. In Figure 11 and Table 12, we present a quantitative summary of the results in a format similar to that of Table 8. Table 11 shows the computational time in the high-frequency regime for the number of sources up to $N = 14$ million and domain sizes up to $D = 12\lambda$. The trend of the computational time is consistent with that in the low-frequency regime, with a reason-

able increase due to higher grid density. As in the low-frequency case, the speed increase brought by the newer generation GPU is around two-fold.

Table 11 The computational time of the NGIM on GPUs in the high-frequency regime

# of Unknowns N	8K	64K	256K	1M	4M	14M
D / λ	1.0	2.0	3.0	5.0	8.0	12.0
Level L	3	3	3	4	5	5
GPU Time GTX 480	5.0e-3	2.8e-2	2.0e-1	9.6e-1	4.9e0	2.1e1
GPU Time TESLA C1060	1.4e-2	6.1e-2	3.8e-1	1.8e0	1.0e1	4.4e1

The number of sources N is taken such that there are around 20 sources per a linear wavelength. The density of NGs are to keep the L_1 error less than 5×10^{-2} for all problem sizes.

Table 12 shows the computational time in the mixed-frequency regime with the hybrid NG-CG transformation scheme. Computational times for the same N are noticeably smaller comparing with those in the high-frequency regime. This means that the hybrid scheme does save some operations due to constant CG samples in the low-frequency levels.

Table 12 Computational times of the NGIM on GPUs in the mixed-frequency regime

# of Unknowns N	64K	256K	1M	4M	8M	64M*
Level L	3	4	5	5	6	6
GPU Time (GTX 480)	2.6e-2	9.2e-2	3.7e-1	1.8e0	3.1e0	N/A
GPU Time (TESLA C1060)	4.4e-2	1.7e-1	8.6e-1	3.1e0	7.0e0	8.0e1

The domain size is set to be $D = 2\lambda$. The L_1 error is 6.0e-2 for all cases. For the case $N = 64\text{M}$, the optimal level L should be 7 but due to the memory limitations the results are shown for $L = 6$.

(c) Memory usage

Memory usage is a very important factor affecting the applicability of GPU-accelerated algorithms since GPUs typically have smaller amounts of memory than CPUs. For example, in the Dell Precision T7400 workstation we used to test our algorithm, any core of the two Xeon processors can take the entire 32 gigabytes of the main RAM. In our GPU platform, NVIDIA GeForce GTX 480 has 1.5 GB global memory.

In our current version of the code, the memory usage of GPU code is determined by both L and N. With 1.5 gigabytes of memory, NVIDIA GeForce GTX 480 can go up to $L = 6$ and up to $N = 28\text{M}$. The memory consumption of our CPU NGIM is significantly larger (while at the same time, much slower). For example, the memory required by NGIM CPU implementation for a problem of $N = 8\text{M}$ is 18.1 GB for the same accuracy requirements. This is almost 50 times more than that of the

GPU code. Most of the CPU memory is used for storing the interpolation coefficient matrices, impedance matrices and other grid sample information, which if eliminated, would make the CPU code several orders of magnitude slower and completely not usable.

4.2.4 Summary and future directions

From this section, it is clear that the NGIM is a very good candidate for massive parallelization including on GPUs. Furthermore, the high-frequency NGIM method, without the CG, would be very easy to scale to multiple nodes on high-performance computing clusters. These options will be explored in the next revision of the NGIM code.

4.3 Box Adaptive Integral Method (B-AIM)

Box Adaptive Integral Method (B-AIM) is one variation of FFT-accelerated fast methods that are designed specifically for massively parallel computing architectures. Comparing with the other popular variations, B-AIM has the same computational complexity but much higher execution efficiency on current and possibly future many-core computer systems, including GPGPUs.

All FFT-based algorithms follow similar philosophy and flow. In these algorithms, two auxiliary sparse uniform grids are created, one interacting with sources,

called the source grid and the other interacting with observer, called observer grid. The source grid points are used as virtual sources participating in the source-observer field calculation through FFT. The observer grid points are used as the virtual observers. After the fields on the observe grid points are where the fields on actual observers are interpolated from. The process of calculating the amplitudes on those "virtual sources" are called "projection" or, as in Ref. [36, 177], "anterpolation" and the reciprocal process of obtaining field strengths on actual observers from the "virtual observers" is called "interpolation". The projections and interpolations introduce errors and the errors might be unacceptably large when sources and observers are too close to each other. B-AIM, AIM and pFFT algorithm all have correction mechanisms to neutralize this inaccuracy. For each observer, interactions from sources residing within a certain range of observers are identified as "near-fields", and are supposed to be inaccurate. Since they are inseparable from other "far-fields" while being calculated through FFT in the first pass, they have to be calculated separately again and subtracted. Then, accurate near-fields are added through direct superposition. Detailed description of the process can be found in Ref. [11, 94, 136].

4.3.1 Procedure of B-AIM

The B-AIM algorithm presented in this chapter does have similar stages as the traditional AIM and pFFT algorithms but it follows a different approach in projections and corrections. In the following, we describe the stages of the algorithm.

Preprocessing (Stage 0):

The operations described in this stage are only to be done once before the iterative solution of fields starts. The computational domain is divided into multiple subdomains, called boxes, as shown in Figure 13. The number of boxes can be set by the user or by some criteria such as memory usage or computational time requirement. Boxes with no sources or observers are excluded from the computations. Two grids are constructed for each non-empty box, one for emulating the field generated by actual sources, referred to as the source grid and the other for estimating the field resulted on actual observers, referred to as the observer grid. In many cases (e.g. for free space problems) these two grids can overlap, thus saving half of the storage space, but for some specific cases, such as periodic problems, these two grids are shifted to ensure fast convergence of the periodic Green's function [100]. These grids of individual boxes are then combined and form two larger grids covering the whole computational domain. These two large grids can be expressed as two N_g-length vectors, $\mathbf{I}$ and $\mathbf{U}$, respectively.

After the grids have been established, the algorithm proceeds as follows

Projection (Stage 1):

The source amplitudes are projected from actual sources to the source grid points for them to emulating the influence of the actual sources. Lagrange interpola-

tions are used. The result of the projection operation can be expressed as an $N \times N_g$ matrix $\mathbf{V}$, so that $\mathbf{I} = \mathbf{VQ}$.

Grids interaction calculation (Stage 2):

Field generated by the source grid points are calculated at each observer grid points via a convolution $\mathbf{U} = \mathbf{G_{grid}} \otimes \mathbf{I}$. This is done by convolving the grid sample matrix with the Green's function matrix via FFT.

Interpolation (Stage 3):

In this stage, the fields at actual observers are found by interpolating from the field values of observer grid points. The interpolation operation is the reciprocal operation of the projection so can be expressed as the transpose of $\mathbf{V}$, so that $\tilde{\mathbf{F}} = \mathbf{V^T U}$ where $\widetilde{\mathbf{F}}$ is an approximation of $\mathbf{F}$. Combining the above equations, this coarse estimation can be summarized as $\widetilde{\mathbf{F}} = \mathbf{V^T}\text{invFFT}\{\text{FFT}\{\mathbf{G_{grid}}\}\bullet\text{FFT}\{\mathbf{VQ}\}\}$.

Near field correction (Stage 4):

The approximation near-field parts of $\widetilde{\mathbf{F}}$ are substituted by field values computed directly. This substitution requires a second pass of the previous stages and direct calculation of a portion of $\mathbf{Z}$. This process has $O(1)$ complexity for a single observer and $O(N)$ complexity overall. This correction can be summarized as

$$\widehat{\mathbf{F}} = \widetilde{\mathbf{F}} - \mathbf{V_{near}}^T(\mathbf{G_{grid\_near}} \otimes (\mathbf{V_{near}Q})) + \mathbf{Z_{near}Q}$$

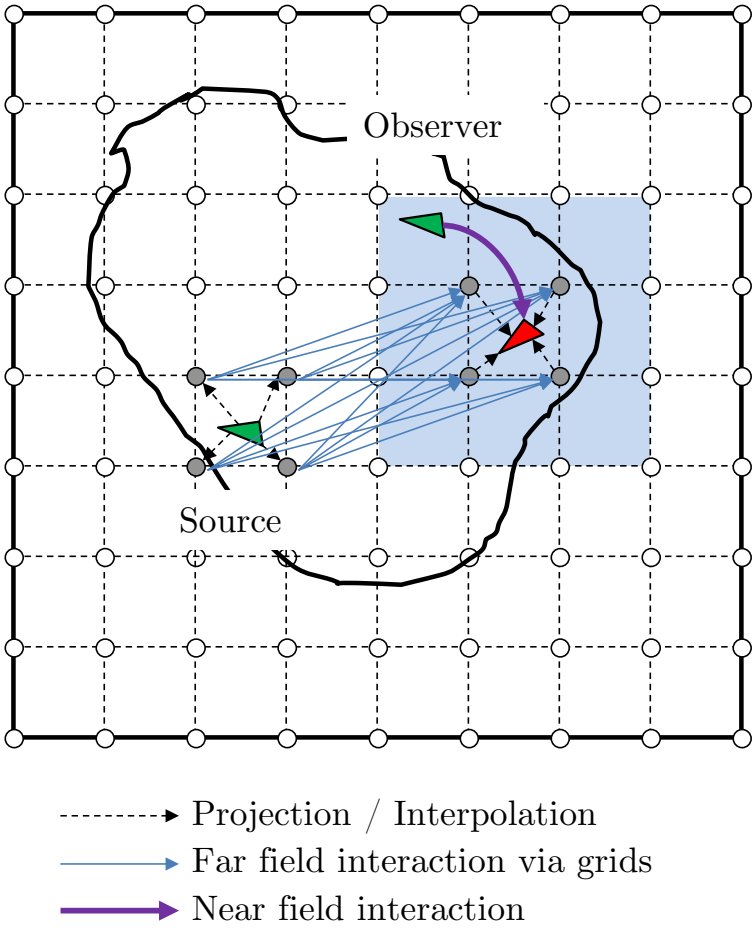


Figure 13 Schematic illustration of B-AIM. Subtraction of inaccurate near-field is not shown as they follow the same procedure as the projection stage.

4.3.2 The GPU implementation of B-AIM

As stated and restated several times in the thesis, implementing any algorithm efficiently on GPUs requires it adapting to the hardware architecture. The B-AIM is no exception.

(a) “On-the-fly” calculation:

In sequential AIM/pFFT implementations, the operations shown in Section 4.3.1 are usually implemented as several matrix-vector multiplications and the matrices are usually pre-filled in the processing stage to save CPUs from calculating the unchanging matrix elements at very iteration. This is not done in the GPU B-AIM because the GPUs are very good at arithmetic calculation and have much less memory to store the pre-filled tables. So, GPUs calculate each matrix element right before the matrix-vector multiplication is required. This is called the “*on-the-fly*” mechanism, and has been explained in the Section 4.2. This technique inevitably increases the total number of arithmetic operations but, with appropriate task arrangement actually utilizes GPUs’ extremely wide SIMD SMs well. The final results in Section 4.3.3 shows that the *“on-the-fly”* approach produces incredible speed-ups comparing with the CPU AIM/pFFT that uses the prefilling approach.

(b) Box-level domain decomposition and regulation:

Another critical issue in the tradition AIM algorithm that limits its efficiency on GPUs is the inhomogeneous computational burden across sources. This happens in the projection, interpolation as well as in the near-field correction stage. For example, in the near-field correction stage, common implementations of AIM maintain a list of near sources for each observer (or basis function). The fields exerted by the sources on this list are to be corrected as the corresponding sources are too close to be calculated

via FFT. In order to do this, a sparse matrix is formulated with its non-zero elements representing the Green's function between these near-sources and observers. However, non-zero entries are highly dependent on the geometrical distribution of sources and can be unstructured. Accelerating unstructured sparse matrix-vector multiplication has been shown in many research literatures to have relatively limited GPU-CPU speed-ups [10, 88]. AIM implementations adopting this approach confirm the conclusion by showing less than 10x speed-ups [51, 133].

Our B-AIM solves this problem by grouping close sources and observers into boxes and replacing the separate near-source list of each observer by a unified near-source list shared between observers in the same box. As mentioned in Section 4.3.1, the computational domain is divided into subdomains called "boxes". Sources and observers are then associated with their enclosing boxes. The relationship between a pair of source and observer is no longer determined by their distance but by the boxes they belong, whose relationship is in turn determined by the distance between the centers of themselves.

Using this double-layer mapping scheme, observers that belong to the same box always have the same near and far field sources and the same interpolation grid points too. In fact, adopting this box-to-box mapping eliminates the near-source lists altogether because any observer box can find their near boxes using their index numbers alone. This not only saves the precious memory resources of GPUs but also dramatically

reduces the number of global memory access, which is a critical for improving the execution efficiency of the code.

(c) Pre-sorting:

The source information such as the coordinates and amplitudes are pre-sorted in the preprocessing stage, after the box decomposition, so that the source information belonging to the same box occupies contiguous memory spaces. This increases the data locality of the algorithm, so the code would have much higher global memory cache hit rate.

(d) Block-box mapping:

Similar to the tactics we adopted in the NGIM, one block of threads are always responsible for processing observers that belong to the same box. This intrigues coalesced memory access that are critical for achieving high global memory accessing throughput, especially on older architectures like G80 or GT200. This block-box mapping techniques also significantly improves the shared memory utilization so that global memory access can be further reduced.

(e) Lagrange projection and interpolation schemes:

There are many projection and interpolation schemes to be chosen for AIM or pFFT, and comparison studies have been done by researchers [174]. However, traditionally the selection of schemes rarely considers their computing efficiency on specific

hardware architectures. In the B-AIM algorithm, Lagrange interpolation scheme is chosen for both projection and interpolation because they can be done on GPUs through either hardware texture filtering units using intrinsic commands or through Lagrange polynomials evaluations that can take advantage of the very fast constant memory. In other schemes, the interpolation coefficients would require enormous amount of operations such that they have to be calculated in the preprocessing stage and tabulated for future usage, which are ineffective or infeasible for GPUs.

4.3.3 Computational complexity and result analysis

In principle, the user would keep the average number of sources per box to be a constant. This means the number of boxes and the total number of grid samples is proportional to number of sources N. This leads to $O(N_q \log N_q)$ computational complexity of the stage 2. Stages 1, 3, and 4 contain only local operations so they have an $O(N_q)$ complexity. The overall asymptotical complexity of the algorithm is $O(N \log N)$. The memory complexity of the algorithm is $O(N)$.

Computational times of B-AIM are shown in the Table 13 for cubic and linear order of interpolations with 1e-4 and 1e-5 average L1 error, respectively. The parallel GPU B-AIM using a single NVIDIA Geforce GTX 680 card is compared against serial CPU B-AIM using one core of Intel i7-950 CPU. All computational times are obtained using optimal settings for CPU and GPU code. We can see that the times of both the CPU and GPU codes have close to linear scaling starting from very small problems

sizes. The speed-ups between GPU and CPU implementations are around 100-200. For example, one field evaluation using GPU B-AIM costs 0.246 secs in cubic settings, for a problem of 1 million unknowns, which is 162x and 143x faster than CPU B-AIM and GPU direct method, respectively.

Table 13 Computational times of serial B-AIM on CPU and parallel B-AIM on one GPU card

N	Direct CPU	Direct GPU	B-AIM CPU (cubic)	B-AIM GPU (cubic)	B-AIM GPU (linear)
16K	7.02e-1	3.74e-3	2.65e-1	3.10e-3	2.10e-3
64K	4.47e1	1.45e-1	2.28e0	8.40e-3	5.70e-3
256K	7.17e2	2.13e0	9.87e0	3.28e-2	2.05e-2
1M	N/A	3.53e1	3.99e1	1.34e-1	8.89e-2
4M	N/A	N/A	N/A	5.76e-1	3.90e-1

It is interesting to see that the speed-ups between GPUs and CPUs B-AIM far exceed the difference of their respective theoretical computing power. The reason is two-fold. It happens that the commercial x86 CPUs lacks accessible APIs to exploit the massive parallelization opportunities in the B-AIM algorithm. Developers usually rely on automatic optimization and vectorization from compilers or directive-based OpenMP APIs to utilize the cores on chips. Two x86 CPU vendors, Intel and AMD do provide SSE or AVX vectorization commands, but they are at the assembly language level so it requires extraordinary efforts to use. The other reason is that GPUs' onboard memory has much higher throughput. Even the slowest "main" memory, which is called the global memory in CUDA and OpenCL API, is two generations advanced of the

main memory of CPU and has a much wider bus. The shared memory on die provides several terabytes per second throughput and can be accessed by tens of SPs at the same time. With appropriate accessing patterns, these super-fast memories may provide one or two orders of magnitude larger throughput, comparing with the CPUs' main memory. With those being said, we should be aware that any comparison between CPUs and GPUs in term of the absolute times should be made with clear explanation of the implementation, optimization and execution environment. Obviously, the speed-ups between CPU and GPU vary a lot among different algorithms [88], and this is due to the nature of mathematical operations involved.

The memory consumption of B-AIM is shown in Table 14. The asymptotical complexity of the memory consumption of B-AIM is $O(N)$. For problems with the same size, the cubic interpolation B-AIM uses approximately 6 times more memory due to larger grids.

Table 14 The memory consumption of B-AIM under different interpolation schemes

N	16K	64K	256K	1M	4M
Linear	0.8M	3.2M	9M	31M	111M
Cubic	6.9M	14.9M	43M	163M	673M

4.3.4 Multi-GPU B-AIM

The B-AIM algorithm has been further parallelized across multiple GPUs. The flow chart of the multi-GPU B-AIM is shown in Figure 14. The main difference between the multi-GPU version and the single-GPU one is multiple scattering and collecting stages between B-AIM stages. This is necessary when only partial data is situated on each computing nodes. With the execution being carried by multiple devices, not only the computational time is cut drastically, but also the largest problem sizes the algorithm can handle is increased significantly. The largest problem that has been tested on the four GeForce GTX 570 GPUs with 1.2 GB memory each is $N = 2^{26}$, and the a single field evaluation takes 3.19 seconds to complete.

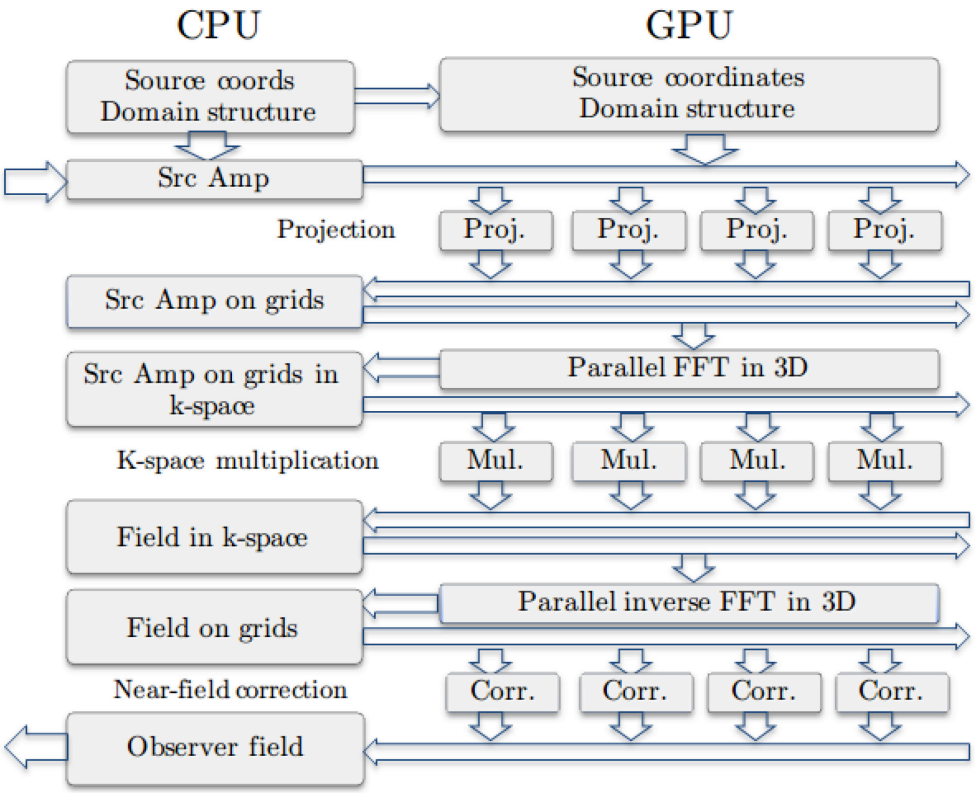


Figure 14 The flow chart of multi-GPU B-AIM. The details of parallel FFT can be found in Ref. [33, 42].

Computational times of B-AIM on four GeForce GTX 570 GPUs are shown in Figure 15, with the times on a single GTX 470 GPU card shown as a reference. It can be seen that the blue curve drops below the red one at approximately N=30K. At approximately N=1 million, the time indicated by the blue curve is 38% of that of the red curve, which corresponds to approximately 65% of parallel efficiency (here, the parallel efficiency, in term of *strong-scaling*, is defined as $\eta = nt_n / t_1$, where n is number of nodes the algorithm is running on, t_n is execution time on n GPUs and t_1 is

the execution time on the single GPU). The parallel efficiency of the near-field calculation is extremely high since they do not require communication between nodes at all. The 65% of efficiency is partly due to element rearranging procedure to transform a random scattered set of sources into a sorted one, which is just happened to be necessary at all iterations in our specific solver. The relationship of the computational times as a function of the number of GPU cards are shown in Figure 16.

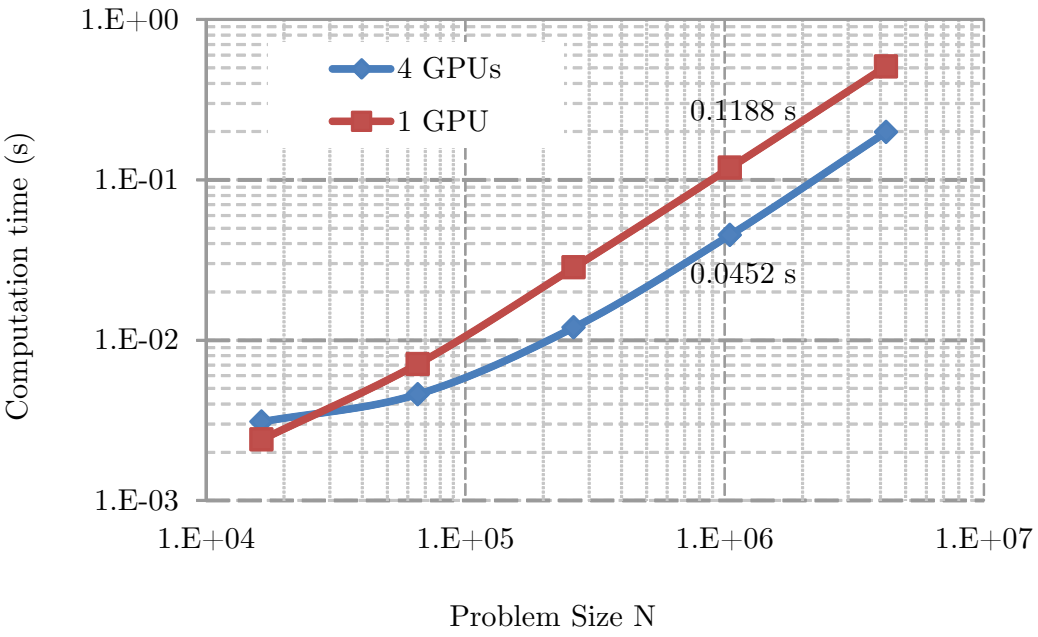


Figure 15 shows the computational time as a function of number of nodes used. The code has been tested up to 4 GPUs and we could see that the blue line deviates further from the 100% efficiency reference line at n=4. This is due to suboptimal integration between the solver and the B-AIM, which leads to unnecessary rearranging of sources at every field evaluation call.

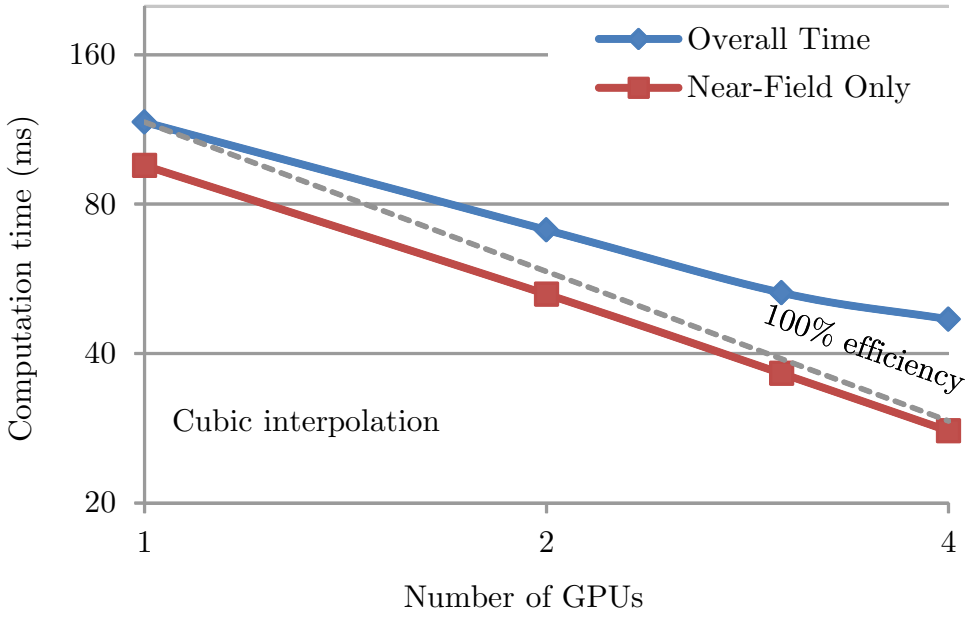


Figure 16 The parallel efficiency of the B-AIM in terms of strong scalability

4.3.5 Summary

It has been shown that the FFT-based fast methods for evaluating convolution in Eq. (2.11) and Eq. (2.19) can be accelerated by GPUs effectively. The speed-up ratios between parallel GPU code and the sequential CPU code are generally above 100. Significant changes are needed in order to adapt the algorithm with GPU's unique architectures. Specifically, we successfully reduced the memory consumption of the B-AIM algorithm while achieving such high computing throughput. Using less memory also makes the code to be able to process much larger problem and we have the algorithm handling problems with hundreds of millions sources using merely 4 Tesla GPUs that can be put into a desktop computer case.

4.4 General designing guidelines for algorithms running on GPUs

In this section, we summarize the general features of algorithms that would be favorable for GPUs and present some general strategies for designing algorithms running on GPUs. Most of the materials presented here can be found in Ref. [126] and various other documents from NVIDIA or third-party tutorials. Programmers using AMD GPUs would also be benefited from these guidelines [2]. Here we list only those that are related to common algorithms met in computational electromagnetics, computational micromagnetics and other areas that use iterative linear equation solvers.

Designing and optimizing a specific algorithm would definitely require much more efforts than just following the points listed below. Sometimes critical decisions have to be made when trade-offs are inevitable. The algorithms described in this chapter are designed, implemented and tuned from scratch by the author so to achieve encouraging performance in terms of both time and memory consumption. In general, we believe, as stated in Section 1.4, that adapting algorithms to appropriate hardware architectures at relatively low-level is necessary for extracting computing performance out of any computing systems.

4.4.1 Massive parallelism

One of the major differences between GPUs and CPUs are the scale of parallelism and threading model. Although CPUs are drastically increasing the number of cores they contain in the recent years, they are not at all comparable to the number of cores per chip on GPUs. In desktop computing systems, the highest-end CPU systems may have 8 or 10 cores per die. Meanwhile, a single high-end GPU card, like Geforce GTX 680, consists of 1536 cores. Therefore, one critical factor determining the execution efficiency of an algorithm on GPUs is the available parallelism.

Many algorithms have massive parallelization opportunities, such as the matrices multiplication example shown in the Ref. [127]. This category of algorithms is generally very suitable for GPUs and the GPU-CPU speed-up ratio could reach up to several thousands. However, there are many other algorithms that are intrinsically sequential, such as the Gaussian elimination for solving linear equation systems. These algorithms are harder to accelerate by GPUs. Therefore, GPUs are not the panacea but a specific dose of medicine under appropriate situations where hundreds of thousands of concurrent threads can be launched to solve a compute intensive problem.

In our micromagnetic solver, the number of discretized magnetization elements and other intermediate data points can be in the range from several thousands to multiple millions. It is rather common to launch a few million threads to process these elements and the GPU can generally achieve full utilization in these simulations.

4.4.2 Memory exchange between host and device

The peak global memory bandwidth of GPU is extremely high (192.2 GB/s for GTX 680 card) but the memory transfer bandwidth between GPUs and CPUs are through much slower PCI Express bus. Hence, to achieve the fastest execution efficiency, it is critical to minimize the data transfer between CPUs and GPUs.

Taking it as a general principle, the complexity of a certain algorithm or part of it should be high enough to compensate the extra time spent on moving data back and forth to and from the devices. Ideally, that part of the algorithm suitable for GPUs should contain large amount arithmetic operations that can be handled in parallel fashion. Once the programmer decides to process this part on the GPUs, one should keep the same data on the device as long as possible. This sometimes requires redesigning the algorithm to increase the temporal locality of the data accessing. For example, in our micromagnetic solver, all the components of the effective field are computed on GPUs though we gain almost nothing from lightweight kernels for calculating the Zeeman field. If possible, the entire algorithm should be run on the GPUs. For example, in our future micromagnetic solvers, an efficient ODE time integrator is being ported to GPUs so that no CPU-GPU memory transfers are required at every time integration step.

4.4.3 Floating point intensive and memory intensive applications

Memory operations and arithmetic operations are two most commonly met types of operations in many scientific computing applications. These two types of

operations follows very different execution path, from the hardware architecture perspective. GPUs indeed offer high global memory bandwidth but they possess much more arithmetic computing power. Generally speaking, memory access initiated by GPUs requires very special accessing patterns, called "coalesced access", to hide the intrinsic latency of GDDR RAM and this latency cannot be eliminated by adding more SPs. On the contrary, the arithmetic operations usually executed on registers that have no latency and can be accomplished in a few cycles. Furthermore, the floating point number computing power of GPUs can be scaled further by adding more SPs. Therefore, massively parallel processors like GPUs prefer higher arithmetic intensity operations. In our electromagnetic solver, the Green's function evaluation kernel has very high arithmetic intensity. The Helmholtz-type potentials need more than 30 floating point number operations (FLOPS) to process a single source-observer pair, with only 4 global memory loading and storing operations.

4.4.4 Using shared memory to avoid global memory access

The shared memory is a type of on-chip and extremely fast memory with several unique features. The shared memory can serve as the scratch pad for threads to share the intermediate results, relieve the register pressure and also reduce the global memory access. The shared memory has extremely low latency and can be read and written by all threads within a block at the same time, providing they are not accessing the address in the same bank. Using shared memory could significantly improve the

data readiness for parallel processors, as mentioned in the Chapter 1 when the *memory wall problem* is discussed. In our solver, shared memory is actively used due to our box-level decomposition of the domain, after which, multiple observers within the same box would have exactly the same sources to be interacted with. In this case, threads processing these observers would have same execution paths and be able to *share* same set of information through the shared memory.

4.4.5 Coalesced access to global memory

When global memory accesses are unavoidable, coalesced memory accesses would be the most important performance consideration. Coalesced memory access combines the memory loading operations by threads of the same warp into a single transaction when necessary conditions are met. For NVIDIA GPUs with compute capability of greater than 2.0, the coalesced memory access happens when a warp of threads access a single 128-byte contiguous memory piece in the global memory, meaning each thread handling a single 4-byte floating point number. Our simulator employs this access pattern wherever possible. Pre-sorting of the coordinates of sources into boxes means when a block of threads could load these coordinates into shared memory, they always follows the box boundaries and are guaranteed to load a significant chunk of data in contiguous memory locations.

4.4.6 Occupancy

Occupancy is also an important concept in understanding how hardware actually process multi-million parallel tasks. As we known from the Chapter 3, a warp of threads are always mapped to a single SM. But in order for a SM to hide the latencies of certain operations, it holds multiple warps simultaneously and switches to other execution-ready warps when the currently active warp is temporarily stalled. The number of warps that can be simultaneously situated on a SM is limited by available resources, particularly speaking the registers, the shared memory and the number of different blocks each SM can have. The way to calculate the occupancy can be found in Ref. [127] or using occupancy calculator provided with the CUDA compiler. Higher occupancy does not guarantee higher performance [167], but in general, it can provide better chance for the SM's warp scheduler to hide the latencies resulting from memory operations or register dependencies.

4.4.7 Branching and divergence

The flow control instructions in kernels significantly affect the instruction throughput if divergent branches happen within a warp. If this happens, different execution paths would be serialized and threads not participating in a certain path are disabled temporarily for their turns. If the algorithm has extremely fine parallelized tasks, it might be hard to deploy a warp of 32 threads without branching among them. However, our solvers do not have this problem as long as most of the boxes contain

more than 32 sources or observers. The boxes make the whole computational domain parallelized adequately but not too finely.

4.5 Summary

The GPU-accelerated fast methods for IE solvers can achieve extremely high performance in both the computational time and memory consumption but non-trivial efforts have to be made before "porting" any existing algorithms to GPUs. The unique architecture of GPUs requires rethinking of the fundamental data-structures and execution paths of algorithms. Various techniques has to be used to achieve high efficiency, such as ordering the data in a way that GPUs can access with minimal latencies and homogenize the computational tasks that handles by a single SM. The outcomes are encouraging. Over 100x or more speed-ups are achieved for problems of a wide range of sizes comparing with a highly efficient sequential version of the same algorithms running on a high-end CPU.

4.6 Acknowledgement

The contents in this chapter contain the materials that have been previously published in the following papers.

- S. Li, B. Livshitz, and V. Lomakin, “Fast evaluation of Helmholtz potential on graphics processing units (GPUs),” Journal of Computational Physics, vol. 229, no. 22, pp. 8463-8483, 2010.
- S. Li, R. Chang, and V. Lomakin, "Chapter 19 - Fast Electromagnetic Integral Equation Solvers on Graphics Processing Units," GPU Computing Gems Jade Edition, W. H. Wen-mei, ed., pp. 243-266, Boston: Morgan Kaufmann, 2012.
- S. Li, R. Chang, and V. Lomakin, “Fast integral equation solvers on Graphics Processing Units for Electromagnetics,” IEEE Antennas and Propagation Magazine, to appear in 2013.

5 Fast Methods for Periodic Boundary Problems

Periodic structures are widely used in microwave engineering and optics, and their efficient computational analysis is important in a multitude of applications. In many cases, a periodic structure can be considered as an infinite array of unit cells and the fields are solved for only in a single unit cell.

Integral equation (IE) solvers are often used to analyze such periodic unit cell problems [135]. When solved iteratively, IE solvers require evaluating spatial convolutions between a source distribution within the unit cell and the periodic Green's function (PGF) accounting for the presence of an infinite number of unit cells. Periodic convolutions may be also required in micromagnetic solvers when a single unit cell of an array of magnetic elements is of interest. Evaluating such convolutions is similar to that described in Chapter 4 except for the evaluation of free space Green's functions being replaced by the evaluation of PGFs. The Green's functions need to be computed $O(N^2)$ times to fill up an IE impedance matrix for directly implemented IEs, where N is the number of sources and observers. Then, the fields are found by convoluting the PGF results and the source amplitudes.

In this chapter, we present a fast periodic interpolation method (FPIM) for computing fields generated by N sources and observed at N observers in a unit cell of an infinitely periodic problem at a small ($O(N)$ or $O(N \log N)$) number of operations in the low-, high-, and mixed-frequency regimes. This method is based on separating

the PGFs into its near- and far-field components and analyzing them separately. The near-field component may be analyzed using any conventional fast methods for a finite distribution of sources and observers such as NGIM and B-AIM, presented in the previous chapter. The far-field component has slow variations thus may be solved on a sparse grid and then followed by interpolation to the source and observer locations. This would save a great number of PGF evaluations, reducing the computational cost. The FPIM is kernel independent and can handle different periodic problem types, including arrays of different dimensionality in free-space, metallic waveguides, and layered media. More importantly, FPIM allows using any available methods for evaluating the PGF, including simple Floquet summations. Practically achievable computational times can match those of conventional non-periodic fast methods that have $O(N)$ or $O(N \log N)$ complexity [100].

5.1 Problem formulation

The periodic boundary problems are slightly different from free space problems presented in Chapter 2 and 4. Using a slightly different set of terminology, we present their definition in this section before starting to describe the procedures of the FPIM.

Consider an infinite periodic array of unit cells residing in free space. Each unit cell of the array comprises N scalar point sources, labeled q_n, located at the source locations

r_n, and the same number of coinciding observers. The sources can be distributed on a surface or in a volume of a linear size D. The array can be one-, two-, or three dimensional (1D, 2D, or 3D) and it resides in an infinite homogeneous 3D space. The array's periodicity is L_x, L_y, L_z in the three possible (x, y, z) periodicity directions (with $D < \{L_x, L_y, L_z\}$). There is a linear phase shift with wavenumbers k_{x0}, k_{y0}, k_{z0} in the three dimensions, between the sources and fields in the unit cells. Compared to the wavelength λ, the domain size can be small , (i.e. $D \ll \lambda$), moderate ($D \sim \lambda$), or large (i.e. $D \gg \lambda$).

The scalar field in the prime unit cell is given by

$$u(\mathbf{r}_m) = \sum_{n=1}^{N} G_p(\mathbf{r}_m, \mathbf{r}_n) q_n \tag{5.1}$$

where G_p, the PGF describing the periodic array, is found by summing up the Green's function of a single source over all unit cells in a periodic array.

$$G_p(\mathbf{r}, 0) = \sum\nolimits_{\mathbf{i}=-\infty}^{\infty} e^{-j\mathbf{k}_t \cdot \mathbf{r_i}} G_0(\mathbf{r}, \mathbf{r_i}) \tag{5.2}$$

Here, $G_0(\mathbf{r}, \mathbf{r_i})$ is the free-space Green's function, given by

$$G_0(\mathbf{r}, \mathbf{r_i}) = G_0(\mathbf{r} - \mathbf{r_i}, 0) = \frac{e^{-jk|\mathbf{r}-\mathbf{r_i}|}}{4\pi \mid \mathbf{r} - \mathbf{r_i} \mid} \tag{5.3}$$

The vector $\mathbf{r_i}$ is the coordinate of the source in the $\mathbf{i^{th}}$ unit cell, and the wave vector $\mathbf{k}_t = (k_{x0}, k_{y0}, k_{z0})$ with generally complex components describes a linear phase shift between the source amplitudes and field in the unit cells. The index $\mathbf{i}$ is under-

stood in a general form and it can be referred to 1D, 2D, and 3D arrays. A more detailed description of this index as well as spatial and Floquet (spectral) summations for the 1D, 2D, and 3D cases are given in Appendix B. For arrays in free-space, $G_p(\mathbf{r},\mathbf{r}')$ is shift invariant, i.e. $G_p(\mathbf{r},\mathbf{r}') = G_p(\mathbf{r}-\mathbf{r}',0)$. Although the formulation presented here is for 3D free-space environments, the method is kernel independent, so a number of other environments can be handled in a similar manner.

Solving Eq. (5.1) is an essential step for iterative IE methods. As presented in Section 2.2, in mixed-potential electric field IE solvers the summation in Eq. (2.16) can be used directly. In this case, the summation is computed four times, for the scalar potential ϕ and the three components of the vector potential $\mathbf{A}$. Computationally inexpensive local corrections are used to account for the impedance matrix corresponding to overlapping basis functions. For magnetic field IEs, the task of solving Eq. (5.1) can be modified to include PGFs with a gradient. In addition, vector fields generated by vector current sources can be found with dyadic PGFs, which can be important e.g. in discrete dipole approximation approaches. The problem in Eq. (5.1) is also important in various physics and chemistry problems involving interactions between collections of particles.

The computational cost of computing $u(\mathbf{r}_m)$ at N observers due to N sources using direct evaluation of Eq. (5.1) scales as $O(N^2)$. It requires N^2 evaluations of the PGF to fill the impedance matrix. For iterative IE solvers, the conventional approach

is to build an impedance matrix with tabulated impedance values in a preprocessing stage and used merely matrix-vector multiplication for all subsequent iterations. This approach requires $O(N^2)$ memory consumption and $O(N^2)$ floating point operations for computation.

This quadratic computational cost is very high even for a small problem size N and even higher is the matrix-filling stage in which PGFs are evaluated. To allow analyzing complex periodic unit cell problems, the summation in Eq. (5.1) needs to be calculated rapidly with a reduced number of PGF evaluations. FPIM is designed to reduce this high computational cost.

5.2 Fast Periodic Interpolation Method (FPIM)

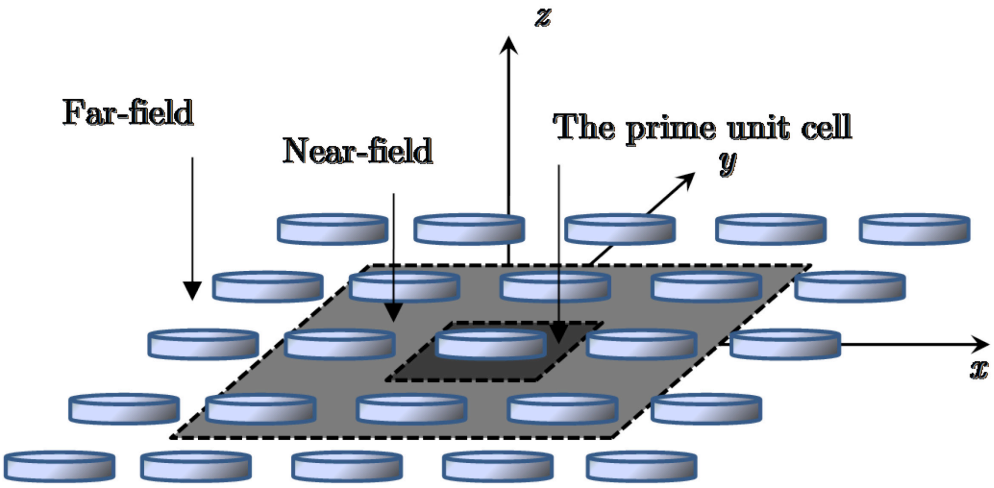


Figure 17 An example periodic structure comprising an infinite 2D periodic array in free space. 1D and 3D arrays in 3D space are also considered. The method is also applicable to many other periodic structures for which a PGF can be computed and a far-field PGF with smooth behavior can be defined.

The idea of FPIM is based on splitting the PGF and field into the near- and far-field components (see Figure 17). The PGF can be represented in the following form:

$$G_p(\mathbf{r},0) = G^p_{\text{near}}(\mathbf{r},0) + G^p_{\text{far}}(\mathbf{r},0) \tag{5.4}$$

where $G^p_{\text{near}}(\mathbf{r},0)$ and $G^p_{\text{far}}(\mathbf{r},0)$ are the near- and far-field components of the PGF given by

$$\begin{aligned} G^p_{\text{near}}(\mathbf{r},0) &= \sum\nolimits_{\mathbf{i}=-i_d}^{i_d} e^{-j\mathbf{k}_t\cdot\mathbf{r}_\mathbf{i}} G_0(\mathbf{r},\mathbf{r}_\mathbf{i}), \\ G^p_{\text{far}}(\mathbf{r},0) &= G^p(\mathbf{r},0) - G^p_{\text{near}}(\mathbf{r},0) \end{aligned} \tag{5.5}$$

Here, $G^p_{\text{near}}(\mathbf{r},0)$ is given as a summation of the simple Green's functions $G_0(\mathbf{r},\mathbf{r}_\mathbf{i})$ around the prime unit cell, which is similar to the expression in Eq. (5.2) but with a finite summation over a limited range determined by i_d ($i_d = 1$ can be chosen

for most cases). The far-field component of PGF, $G^p_{\text{far}}(\mathbf{r},0)$, is given as the sum of contributions from the infinite number of remaining unit cells.

Based on the PGF decomposition, the field is also decomposed as

$$u(\mathbf{r}_m) = u_{\text{near}}(\mathbf{r}_m) + u_{\text{far}}(\mathbf{r}_m) \tag{5.6}$$

in terms of its near- and far-field components

$$u_{\text{far}}(\mathbf{r}_m) = \sum_{\substack{n=1,\\ n\neq m}}^{N} G^p_{\text{far}}(\mathbf{r}_m,\mathbf{r}_n)q_n \tag{5.7}$$

$$u_{\text{near}}(\mathbf{r}_m) = \sum_{\substack{n=1,\\ n\neq m}}^{N} G^p_{\text{near}}(\mathbf{r}_m,\mathbf{r}_n)q_n \tag{5.8}$$

The task of evaluating the near-field field u_{near} is mostly identical to that of evaluating the field in non-periodic structures using B-AIM or NGIM; some minor modifications that may be required for the summation for u_{near} as in Eq. (5.8) are outlined in Section 5.3. The rapid evaluation of the far-field u_{far} as in Eq. (5.7) is a more complicated task for general problems, which will be addressed in Section 5.4.

5.3 Evaluation of the near-field periodic field in FPIM

To evaluate the near-field u_{near} as in Eq. (5.6) with the near-field PGF component in Eq. (5.5), u_{near} can be recast in an alternative form as

$$u_{\text{near}}(\mathbf{r}_m) = \sum_{\substack{n=1 \\ n\neq m}}^{N} \sum_{\mathbf{i}=-i_d}^{i_d} G_0(\mathbf{r}_m, \mathbf{r}_n) q_n e^{-\jmath \mathbf{k}_t \cdot \mathbf{r}_\mathbf{i}} = \sum_{\substack{n=1 \\ n\neq m}}^{N_d} G_0(\mathbf{r}_m, \mathbf{r}_n) q_n^t \tag{5.9}$$

Here, the first summation is for N sources in $(2i_d+1)^\xi$ units cells including the unit cell of interest and its surrounding unit cells, where ξ is the dimensionality of the array ($\xi = 1,2,3$ for 1D, 2D, 3D arrays). The second summation in Eq. (5.9) is over an extended index running over all $N_d = (2i_d+1)^\xi N$ sources in the $(2i_d+1)^\xi$ (prime and adjacent) unit cells with the sources $q_n^t = q_n \exp(-\jmath \mathbf{k}_t \cdot \mathbf{r}_\mathbf{i})$ defined to include the phase shift between the unit cells.

The evaluation of the second summation in Eq. (5.9) for free-space problems is a standard task that can be handled using any available fast methods, such ones presented in this thesis. For instance, if we use NGIM described in Section 4.2, the computational domain (comprising $(2i_d+1)^\xi$ cells) for computing u_{near} will be divided into a multilevel hierarchy of levels of boxes. Interactions between "well separated" boxes are accounted via spatial interpolations, whereas the interactions between the neighboring boxes containing a small number of source-observer points are accounted for via direct superposition. This results in a computational cost of $O(N)$ or $O(N \log N)$, depending on the frequency domains of the calculation. Other methods are extensively documented in literatures (see e.g. [35]) and therefore no further details are provided.

5.4 Evaluation of the far-field periodic field in FPIM

The far-field component of the PGF G^p_{far} and the field u_{far} observed in the prime unit cell are generated by sources residing outside of the i_d-th unit cells. The far-field PGF and the field itself have bounded spatial variations within the prime periodic unit cell. This means that G^p_{far} and u_{far} at the observers can be calculated by interpolation from a set of relatively sparse samples, called observer grid. Due to reciprocity, the far-field PGF G^p_{far} has bounded variations with respect to the source coordinates, which can also be computed at the source locations by interpolating from a sparse source grid. The sampling rates for the grids are determined by the Nyquist criterion, which requires the grid step sizes Δx, Δy, and Δz to be less than $\lambda/2$ (in Cartesian coordinates). To account for different electrical sizes of the cells, the step sizes are chosen as $\min\{\lambda/2, D\}/\Omega$, where $\Omega \geq 1$ is an oversampling ratio. Based on this understanding, the proposed FPIM is accomplished in three stages.

5.4.1 Stage 1: Evaluating G^p_{far} at source and observer grids

In this stage, the far-field PGF G^p_{far} is evaluated at in the grid points of the prime periodic unit cell (Figure 17). Let the grid $\{\mathbf{r}^s_n = (x^s_n, y^s_n, z^s_n)\}_{n=1}^{N_g}$ be the source grid and $\{\mathbf{r}^o_m = (x^o_m, y^o_m, z^o_m)\}_{m=1}^{N_g}$ be the observation grid.

In our FPIM implementation, we choose uniform Cartesian lattices (Figure 18) because we will later use FFT to accelerate the calculation, as being done in B-AIM

discussed in Section 4.3. The number of grid points for both grids in the x, y, and z directions is N_g^y, N_g^x, and N_g^y, respectively, and the total number of possible grid points is $N_g = N_g^x N_g^y N_g^z$. The source and observation grid coordinates are staggered by about half of the interval in all dimensions, so the relationship between the coordinates is represented as $\mathbf{r}_n^o = \mathbf{r}_n^s + (\mathbf{x}\Delta x + \mathbf{y}\Delta y + \mathbf{z}\Delta z)/2$. This choice of grids assures a rapid convergence of the PGF from the source grid points to the observation grid points via simple Floquet summations [135], thus allowing using the latter with FPIM efficiently.

For free-space problems with translation invariant PGFs, there are only $O(N_g)$ different values of $\mathbf{r}_{m'}^o - \mathbf{r}_{n'}^s$, which leads to $O(N_g)$ evaluations of PGF (Figure 18). Moreover, G_{far}^p need to be evaluated only at the number $N_g' \leq N_g$ of grid points around the sources and observers (gray samples in Figure 18). Here, due to the Nyquist criterion, the number of relevant grid points is $N_g' = O\left(\max\{2D/\lambda, 1\}^2\right)$ or $N_g' = O\left(\max\{2D/\lambda, 1\}^3\right)$ for surface or volume charge distributions, respectively.

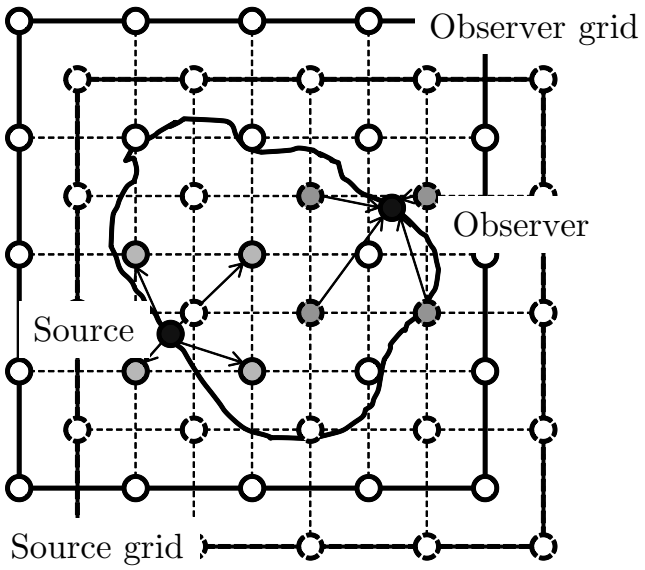


Figure 18 The schematic illustration of the source and observer grids. The grids are chosen as shifted Cartesian lattices to allow for using simple Floquet summations for PGFs. The choice of grids, however, is flexible and other grid types can be used. The grey dots represent the grid points around the computational domain for which PGFs need to be computed.

The PGF between grid points, $G^p_{\text{far}}(\mathbf{r}^o_m, \mathbf{r}^s_n)$, can be evaluated using any available methods, including various acceleration techniques [13, 27, 83, 120, 162, 164]. However, even simple spectral Floquet mode expansions can be very efficient for field calculations (see Floquet expansions for arrays in free space in Appendix B). This is possible due to both the grids and the interpolation procedure can be set up to avoid specific source-observer arrangements for which the Floquet series convergence is slow. Additional details are given in Section 5.5.

The computational cost of this stage scales as $c_{PGF}N'_g$, where c_{PGF} is the cost of evaluating a single PGF G^p_{far}. The evaluation of G^p_{far} at the grids depends only on

the unit cell parameters and wavelength, and not on the source distribution. In the framework of IE iterative methods, the far-field PGF on at the grids is therefore required to be tabulated only once.

5.4.2 Stage 2: Evaluating u_{far} at the observation grid

In this stage, the field $u_{\text{far}}(\mathbf{r}_m^o)$ is evaluated at the observation grid points. Two approaches can be followed.

Approach 1:

The calculation of the field $u_{\text{far}}(\mathbf{r}_m^o)$ can be accomplished in two steps. First, the far-field component of the PGF $G_{\text{far}}^p(\mathbf{r}_m^o, \mathbf{r}_n)$ from the actual source locations $\mathbf{r}_n$ to the observation grid points $\mathbf{r}_m^o$ is calculated by locally interpolating from the source grid points $\mathbf{r}_n^s$

$$G_{\text{far}}^p(\mathbf{r}_m^o, \mathbf{r}_n) = \sum_{n'=1}^{N_q} w^s(\mathbf{r}_n, \mathbf{r}_{n'}^s) G_{\text{far}}^p(\mathbf{r}_m^o, \mathbf{r}_{n'}^s) \tag{5.10}$$

where $w^s(\mathbf{r}_n, \mathbf{r}_{n'}^s)$ are interpolation coefficients and $N_q \leq N_g$ is the number of grid points used for interpolation. For example, $N_q = (q+1)^3$ for Lagrange interpolation of order q. Other interpolation approaches can be also used, including Chebychev or simplex interpolations. Once $G_{\text{far}}^p(\mathbf{r}_m^o, \mathbf{r}_n)$ is found, the far-field field $u_{\text{far}}(\mathbf{r}_m^o)$ at the observation grid is found via the summation similar to that in Eq. (5.1)

$$u_{\text{far}}(\mathbf{r}_m^o) = \sum_{n=1}^{N} G_{\text{far}}^p(\mathbf{r}_m^o, \mathbf{r}_n) q_n \tag{5.11}$$

The resulting computational cost is $\left(c_1 N_q + c_2 N_q N_g' + c_3 N_g'\right) N$ with constants $c_{1,2,3}$, which includes the cost of $O(N_q N)$ for the interpolation coefficients $w^s(\mathbf{r}_n, \mathbf{r}_{n'}^s)$ in Eq. (5.10) (to be executed only once for an iterative IE solver), $O(N_q N_g' N)$ cost of evaluating the far-field PGF $G_{\text{far}}^p(\mathbf{r}_m^o, \mathbf{r}_n)$ in Eq. at required observer grid points from all source points (to be executed only once), and $O(N_g' N)$ cost of calculating $u_{\text{far}}(\mathbf{r}_m^o)$ in Eq. (5.11) (to be executed in each iteration for an iterative IE solver). Conceptually, this procedure is similar to the evaluation of the local fields in the framework of the NGIM in the Section 4.2. The computation cost in the above procedure is of $O(N_g' N)$, which can be much smaller than the cost of the direct approach but potentially still may be significant. This cost can be further reduced as described next.

Approach 2:

An alternative procedure for calculating the field $u_{\text{far}}(\mathbf{r}_m^o)$ is by combining Eq. (5.10) and Eq. (5.11) and extending the n' summation to all possible N_g source grid points for the chosen uniform grid. The result is the following representation

$$\begin{aligned} u_{\text{far}}(\mathbf{r}_m^o) &= \sum_{n'=1}^{N_g} G_{\text{far}}^p(\mathbf{r}_m^o, \mathbf{r}_{n'}^s) Q_{n'}, \\ Q_{n'} &= \sum\nolimits_{n=1}^{N} q_n w^s(\mathbf{r}_n, \mathbf{r}_{n'}^s) \end{aligned} \tag{5.12}$$

Here, each $Q_{n'}$ has the meaning of the surrounding N_q sources projected onto a sparse grid point at $\mathbf{r}_{n'}^s$ via the second expression in Eq. (5.12). This projection follows the same logic and operation as in B-AIM, described in Section 4.3. These

projected sources are obtained by superimposing the original sources weighted with the interpolation coefficients $w^s(\mathbf{r}_n, \mathbf{r}^s_{n'})$ to N'_g grid points (with N_q grid points for each source $\mathbf{r}_n$) and zero padding to the remaining $N_g - N'_g$ grid points. Using uniform grids and defining the summation in Eq. (5.12) for all N_g grid points, this summation can be calculated via FFT, which significantly reduces the computational cost. The resulting cost of computing $u_{\text{far}}(\mathbf{r}^o_m)$ via Eq. (5.12) is $c_1 N_q N + c_2 N_g \log N_g$ (for every iteration of an IE solver), including the cost of $O(N_q N)$ for computing $Q_{n'}$ and $O(N_g \log N_g)$ for computing $u_{\text{far}}(\mathbf{r}^o_m)$. So, conceptually, the whole approach in Eq. (5.12) is similar to the B-AIM, but it allows using much sparser grids whose density can be independent of the density of the source distribution. Furthermore, the interpolation coefficients for projections are available in closed form, which reduces the computational cost and memory requirements.

Approach 1 is more efficient for very small N. However, for most practical cases Approach 2 is more efficient and it is therefore implemented in the rest of the paper.

5.4.3 Stage 3: Evaluating u_{far} at the actual observers

The far-field field $u_{\text{far}}(\mathbf{r}_m)$ at all N observers $\mathbf{r}_m$ is obtained by interpolating from the field $u_{\text{far}}(\mathbf{r}^o_m)$ at the observation grid:

$$u_{\text{far}}(\mathbf{r}_m) = \sum_{m'=1}^{N_q} w^o(\mathbf{r}_m, \mathbf{r}^o_{m'}) u_{\text{far}}(\mathbf{r}^o_{m'}) \tag{5.13}$$

where $w^o(\mathbf{r}_m, \mathbf{r}^o_{m'})$ are interpolations coefficients similar to those for the source grid. The computational cost of this stage scales as $O\left(N_q N\right)$.

The choice of uniform Cartesian (source and observation) grids (see Figure 18) allows using FFT to accelerate computations in both low- and high-frequency regimes. The choice of uniform grids still allows for a seamless handling of non-uniform source-observer distributions (e.g. $q_n, \mathbf{r}_n$ in Eq. (5.12) and $\mathbf{r}_m$ in (5.13) can have any distribution).

5.5 Computational complexity

The computational complexity of FPIM can be split into the cost of the preprocessing, to be evaluated only once for an iterative solution of fields, and the cost of the field evaluation, to be done at every iteration. The preprocessing cost is given by $c_{PGF} N_g + c_{\text{interp}} N$, where c_{interp} is the cost of evaluating the interpolation coefficients for a single source or observer. The cost of c_{interp} scales as $O(1)$ (assuming $q = O(1)$), whereas the cost of c_{PGF} scales differently in the low- and high-frequency regimes. The cost of field evaluation is dominated by the cost of Stage 2, which is given by $c_1 N_q N + c_2 N_g \log N_g$, where $c_{1,2}$ are constants of $O(1)$.

Below we detail the computational cost of FPIM for the low- and moderate-frequency regime ($D < \lambda$ or $D \sim \lambda$), high-frequency regime ($D \gg \lambda$), and mixed-

frequency regime. The low- and moderate-frequency regimes are met in the majority of applications of periodic structures related to radiation, scattering, and propagation. High- and mixed-frequency regimes may be important for certain antenna array problems and random problems, e.g. where periodic continuations can be used instead of truncating the domain of interest.

5.5.1 Low- and moderate-frequency regime

For the low-frequency regime the PGF and field vary slowly and the grids can be very sparse with $N_q, N_g', N_g = O(1)$, resulting in the computation cost of $O(N)$. The total number of grid points N_g and the number of required grid points N_g' can be chosen to be the same.

The far-field Green's function $G^p_{\text{far}}(\mathbf{r}^o_m, \mathbf{r}^s_n)$ at the sparse grids can be evaluated using any available methods including simple spectral Floquet mode expansions (see Appendix B) or any accelerated techniques. The Floquet mode expansions can be efficient because the grids are chosen such that all the source-observer pair has sufficient separation, under which Floquet expansions become rapidly convergent. For example, the choice of the shifted Cartesian grids as in Fig. 2 assures that the minimal transverse separation between the source and observation grid points is $\min\{\Delta x, \Delta y, \Delta z\}\big/2$. As a result, the far-field PGF $G^p_{\text{far}}(\mathbf{r}^o_m, \mathbf{r}^s_n)$ can be evaluated efficiently with $O(L_x\big/\Delta x)$ terms in the Floquet summation for the case of 1D arrays and $O\Big((L_x L_y)\big/(\Delta x \Delta y)\Big)$ terms for 2D and 3D arrays. For the low-frequency regime, N_g^x,

N_g^y, N_g^z are of $O(1)$, resulting in the minimal source-observation grid points separation of $O(L_x)$. This separation means the cost of evaluating the PGF using simple Floquet expansions will be a constant.

Summarizing, the preprocessing cost in the low-frequency regime scales as $O(1)$ and the computation cost scales as $O(N)$. Because of the simplicity of operations, the total absolute cost is very low as shown in Section 5.6. This cost remains unchanged for any source-observer distribution, including volumetric and surface uniform and non-uniform distributions.

5.5.2 High-frequency regime

The high-frequency regime is typically defined such that the computational domain size is electrically large, i.e. $D, L_{x,y,z} \gg \lambda$, and the source distribution is smooth so that the number of source-observer points is $N = O((D/\lambda)^2)$ or $N = O((D/\lambda)^3)$ for surface and volumetric distributions respectively. For such high-frequency problems, $\Delta x, \Delta y, \Delta z < \lambda/2$ according to the Nyquist criterion and the number of the grid points at which the PGF is computed is $N_g' = O(N)$. The preprocessing cost is dominated by the PGF tabulation, which scales as $O(c_{PGF}N)$. The evaluation of PGF for the high-frequency case may be time consuming. Using the Floquet summation approach in Appendix B, it can be shown that $c_{PGF} = O(D/\lambda)$ for 1D arrays and $c_{PGF} = O((D/\lambda)^2)$ for 2D and 3D arrays. Taking into account the relation between N, N_g', and D/λ, this preprocessing cost is significantly higher than

that of the low-frequency regime. Currently there are no available methods that can reduce this cost. (Other available approaches either cannot be used for the high-frequency regime or lead to the same or even higher cost.)

The computation cost of the high-frequency regime is dominated by the first summation in Eq. (5.12), which can be evaluated via FFT. The use of FFT requires extending the grids to cover the entire volume around the structure, which results in different computational costs for volumetric and general surface source distributions. The resulting computation cost scales as $O(N \log N)$ for volumetric and quasi-planar surface problems and it scales as $O(N^{3/2} \log N)$ for general surface problems.

As clear from the above discussion, the preprocessing and computation times of FPIM in the high-frequency regime are higher than those in the low-frequency regime. However, these times are still much smaller than those required for direct evaluation of Eq. (5.1).

5.5.3 Mixed-frequency regime

In the mixed-frequency regime, the source distribution is electrically large, i.e. $D, L_{x,y,z} \gg \lambda$, but the distribution also is dense in at least some parts of the domain. The FPIM procedure remains unchanged. The grid density is chosen as in the high-frequency regime such that $\Delta x, \Delta y, \Delta z < \lambda/2$, resulting in $N_g' = O((D/\lambda)^2)$ or $N_g' = O((D/\lambda)^3)$ grid points at which the PGF is computed for surface or volumetric

source distributions, respectively. The number of grid points for a single interpolation is $N_q = O(1)$.

The preprocessing and computation costs can be obtained based on the costs derived for the low- and high-frequency regimes. Specifically, the preprocessing cost scales as $O(N\,D/\lambda)$ for 1D arrays and $O(N\,(D/\lambda)^2)$ for 2D and 3D arrays. The computation cost scales as $O((D/\lambda)^3 \log(D/\lambda) + N_q N)$ for all array types. These costs are much lower than those for the direct field evaluation, and are significantly lower than the cost of B-AIM. In particular, FPIM makes the low-frequency part of a mixed-frequency problem much faster.

5.6 Results

5.6.1 Computational times in various frequency regimes

FPIM was implemented in FORTRAN, compiled with Intel Fortran v11.1 at –O3 optimization, and run on a desktop with Intel i7-920 2.66GHz CPU on a single core. The source and observer grids are shifted Cartesian grids as shown in Fig. 2. The interpolation was chosen to be Lagrange type with the order ranging from linear to sixth. The parameter i_d in Eq. (5.5) was chosen as $i_d = 1$.

We start by considering a 1D periodic array in free 3D space in the low- and moderate-frequency regime. The periodicity of the array is $L_x = \lambda/2$ and the phase

shift wave number is $k_{x0} = (1.2 - 0.01j)k$. The sources are distributed randomly in a cube of edge length $D = L_x$. The results in Figure 19 (as well as Figure 20 and Figure 21) are similar for various arrays and source distributions, including surface distributions and other highly non-uniform distributions, as further demonstrated in Table 15.

For the 1D array, we have implemented two methods to compute the PGF. The first method uses the simple Floquet mode expansion given in Appendix B (the first equation in Eq. (8.3)) while the other uses the fast hybrid spatial-spectral representation introduced in [164]. The Floquet summation method is efficient only for sufficiently large source-observer separations transverse to the array axis. For small transverse source-observer separations it converges very slowly, and it diverges on the array's axis for 1D arrays. As a result, the Floquet expansion method is unacceptable for directly computing the summation in Eq. (5.1). However, it can be used efficiently in FPIM as the minimal off-axis separation between the grid points is fixed. The alternative PGF representation of [164] is applicable to virtually any source-observer separations near the array axis with very robust performance. Both approaches for obtaining the PGFs are used and compared with each other. It is demonstrated that FPIM is very efficient even with the simple Floquet summation, with computational times comparable to those of finite (non-periodic) electromagnetic N-body problems.

Figure 19 shows the time for computing the far field component u_{far} using FPIM and the direct superposition of Eq. (5.1) in the low-frequency regime. The times

are shown separately for the pre-processing and the computational stages. The pre-processing stage is to be executed only once in an IE code, consisting of tabulating the PGF at the sparse grids (Stage 1 in Section 5.4.1) and calculating the interpolation coefficients (Stage 2 in Section 5.4.2). The computation stage consists of computing the field at the sparse grid via *Approach 2* of Stage 2 as in Section 5.4.2 and calculating the field at the observers in Stage 3 as in Section 5.4.3. This stage is to be executed at every iteration step for an IE solver. The number of source and observer grid points was $N_o = N_s = 512$ and cubic (Lagrange) interpolation was used resulting in an RMS error at the level of 10^{-3}.

The computation time of the direct summation (dashed curve) and fast FPIM (solid curve) scale as $O(N^2)$ and $O(N)$, respectively. The absolute time of the FPIM is significantly smaller than that of the direct summation for all considered problem sizes. For example, for $N = 64$, the speed-up is 87x and for $N = 2$ million the computational time of the FPIM is 13 seconds and the speed-up is 1.9e7 (with the time of the direct approach extrapolated to this large N). Practically, the obtained computational time for the far-field is comparable to the time of evaluating the field for a conventional non-periodic problem of the same size.

The preprocessing times shown in Figure 19 for small N saturate at the lower end due to constant time of tabulating the PGF on the sparse grids (Stage 1 in Sec. 5.4.1). As expected, this saturation time is smaller for the faster alternative PGF

calculation approach of [164] (dotted curve). However, even for the conventional and simple Floquet summation approach (dash-dotted curve), this saturation time is very small. For larger N the times are nearly the same for both approaches since the preprocessing stage is dominated by the construction of the interpolation coefficients of Stage 2 in Sec. 3.C, which is unrelated to the PGF evaluations. The preprocessing time is smaller than the direct time at $N = 80$ for the PGF computed as in [164] and $N = 200$ for the PGF computed via Eq. (8.3) . The computational time is much smaller for larger N. Moreover, there is no significant benefits of using faster methods for the PGF when N is greater than only a few hundreds. Therefore, FPIM is very efficient, even with simple Floquet mode summations for evaluating PGFs.

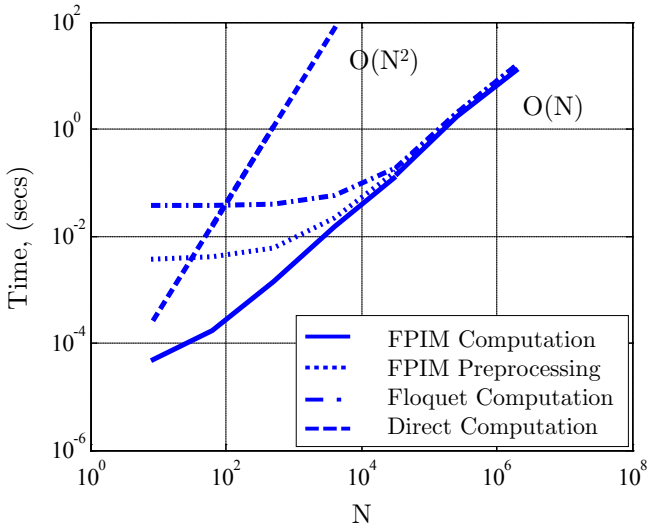


Figure 19 The preprocessing and computational times vs. N in the low-frequency regime for a linear (1D) array with $k_{x0} = \left(1.2 - 0.01j\right)k$. The sources are distributed randomly in a cube of linear size $D = L_x = \lambda / 2$. The times for two different methods for the PGF are shown, including the Floquet summation in Eq. (8.3) and the alternative (faster) approach of [164] . The number of grid points is $N_g = 8^3$ and the cubic interpolation is used. The RMS error is 1×10^{-3}.

Figure 20 shows the preprocessing and computation time for a linear array in the high-frequency regime. As in Figure 3, the sources are distributed in a cube of linear size $D = L_x$. However, the number of sources N scales with the electrical size of the computational domain as $N = (16\, D/\lambda)^3$, i.e. there are 16 sources per wavelength. For D in the range from 0.5λ to 8λ the number of sources N is in the range from 512 to 2 million.

To account for the variation of high-frequency fields, the number of the grid points N_g also scales with the domain size, as $N_g = (12\,D/\lambda)^3$. The increased number of the grid points leads to a noticeable increase of the preprocessing time as compared to that in the low-frequency regime in Figure 19. While being slower in the high-frequency regime as compared to the low-frequency regime, FPIM is still significantly faster than the direct method for all practical problem sizes. For example, the speed-ups in Figure 20 are in the range from 24 for $N = 4096$ to 2000 for $N = 2$ million.

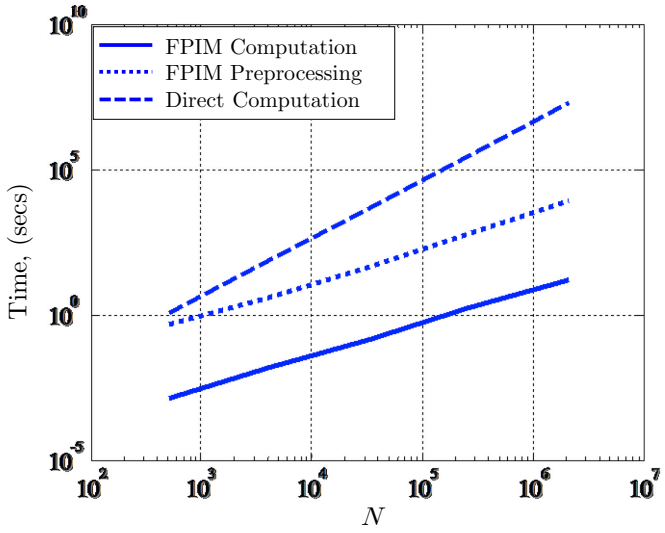


Figure 20 The preprocessing and computational times vs. N in the high-frequency regime for a linear array with $k_{x0} = (1.2 - 0.01j)k$. The size of the computational domain varies from $\lambda/2$ to 8λ and $N = (16\,D/\lambda)^3$. The number of grid points is chosen as $N_g = (12\,D/\lambda)^3$. The PGF is computed via the Floquet expansion in Eq. The cubic interpolation is used. The RMS error is 3×10^{-3}.

Figure 21 shows the computation time in the mixed-frequency regime for a 1D array on the x axis with a quasi-planar computational domain. As described in Section 5.5.3, in the mixed-frequency regime the size of computational domain is large but at least a part of the domain contains a dense source distribution. In our simulations, the computational domain size is $8\lambda \times 8\lambda \times 0.1\lambda$ with $L_x = 8\lambda$ and the number of grid points is $N_g = 96 \times 96 \times 4$. Lagrange cubic interpolation is used in all dimensions. The preprocessing time is almost a constant since the number of grid points is relatively large and is determined by the large electrical size of the computational domain. The computation times are nearly identical to those of the low-frequency regime in Figure 19.

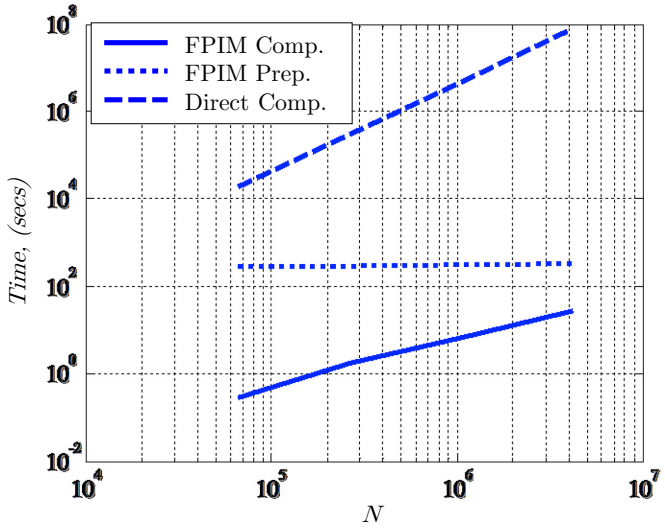


Figure 21 The preprocessing and computational times vs. N in the mixed-frequency regime for a linear array. The array is oriented along the x axis with $k_{x0} = (1.2 - 0.01j)k$ and a quasi-planar source distribution. The size of the computational domain is $8\lambda \times 8\lambda \times 0.1\lambda$. The sources are arranged in four identical horizontal layers in the $x - y$ plane. In each layer, the source distribution is a combination of two set of sources, including a number of $128 \times 128 = 16384$ sources, which represent the high-frequency regime with the uniform source density determined by the source-to-source separation of $\lambda/16$, and a number of $(N - 65536)/4$ sources, which represent the low-frequency regime with a density increasing as $1/(xy)$ towards the origin. The number of grid points is $N_g = 4 \times (12\,D/\lambda)^2$ and the cubic interpolation is used. The simple Floquet expansion in Eq. (8.3) is used for the PGF. The RMS error is 3×10^{-3}.

The results in Figure 19, Figure 20 and Figure 21 show that the FPIM is efficient for problems in a broad range of frequencies.

5.6.2 Computational times for various kernels

Table 15 shows the tabulation time for different kernels, in the low- and high-frequency regimes. In the low-frequency regime (second column), the number of the grid points is $N_g = 64$ and the array parameters are chosen as $L_x = L_y = L_z$ and

$k_{x0} = k_{y0} = k_{z0} = 0$. The sources are distributed randomly in a cube of size $D = L_x = \lambda/2$. The 1D PGF is computed via the alternative approach in [164] and via the Floquet expansion in Appendix B (the first equation in Eq. (8.3) The 2D and 3D PGFs are computed via the Floquet expansion in Appendix B (the second and third equations in Eq. (8.3) .All results are given for an RMS error at the level of 10^{-3}. In this low-frequency regime the tabulation time is independent of N since N_g is a constant of $O(1)$. The obtained times are comparable for all considered PGF cases, and are lower than the time of direct evaluation for the 1D case via the fast alternative method of [164] for all practical problem sizes (with $N \geq 100$). Any direct methods for the 2D and 3D arrays, including those using PGF acceleration techniques such as the Ewald approach, will be even slower. FPIM is therefore efficient for all demonstrated kernel types. For large N, the tabulation time is small compared to the computation time, and the total cost is essentially the same for all kernels.

In the high-frequency regime (last column in Table 15), the computational domain size is a cube of size $D = L_x = 4\lambda$ and the number of sources is $N = 262{,}144$, i.e. there 16 sources per wavelength as in Figure 19. The grid density is also as in Figure 19. Here, the PGF tabulation time depends on N as shown in Figure 19; the high-frequency results in Table 15 are therefore shown for the specific N. The 1D, 2D, and 3D PGF are computed via the Floquet expansions in Appendix B, which in the framework of the FPIM in this case are as efficient as (or even more efficient than)

other methods for evaluating the PGFs. It is found that the evaluation of the 2D and 3D PGFs in this high-frequency regime is slower than the evaluation of the 1D PGF. The tabulation time for the high-frequency regime is significantly larger for all shown kernel types as compared to the low-frequency regime. This is because of the increased grid density and increased cost of the PGF evaluation. However, the obtained computational times for all kernels are much smaller than the times of the direct field evaluation.

Table 15 The PGF tabulation time of different computational kernels.

Kernel	Time (High-frequency)	Time (Low-frequency)
1D alternative [164]	N/A	4.6e-3
1D Floquet	4.2e2	3.6e-2
2D Floquet	1.4e3	1.2e-2
3D Floquet	8.9e3	7.9e-2

Results are shown for both the low-frequency regime (second column) and high-frequency regime (last column) for the RMS error below 10^{-3}. For the low-frequency regime, $D = \lambda/2$, $N_g = 64$; the tabulation time is a constant for a given accuracy since the grid density is fixed for any N. For the high-frequency regime, $D = 4\lambda$, $N_g = 110,592$, $N = 262,144$; the tabulation time increases with N.

5.6.3 Computational accuracy

Figure 22 shows the RMS error and computational time versus the grid density for different interpolation orders q and number of grid points N_g for $D = \lambda/2$. From Figure 22 (a), it is evident that the error can be reduced by increasing the grid density

and/or interpolation order. Such an increase leads to an increase of the computation or/and preprocessing times. Figure 22 (b) depicts the computational and preprocessing time for the 1D array with the PGF computed via the approach of [164] for $N = 524,288$. Increasing q increases both the preprocessing and computation time, while increasing the grid density increases only the preprocessing time. From our numerical experiments, we found that using cubic interpolation is often most beneficial in terms of a trade-off between accuracy and computational time, hence the results in are given with the cubic interpolation. In particular, cubic interpolation with an oversampling ratio Ω in the range 6 to 8 leads to the RMS error at the level of 10^{-3} for all the cases in the paper (and for many other practical cases).

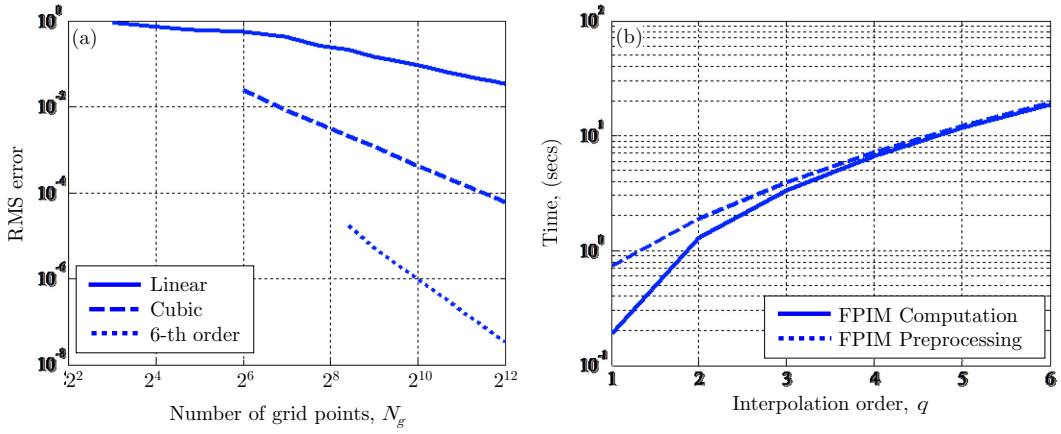


Figure 22 NGIM performance vs. the interpolation order and number of grid points for a linear (1D) array with $k_{x0} = (1.2 - 0.01j)k$. The sources are distributed in a cube of liner size $D = L_x = \lambda/2$. (a) Error vs. the number of grid points for different interpolation orders; (b) The preprocessing and computation time vs. the interpolation order for $N = 524,288$. The PGF is computed using the approach of [164].

5.7 Discussions on extended applications of FPIM

The results in Section 5.6 are presented for 1D, 2D, and 3D rectangular arrays in 3D free-space. However, due to the separation of PGF evaluation and interpolations, the FPIM can handle several other array types, as long as smoothly behaved far-fields can be obtained and the PGF can be calculated. Several possible problem types and details of the FPIM for them are listed.

(a) Periodic 1D and 2D arrays in 2D free-space

For free-space problems in two dimensions, the elementary sources are line sources and the free-space (non-periodic) Green's function is $G_0(\mathbf{r},\mathbf{r_i}) = (j/4)$ $H_0^{(2)}(k \mid \mathbf{r}-\mathbf{r_i} \mid)$, where $H_0^{(2)}$ is the Hankel function of second kind. FPIM will be unchanged as described in Section 5.4, with a difference being that the PGF is defined for 2D space and the interpolations are done in 2D. The PGF can be calculated either using Floquet expansions or using alternative spectral and spatial representations [164].

(b) Periodic 1D and 2D arrays in metal wall waveguides

Another type of problem that can be handled involves 1D or 2D arrays in a parallel metal plate waveguide as well as 1D arrays along the axis of a metal wall rectangular waveguide. Consider a parallel plate waveguide with metallic walls at $y = 0$ and $y = L_y/2$. Inside this waveguide consider a 1D array of periodicity L_x along the x direction with a phase shift wavenumber k_{x0} or a 2D array of periodicities

L_x, L_z along the x, z directions with phase shift wavenumbers k_{x0}, k_{z0}. Assume that the boundary conditions for the field u and the Green's function G_{pw} at the walls are of Dirichlet type, i.e. $u\big|_{y=0,L_y/2} = G_{pw}\big|_{y=0,L_y/2} = 0$. Using the image representation, the Green's function for the waveguide can be given in terms of a superposition of PGFs for two arrays

$$G_{pw}\left(\mathbf{r},\mathbf{r}'\right) = G_p\left(\mathbf{r},\mathbf{r}'\right) - G_p\left(\mathbf{r},\mathbf{r}' - 2y'\mathbf{y}\right) \tag{5.14}$$

For the 1D array in the parallel plate waveguide, G_p is the PGF for the 2D array in free space as defined in Eq. (5.14) with the phase shift wavenumbers $k_{x0}, k_{y0} = 0$. For the 2D array in the parallel plate waveguide, G_p is the PGF for the 3D array in free space with the phase shift wavenumbers $k_{x0}, k_{y0} = 0, k_{z0}$.

The representation in Eq. (5.14) can be further extended to the case of a linear array along the x direction that resides in a rectangular waveguide of a cross-sectional size of $L_y/2 \times L_z/2$ with Dirichlet boundary conditions. For this case, the Green's function of the waveguide is given as

$$\begin{aligned} G_{pw}\left(\mathbf{r},\mathbf{r}'\right) = {} & G_p\left(\mathbf{r},\mathbf{r}'\right) - G_p\left(\mathbf{r},\mathbf{r}' - 2y'\mathbf{y}\right) \\ & - G_p\left(\mathbf{r},\mathbf{r}' - 2z'\mathbf{z}\right) + G_p\left(\mathbf{r},\mathbf{r}' - 2\left(y'\mathbf{y} + z'\mathbf{z}\right)\right) \end{aligned} \tag{5.15}$$

where G_p is the PGF for the 3D array in free space with the phase shift wavenumbers $k_{x0}, k_{y0} = 0, k_{z0} = 0$. Similar representations can be given for walls with boundary conditions of Neumann type.

From the representations in Eq. (5.14) and Eq. (5.15) it follows that the task of evaluating the convolution in Eq. (5.1) for arrays in metallic wall waveguides is accomplished by superposing the results for periodic arrays in free space. Therefore, the computational time results in Section 5.6 apply here as well (but they need to be multiplied by a factor of 2 or 4 for the cases of parallel plate and rectangular cross-section waveguides, respectively).

(c) Periodic 1D and 2D arrays in layered media

Consider a 1D or 2D periodic array of unit cells above and parallel to a layered medium. In this case, the PGF is given similarly to Eq. (5.2) but with G_0 replaced by the layered medium Green's function given via the Sommerfeld integral [36]. FPIM can proceed but several modifications may be required. In particular, the layered medium Green's function may need to be regularized to result in slow spatial variations, e.g. by extracting its quasi-static components. Further details on such regularizations are given in [163]. In addition, the translation invariant property should be defined with respect to the image of the unit cell defined relative to the layered medium top interface. Once such a Green's function is obtained, FPIM remains mostly unchanged.

The situation is somewhat more complicated for unit cells embedded within a multilayered medium, in which case PGF may lose the translation invariant property. This may result in a higher computational cost for tabulation and interpolation. These requirements are similar to those reported in recent works [163], where interpolations

were used to compute PGFs for layered media. Methods developed in these works can be used here as well. Further research is required to fully implement these ideas for general periodic layered media problems.

5.8 FPIM on GPUs

Implementing FPIM on GPUs would be very similar to B-AIM without the near-field correction stage. The near-field component of the FPIM can be separated accelerated by either B-AIM or NGIM and the far-field component consists of projection, FFT transformation and interpolation. The computational time and accuracy behavior would be very similar to those of B-AIM.

The results of GPU FPIM is shown in the Table x, and Figure x. The speed-ups are around 150x for sufficiently large problems. This speed-up is similar to what GPU B-AIM provides while accelerating the CPU sequential B-AIM.

Table 16 Computational time for FPIM on CPUs and GPUs

N	CPU Direct	CPU Prepro-cessing	CPU Execution	GPU Prepro-cessing	GPU Execution	Speed-up
2^{12}	3.54e+02	4.54E-01	2.12E-02	8.90E-01	1.17E-03	18.1
2^{15}	2.26E+04	6.88E-01	1.75E-01	3.07E+00	1.93E-03	90.6
2^{18}	1.45E+06	9.23E+00	1.83E+00	1.19E+01	1.16E-02	157.7
2^{20}	2.32E+07	1.31E+01	7.65E+00	4.83E+01	5.28E-02	144.9

The time shown in the table is in seconds. The CPU version of FPIM is sequential running on a single core of Intel i7-920 CPU. The GPU version runs on NVIDIA Geforce GTX 480 GPU. Only far-field computational time is shown in this paper.

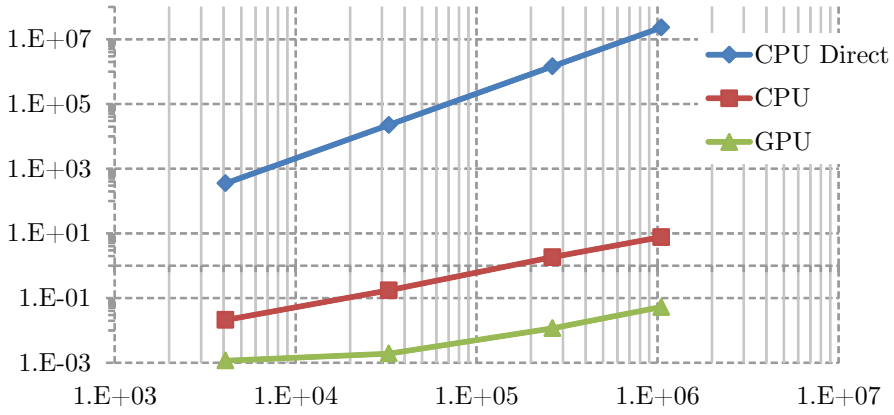


Figure 23 The computational time of FPIM on CPUs and GPUs. The execution times are shown and compared to the CPU direct evaluation time. The asymptotic complexity of FPIM decreases from $O\left(N^2\right)$ to $O\left(N\log N\right)$ and the speed-ups are around 150x for problems larger than approximately 30 K.

5.9 Summary

We have presented a FPIM for the rapid evaluation of electromagnetic potential field at N observers generated by N sources in a general periodic unit cell. FPIM is based on splitting the field into its near- and far-field components. The near-field component represents the field generated by a finite number of sources in and around the prime cell. This component can be evaluated rapidly using any hierarchical or FFT-based method. The far-field component of the field, which has slow spatial variations within the unit cell, is found via three steps: tabulating the PGF at sparse source and observer grids, using these tabulated PGFs to calculate the field at a sparse observation grid, and interpolating from the observation grid to the actual observers. In the low- and moderate frequency regimes, i.e. for small and moderate periodicities, FPIM has the computational cost of $O(N)$ with $O(1)$ evaluations of PGF. In the high- or mixed-frequency regimes, i.e. for electrically large computational domains with dense source constellation regions, FPIM has the computational cost scales of $O((D/\lambda)^3 \log(D/\lambda) + N_q N)$ with $O(N)$ evaluations of the PGF.

The innovations brought by FPIM are summarized:

1) The computational time of FPIM is much smaller than that of the direct evaluation. Significant speed-ups over the direct method are obtained, starting from N as small as 60 and up to any limit determined by the available memory (e.g. sizes up to $N = 2$ million are shown in Sec. 5.6). This performance is obtained

for a wide range of source-observer distributions, including volumetric and surface distributions that can be uniform and highly non-uniform.

2) The PGF evaluations in the FPIM can be done via any existing method. In many practical cases, simple spectral Floquet expansions can be used. This makes implementing complicated acceleration techniques for the PGF evaluation unnecessary.

3) The method is kernel (i.e. PGF) independent, and the preprocessing and field evaluation stages are completely separated. Therefore, FPIM can handle many problem types, which a certain type of PGF is available, including arrays in free space, metal wall waveguides, and layered media. The reason is that only the first step of the algorithm (PGF tabulation) in Section 5.4.1 is kernel dependent. Once the PGF is tabulated at a sparse grid, the rest of the algorithm remains mostly unchanged.

4) FPIM is simple to implement and can be incorporated into existing IE solvers, provided a conventional fast code for the near-field field evaluation is available.

5) GPU acceleration of FPIM is very similar to B-AIM and much simpler as the most time-consuming near-field correction stages in B-AIM is unnecessary for FPIM.

FPIM can be used to accelerate IEs for various periodic unit cell problems with many applications in the microwave engineering and optics, including frequency-

selective surfaces, artificial impedance surfaces, periodic leaky wave antennas, periodic grating filters and couplers, waveguides, and photonic bandgap structures.

5.10 Acknowledgement

This chapter is a reprint, with some minor modifications, for the clarity reasons, of the materials appeared in

- S. Li, D. A. Van Orden, and V. Lomakin, "Fast periodic interpolation method for periodic unit cell problems," Antennas and Propagation, IEEE Transactions on, vol. 58, no. 12, pp. 4005-4014, 2010.

6 Electromagnetic and micromagnetic simulators on GPUs

6.1 The micromagnetic simulator (FastMag)

The micromagnetic simulator that utilizes the fast methods described in Chapter 4 for field evaluation is called FastMag [31]. FastMag takes various kinds of meshes that model the geometries of objects. Meshes may consist of tetrahedrons or hexahedrons for general-purpose simulations or various other elements such as Voronoi cells, for specialized granular magnetic recording media simulation. The solver is modular, so different third-party function units can be added or removed from the solver with ease. It uses the CUBIT meshing application for mesh generations, open source CVODE package to do the time integration, and Paraview for post-processing and visualizing results. The arrangement and relations between components is shown in the Figure 24.

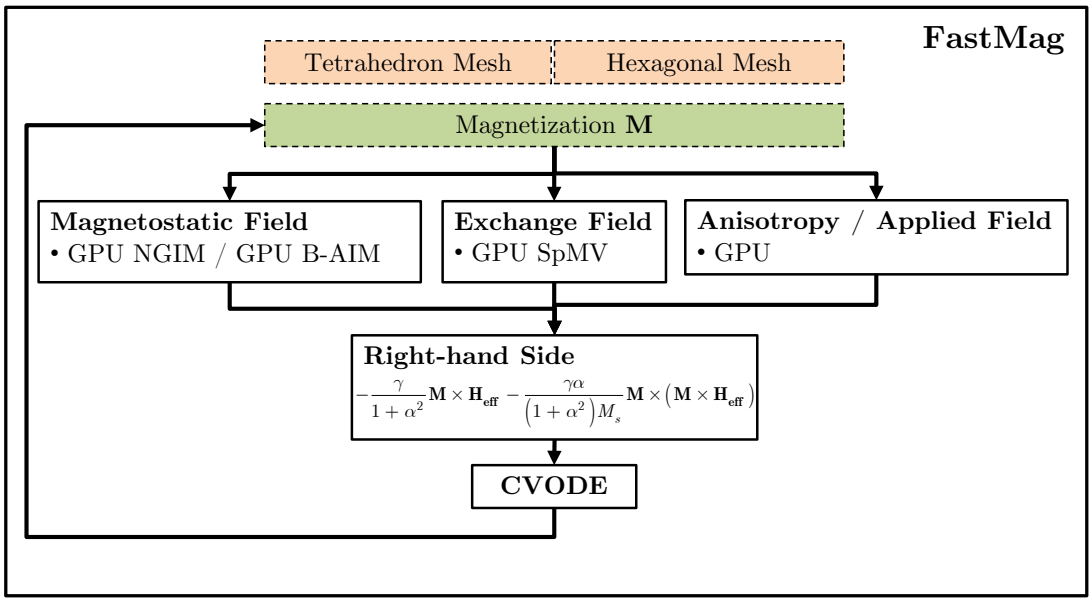


Figure 24 Block diagram of FastMag. The meshing and visualization component are third-party open source packages.

FastMag is distinct in that it has a very high computational performance and it can handle complex and realistic magnetic devices and systems. An important component enabling the high performance of FastMag is the use of GPUs to overcome a number of computational bottlenecks [30, 31].

Fastmag has been used to design advanced magnetic recording systems [49] as well as investigate complex physical phenomena inside magnetic materials [111-113]. e. g. a magnetic recording head meshed to approximately 126 million tetrahedrons could be simulated [31]. Here we listed several sample micromagnetic simulations that have been run on the platform and performance the solvers achieve.

6.1.1 Large scale bit patterned media array simulations

Table 17 demonstrates the performance of the FastMag micromagnetic solver on bit patterned media array simulations. In this series of simulations, a large number of cubic magnetic elements staggered to form a large planar array. Each of the magnetic elements are of single layer with the average saturation magnetization $M_s = 700\,\mathrm{emu}/\mathrm{cc}$, anisotropy field $\overline{H}_K = 25\mathrm{kOe}$, lateral length $w = 12\mathrm{nm}$. The array has inter-element spacing $\overline{b} = 12nm$ and parameter fluctuation of $\sigma_{H_K} = 15\%$ and $\sigma_b = 10\%$. FEM-based solver might have reduced efficiency while tackling this kind of problems because because of many surface nodes, whereas finite difference based solvers would reduce the efficiency because of the white space between the array elements. Moreover, the problem is stiff, so that efficient implicit time integration techniques are required. However, since FastMag evaluates long range magnetostatic fields using integral equation methods and uses implicit time integration methods, so it handles this structure efficiently.

Table 17 The computational times of the bit patterned media array simulation

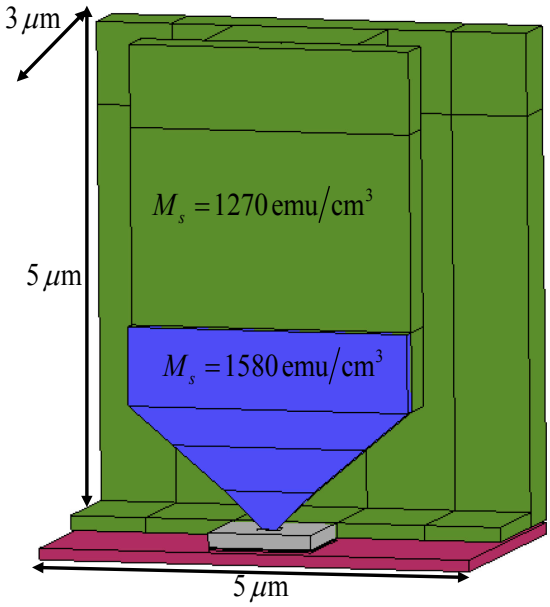
Array size	Number of tetrahedron	Time (System 1)	Time (System 2)
20x20	14.4K	5e-3 s	1.1e-2 s
200x200	1.44M	0.284 s	0.573 s
300x300	3.24M	0.560 s	1.29 s
800x800	23.04M	N/A	9.35 s
1600x1600	92.16M	N/A	40.8 s

Computational times shown in Table 17 are those required to progress one time step in our ODE solver. The system 1 is a workstation with Intel i7-920 CPU and NVIDIA Geforce 480 GPU. The system 2 is a workstation with Intel Xeon X5482 CPU and NVIDIA TESLA C1060 GPU. The larger graphics memory available on the TESLA board extends the capability of our FastMag solver close to problems with 100M tetrahedron.

6.1.2 Magnetic recording head simulations

We have also tested our FastMag solver on objects with more complex geometric shape, such the magnetic recording head. Figure 25 (a) shows a typical structure of a magnetic recording write head. It is a very challenging task to simulate the dynamics in a recording head as the size of the geometric features varies greatly across different parts.

Differences in the magnetization dynamics are found using meshes with different discretization rate. For example, the maximal recording fields generated by the head are shown to be lower than actual field if the coarsest mesh is used. Proper discretization is also required to reveal the interaction between the shields (the gray component) and the soft under layer (the magenta component) [49].



(a) A typical magnetic write head

Largest element	**# of tetrah.**	**Time per 1 ns**
130 nm	130 K	1.75 min
57 nm	1.2 M	17 min
33 nm	4.8 M	107 min
10 nm	126 M	~3 days

(b) Time used to simulate the dynamics inside the head with different discretization

Figure 25 (a) Model of a state-of-the-art magnetic recording head and its geometrical dimensions. (b) The computational time for 1 nanosecond of simulation time using different meshes.

6.2 The electromagnetic simulator

The electromagnetic solver in the author's research group follows a similar structure as other iterative integral equations solvers using the GMRES iterative algorithm. The solver has several revisions, using different basis functions such as RWG basis for surface problems or SWG for volumetric problem and may solve for different unknown sources, fields or potentials. The electromagnetic solver utilizes the fast methods discussed in Chapter 4 for field evaluations, so it runs much faster than its sequential counterparts on CPUs. The small memory footprint of NGIM also gives the

solver ability to handle problems with millions of degrees of freedom or problems in high-frequency domain on a desktop workstation. This solver can also handle periodic problems utilizing the FPIM discussed in Chapter 5. In this section, the author would also like to show several example simulations and the performance of the solver. In all examples, the simulations were done on the same desktop computer as in Section 6.1, with Intel i7-950 CPU, 24 GB of system memory and NVIDIA GeForce GTX 570 GPU with 1.2 GB of global memory.

6.2.1 Scattering from free-standing spheres

Figure 26 shows the radar cross-section (RCS) of a free-standing sphere of diameter D and permittivity $14 - j8$ at the wavelength $\lambda = D / 3.8$. RCS obtained via the Mie scattering approach and via the volumetric integral equations accelerated by GPUs using B-AIM are shown and compared. The number of Mie series terms was chosen to achieve a full convergence. The number of SWG basis functions was 1,320,198. The process of solution contains 174 iterations with a 1e-2 RMS convergence error. The preprocessing time was 9 seconds and the solution time was 36.7 minutes.

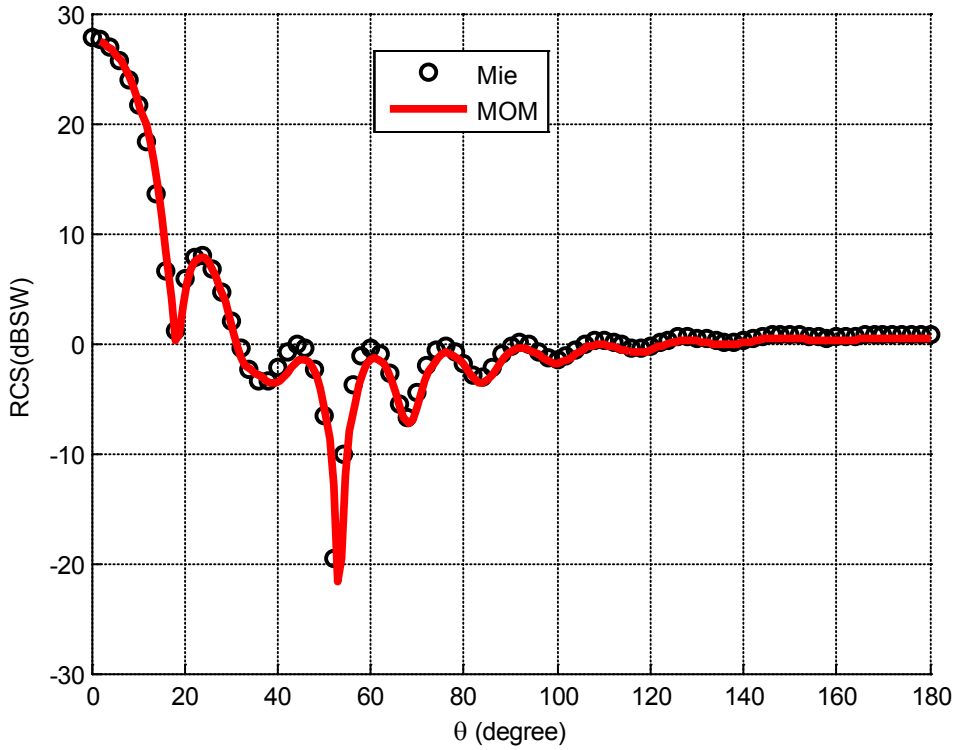
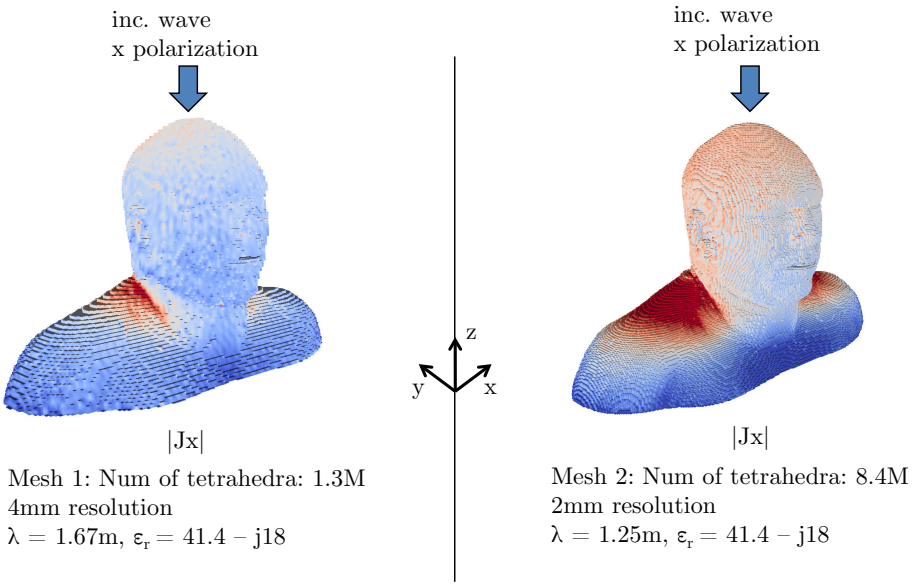


Figure 26 The RCS of a free-standing sphere. The red curves are generated using the IE solver with GPU accelerated fast methods.

6.2.2 Scattering from human upper body

Figure 27 demonstrates the performance of the solver on human body scattering problem. In Figure 27, current density is shown on a highly detailed human body being exposed to incident EM waves. The permittivity of human tissue is $15 - j10$. The mesh resolution in the left sub-figure is 4 mm which leads to 1.3M tetrahedrons and the right sub-figure has 2 mm resolution leading to 8.4 tetrahedrons. The incident wave is a plane wave coming from the top with a wavelength of 0.2 m. The figure shows the real

part of the x-component of the current. The results were compared against those from other EM solvers with different basis functions and variables. The simulation involves around 100 iterations. The smaller problem takes about 10 minutes and the larger problem takes about 48 minutes.



Credit : Human meshes provided by Prof. Ali Yilmaz,
http://web2.corral.tacc.utexas.edu/AustinManEMVoxels/

Figure 27 Electrical current distributions along x axis on human body excited by an incident wave above the head.

6.2.3 Scattering from periodic meta-materials

The volume integral equation solver shown above also works for problems with periodic boundary conditions using appropriate Green's function and acceleration schemes.

Figure 28 shows the reflection coefficients of a doubly periodic array consists of dielectric cubes, illuminated by a normal incident wave along the z-axis. Each unit cell comprises a cube of size 1mm and is made of material of the permittivity 4. The total number of basis functions for this structure is 309,741. From the figure we can observer an obvious resonant behavior around $\lambda / d_x = 1.07$, which is a typical Wood anomaly property of periodic gratings [173]. The total computation time of the VIE solver was 50 min per data point using 450 iterations plus the preprocessing.

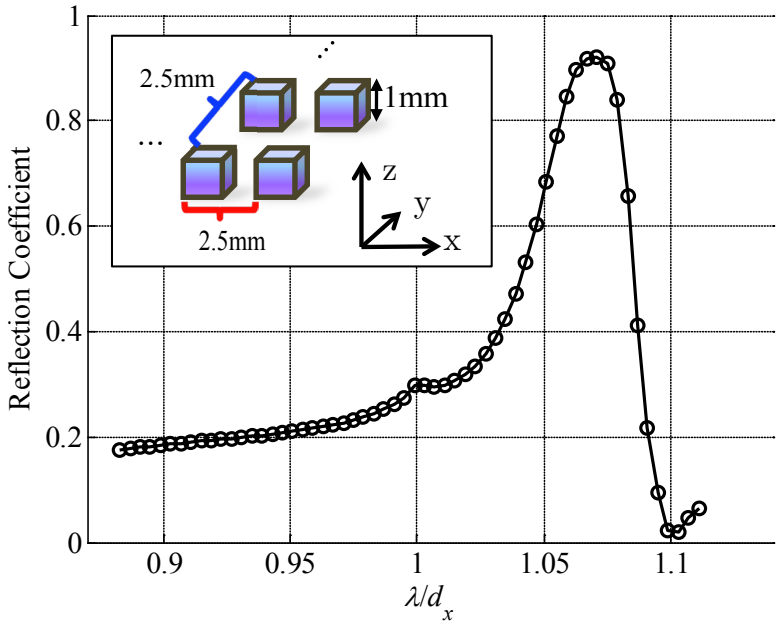


Figure 28 The normal reflection coefficient of a doubly periodic array. The Wood anomaly is achieved around $\lambda / d_x = 1.07$

Finally, Figure 29 shows a larger scale simulation of a doubly periodic structure with a complex unit cell comprising multiple split ring resonators; such structure can

be used as an isotropic negative index meta-material. It has a permittivity of $-4.06 - j2.48$, which is a very popular topic in both academic and industrial world nowadays. The number of iterations required to generate a single data point in this figure is around 11. The total time to generate a data point is about 6 minutes with the preprocessing time included. Again a resonant behavior is observed in agreement with anticipated behavior.

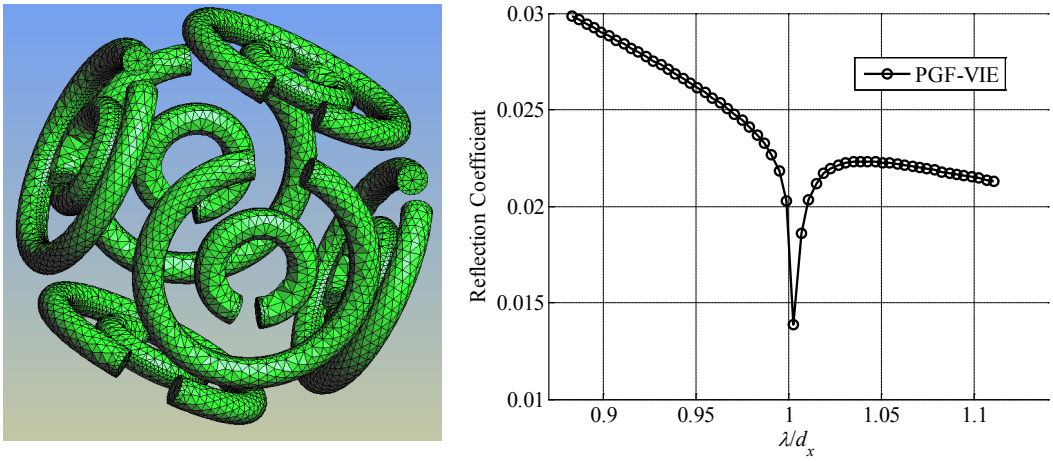


Figure 29 The reflections coefficient of a doubly periodic metamaterial structure

6.3 Acknowledgement

Chapter 6 contains results, figures and tables from:

- M. A. Escobar, M. V. Lubarda, S. Li, R. Chang, B. Livshitz, and V. Lomakin, "Advanced Micromagnetic Analysis of Write Head Dynamics Using Fastmag," Magnetics, IEEE Transactions on, no. 99, pp. 1-1, 2012.
- R. Chang, S. Li, M. Lubarda, B. Livshitz, and V. Lomakin, "FastMag: Fast micromagnetic simulator for complex magnetic structures," Journal of Applied Physics, vol. 109, no. 7, pp. 07D358-07D358-6, 2011.

7 Micromagnetic simulations of advanced magnetic recording media and systems

In this chapter, we show several important simulations done using our micromagnetic solvers. In Section 7.1, simulations are used to investigate physical phenomena within a newly proposed bit patterned media (BPM) configuration, called the capped bit patterned media (CBPM). CBPM is thought to have multiple advantages over BPM with fully uncoupled elements and are being investigated by both simulations and experiment [25, 58, 112, 113]. In Section 7.2, a proposed new recording system that uses ferromagnetic resonance to switch magnetic recording materials with high anisotropy is proposed and investigated. This magnetic recording system is called microwave-assisted magnetic recording (MAMR) and it is considered as one of the energy-assisted recording schemes that can extend the limit of aerial recording [19, 95, 97, 170, 171, 181].

7.1 High density capped bit patterned media

7.1.1 Introduction

Bit patterned media (BPM) comprise arrays of separated magnetic islands. They are expected to provide solutions to the superparamagentic effect [159, 169] that physically limits the magnetic recording density of modern hard disk drives. The array

elements of BPM can be made of different materials and in different geometric shapes. For example, exchange-coupled (composite) elements may be used to achieve reduced reversal fields while keep the energy barrier the same [151, 156]. However, this array of closely situated discrete islands made of magnetic material inevitable introduces significant stray field, which increases the switching field distribution (SFD) across the islands. These distributions lead to bit-errors and can significantly limit the BPM recording densities [1, 72, 142]. In addition, the stray fields lead to a significant loss of the thermal stability.

In this section, we describes a BPM configuration that consists of an array of hard elements and a continuous soft layer, referred to as a "cap layer", which is placed at the bottom of the array and is ferromagnetically coupled to the array's elements through their common interfaces (Figure 30) [58, 96]. The motivation to introduce this structure is to use the exchange interaction in the soft cap layer to counter the magnetostatic interaction between the hard elements thus reducing the distributions of the switching fields as well as improving thermal stability. This concept is conceptually similar to coupled continuous-granular (CGC) perpendicular recording media [122, 153].

7.1.2 Structure configuration

The CBPM comprises hard elements that are arranged into an array and are coupled to a continuous soft layer with a surface energy J_s through the common interfaces. The hard elements have a vertical uniaxial anisotropy of energy density K_h,

length a_x, width a_y, and thickness t_h. The cap layer has thickness t_s and is assumed to be perfectly soft. All materials have a damping constant α, saturation magnetization M_s, and exchange length $.l_{\text{ex}} = \sqrt{A}/M_s = w$. with A the exchange constant. The spacing between the hard elements in the BPM array is B. The geometric structure is shown in Figure 30

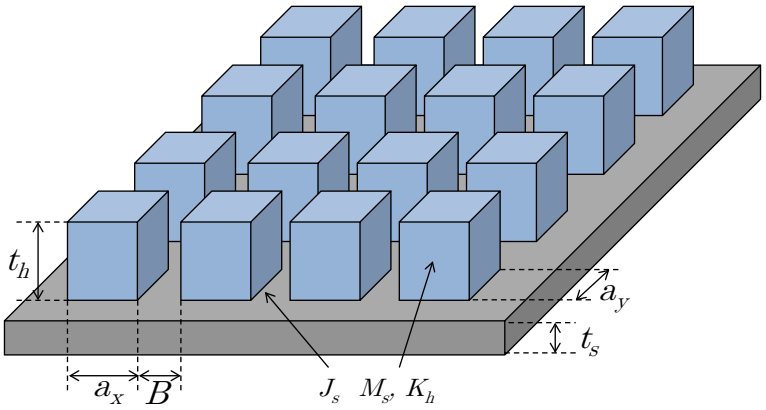


Figure 30 The geometrical structure of CBPM. The structure is shown as a two-dimensional periodic structures.

The switching behavior of the media is simulated with an applied field expressed as $H_{ext} = -H_a erf\left(2t\ /\ \tau\right)\left(\mathbf{y}\cos\varphi + \mathbf{x}\sin\varphi\right)$. This field is applied with an angle to the vertical axis to simulate the field off the edge of a recording head pole. The reversal field H_r is defined as the threshold field that switches magnetization in the hard elements from the initial $\mathbf{y}$ to the $\mathbf{-y}$ direction.

The switching fields of the CBPM structure are studied by numerically solving the Landau-Lifshitz-Gilbert equation with discretization chosen to obtain convergence. In all simulations, $M_s = 1250\,\text{emu/cm}^3$ and $J_s = 11.25 \times 10^6\, t_h$, $\tau = 0.1\,\text{ns}$, $H_K = 60$ kOe. The damping constant varies in the range between $\alpha = 1$ and $\alpha = 0.1$, which with a chosen field rise time τ corresponds to damping and precessional reversal regime, respectively [104, 105]. To exemplify the operation of the proposed structure, we considered an array of three hard elements arranged into a linear array and coupled to continuous cap layer.

7.1.3 Switching field distributions

We did a series of simulations to test our hypothesis that the continuous cap layer can compensate the magnetostatic interaction. Figure 31 shows the switching field as a function of inter-element spacing for three different types of BPM. The solid line is obtained when the initial magnetization in three hard elements is set to be in the same direction (i.e. parallel configuration) and the dash line is obtained when the initial magnetization of the center element is opposite to the neighboring elements.

As expected, due to magnetostatic interactions, the reversal fields are smaller for the parallel configuration and the gap between the reversal fields in the two scenarios decreases with increasing element separation. This gap can be significant for smaller separations between the elements and restrict the achievable recording densities due to

the resulting reversal field distributions and inability to provide a proper writing window.

However, the reversal field behavior is very different for the CBPM. From Figure 31 (c), it can be seen that for this specific setting, H_r is larger under the opposite configuration when the separation is small. This behavior, which is opposite to that seen in Figure 31 (a) and (b), is a manifestation of the influence of exchange interactions through the cap layer. The exchange field tends to align nearby spins in the same direction and transfers this influence to the hard elements via their ferromagnetically coupled common interfaces. This exchange influence reduces with increasing separations and a balancing point can be found where there is no distribution of the reversal field is present. For the scenario shown in Figure 31 (c), this is achieved at $(B - w) / w = 1$. For larger separations, some distribution is present, but this distribution is much smaller than that of the disconnected (homogeneous and composite) arrays.

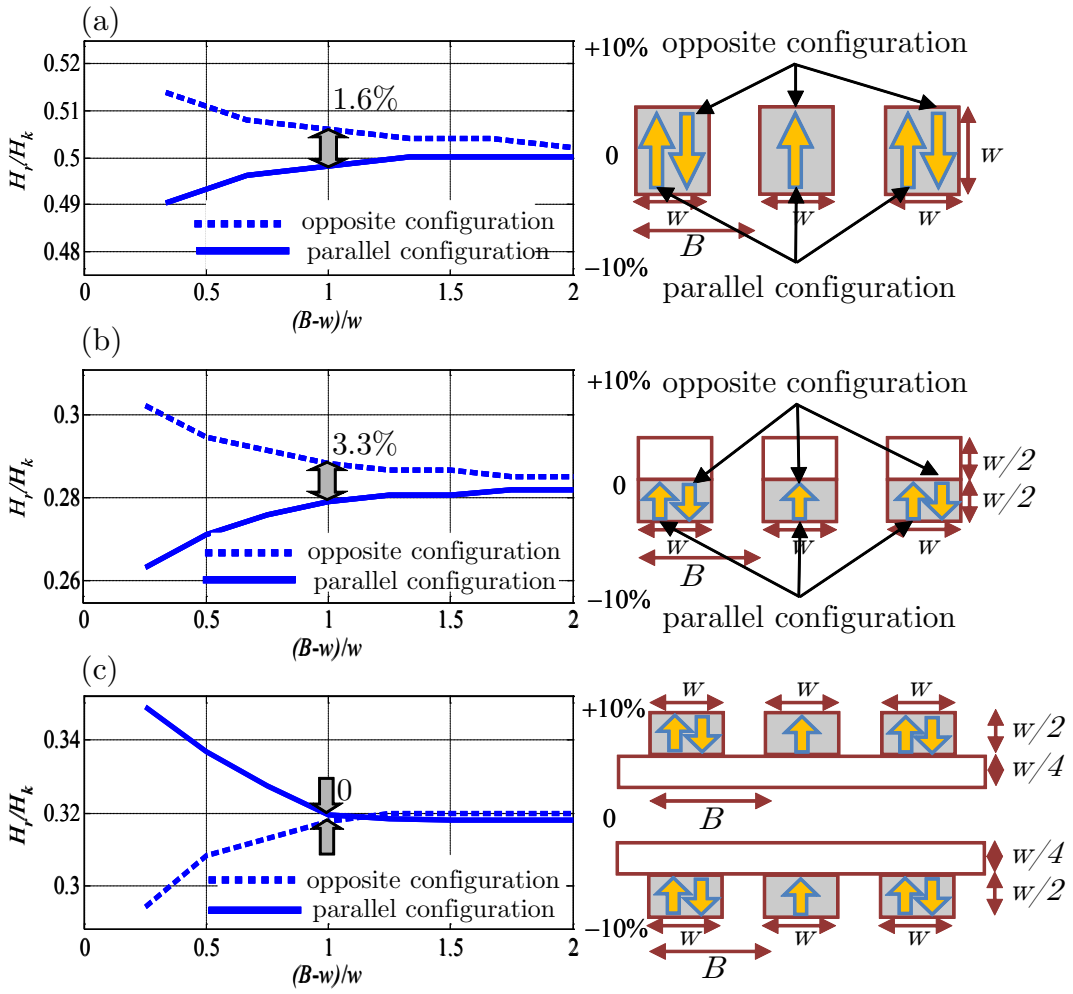


Figure 31 Normalized reversal field H_r / H_k vs. the hard element spacing $(B-w)/w$ for three BPM structures and two magnetization configurations for each structure.

The behavior of the reversal curves, the balancing point, and the distributions of the reversal field can be tuned by carefully choosing the structure parameters. In addition to the coupling strengths and layer thickness, and we have also considered the

influence of a finite anisotropy in the soft capping layer. The details can be found in Ref. [96] and the obtained behavior was qualitatively similar with some quantitative differences. In addition, we studied the CBPM in the regime of precessional reversal, which is obtained for under sufficiently short (but practical for the composite elements) rise times [96]. Similar compensation phenomena were obtained apart from an additional reduction of the reversal field associated with precessional mechanisms. Again, there are only quantitative differences. So as summary, for all consider CBPM, available "knobs" for tuning the balancing point includes J_s, t_s and the anisotropy of the capping layer.

7.1.4 Readback process

In addition to altering the writing process the introduction of the capping layer is also expected to affect the readback signal. The readback signal was calculated using reciprocity and approximate expressions for the head field [96]. We find pronounced differences for CBPM with the capping layer on the top and at the bottom of the hard element array and for different bit separations.

Figure 32 shows the readback voltage of a interleaved bit pattern of six elements for a conventional composite (dual-layer) media, CBPM with capping layer on top, and capped media with capping layer at the bottom ("reverse cap"). For conventional composite media, the readback signal is nearly insensitive to whether the soft section on the top or at the bottom.

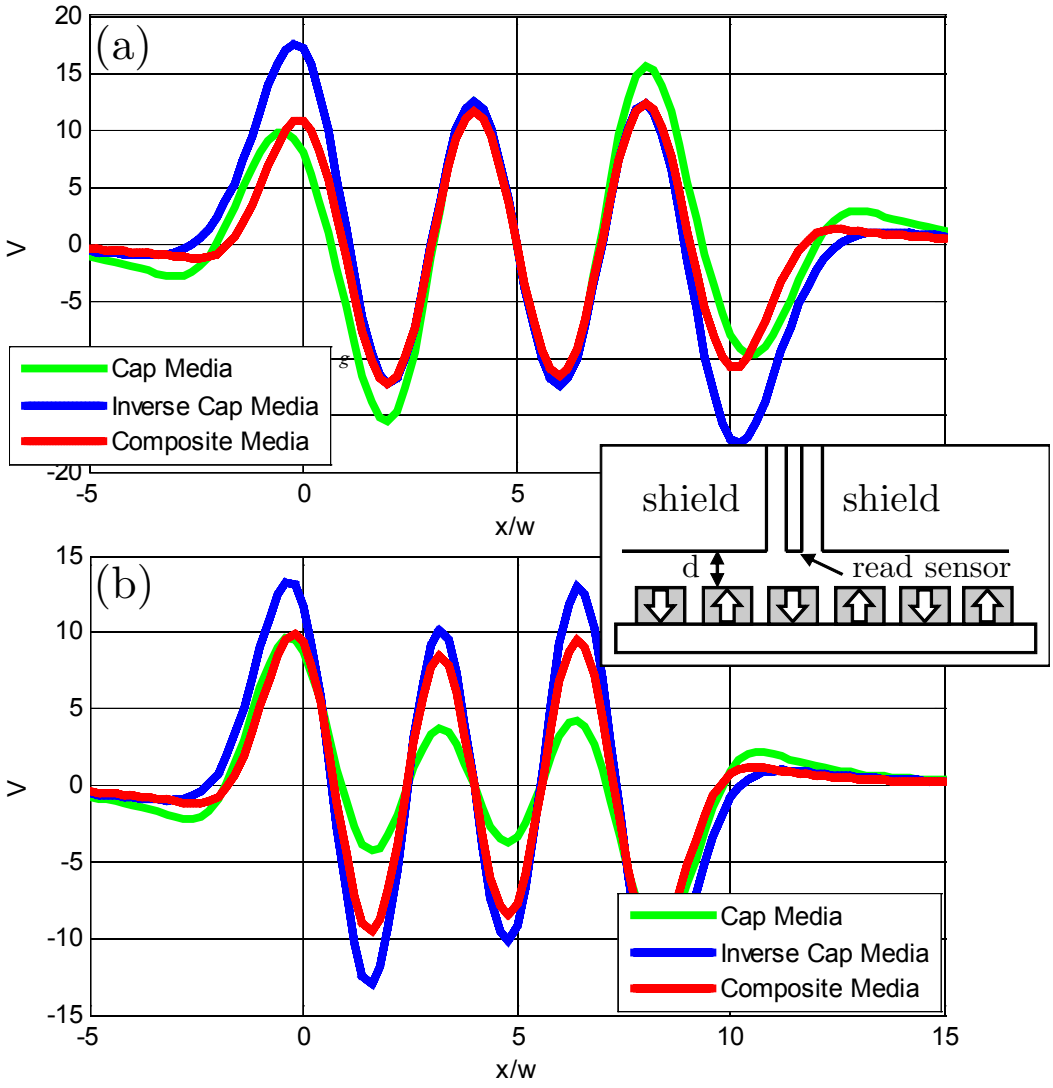


Figure 32 The readback signal of an interleaved bit pattern from a double shielded reading head (shown in inset), for three different material structures: conventional patterned media, the "cap" media and the "inverse cap" media. The spacing between the elements in (a) is the same of the hard element width; (b) 60% of the hard element width. The parameters of the read head defined in the inset are $d = t_h$, $t = 0.4t_h$, $g = 1.5t_h$

For bit spacing greater than $B > 2w$ ($B = 2w$ in Figure 32 (a)), the readback signals for all considered cases are similar with some differences at the left and right edges. For smaller spacing between the elements ($B = 1.6w$ in Figure 32 (b)), the readback signal decreases noticeably for the CBPM with the capping layer on top. This is due to the broadening of signal by the soft layer. For the inverse cap media, the readback signal slightly increases with the approximately the same transition width. Putting the soft layer away from the head might increase the minimal switching field so optimal medium parameters should be found as a trade-off between read-and write-field requirements.

Finally, it should be mentioned that the capping layer does not directly introduce transitional noise as in conventional granular media since it is assumed to be made of perfectly soft continuous material. However, noise can be introduced by distributions of the hard elements' position, material properties, and shape as in conventional BPM.

7.1.5 Summary

In this section we introduce a CBPM configuration that comprises an array of hard elements coupled to a continuous soft layer. There are three benefits brought by this additional layer of soft material. While it substantially lowers the switching field, it also provides a mechanism of to compensate the effects of magnetostatic field interactions between the array elements. Optimal structure parameters can be chosen to

minimize the distributions of switching field brought by the different magnetization states in a BPM array. This can allow for lower bit-error rates and can improve the BPM performance. The readback signal is not noticeably degraded compared to conventional media.

7.2 Microwave assisted magnetic recording

7.2.1 Introduction

A major limitation to the continued evolution of high-density magnetic recording is the superparamagnetic effect, which leads to spontaneous reversal when magnetic particles become too small [159, 169]. Overcoming the superparamagnetic effect requires using materials with increase thermal stability which is often achieved through increased anisotropy. However, high anisotropy often translates into excessively high reversal fields, which are hard to achieve using a traditional recording head. Several methods including heat-, precessional-, and microwave-assisted magnetic recording schemes have been proposed to solve this writability problem [105, 115, 145, 158, 181]. Microwave-assisted magnetic reversal (MAMR) significantly reduces the reversal field when the microwave field frequency matches the ferromagnetic resonance (FMR) frequency of the media elements [149, 157]. Applying this MAMR scheme on ECC media would further pushes the limit of anisotropy of materials that we can write to. In

addition to solving the writability and thermal stability problems, exploiting the resonant properties of the reversal fields may also suggest novel approaches for high density magnetic recording.

In this section, we show result for MAMR in composite media comprising magnetic elements composed of soft and hard sections coupled ferromagnetcally. Such composite elements have been recently shown to be attractive for magnetic recording due to their reversal and thermal stability properties [53, 156, 166]. We show that composite elements have several unique properties important for MAMR. Composite elements with high anisotropy hard sections can be reversed with relatively low reversal fields, microwave fields, and microwave frequencies. We demonstrate that reversal field dependences in composite elements are completely different in the regimes of coherent and incoherent (domain wall) reversal and reversal dynamics may exhibit surprising behaviors. In addition, we show that fluctuations of the reversal fields caused by fluctuations of the hard layer anisotropy field are substantially reduced compared to those for homogeneous elements. Finally, we also show that MAMR schemes can be used for multilevel recording, in which each layer has a distinct FMR frequency and is addressed by tuning the microwave frequency.

7.2.2 Experiment configuration

The elements investigated comprise exchange-coupled soft (top) and hard (bottom) sections (see the inset in Figure 33). The (bottom) hard section is of size

w, w, d_h in the x, y, z dimensions, with vertical uniaxial anisotropy energy density K_h. The (top) soft section is of size w, w, d_s with vanishing anisotropy. Both sections have a damping constant $\alpha = 0.1$, saturation magnetization M_s, and exchange length $l_{\text{ex}} = \sqrt{A}/M_s$ where A is the exchange constant. For all presented results in the following sections, $A = 10^{-6}\,\text{erg}/\text{cm}$, $\alpha = 0.1$, $M_s = 1250\,\text{emu/cm}^3$ and $\tau = 0.1\,\text{ns}$. The sections are coupled ferromagnetically over their common interface with surface energy J_s. An external magnetic field simultaneously comprises a switching field and a microwave field. The switching field is applied with an angle 45° to the vertical (z) axis in the $x - z$ plane and it has the time dependence $H_r \text{erf}\left(2t/\tau\right)$, where H_r is the reversal field and τ is the switching field rise time. The microwave field is applied along the x axis and it has an amplitude H_{mw} and frequency f_{mw}. For given H_{mw} and f_{mw}, there is a minimal bias field amplitude, referred to as reversal field H_r, that leads to the reversal of the element over a reversal time t_r.

All results are obtained by numerically solving the Landau-Lifshitz-Gilbert equation as described in Section 2 with discretization chosen to obtain full convergence. More simulations with a wide range of α, τ, and J_s were also done but leads to very similar results so are not shown in this thesis. For example, a bit patterned media with pitch of 8nm and $w = d_h = 5\,\text{nm}$ results in a recording density of $10\,\text{Tbit}/\text{in}^2$ with thermal stability above $70\,k_B T$ with the Bolztman constant k_B and temperature

$T = 400\ K$. The chosen parameters are representative of practical materials for high-density recording, such as FePt.

7.2.3 Reversal mechanism for homogeneous and composite media

First, we compare H_r, H_{mw}, and f_{mw} for homogenous elements and composite elements with different thickness of the soft layer. Figure 33 depicts H_r vs. f_{mw} for different type of BPM dots with the same hard layer ($H_K = 60\,\mathrm{kOe}$, $t_h = 1.5w$) but different soft layer. The reversal field dependences for all elements exhibit deep minima. The homogeneous element and composite element with a thin soft section exhibit a typical behavior attributed to MAMR, i.e. resonant curves with deep minima are obtained and reversal occurs for any values of H_a greater than the reversal field H_r (this is visualized by the shadowed areas Figure 33). For the composite element with a thicker soft section, the behavior is completely different. For this type of dots, reversal is only possible in a certain areas in the $H_a - f_{mw}$ plane. Two areas are observed. The top area is the same obtained without any microwave field. The bottom (relatively small) area only exists under microwave field and exhibits resonant properties. Surprisingly, there is a gap between these two areas in which no reversal occurs.

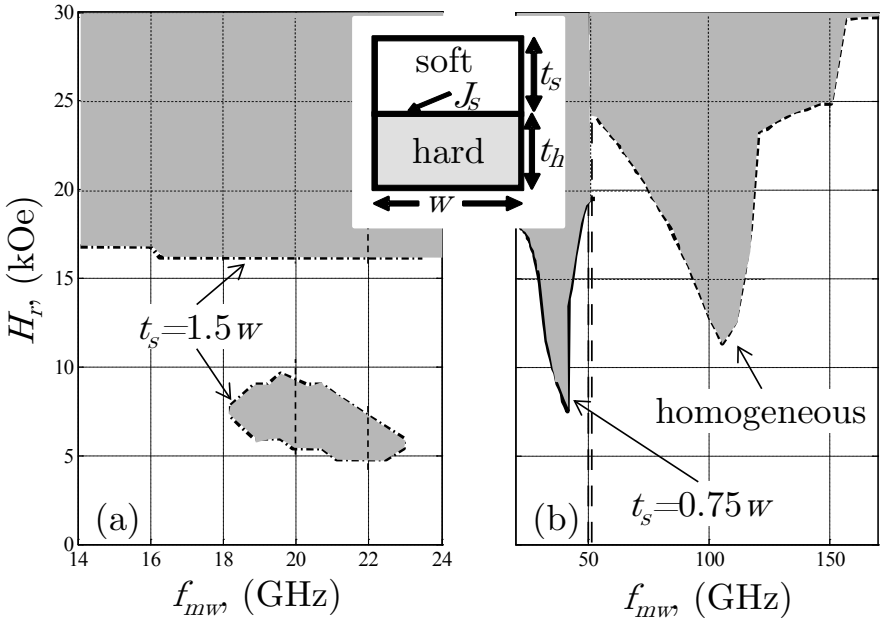


Figure 33 Reversal field vs f_{mw} for different elements with the coercivity $H_K = 60\text{kOe}$, damping constant $\alpha = 0.1$, exchange field $l_{ex} = 1.6w$ and thickness of the hard layer $t_h = 1.5w$. For the composite elements, in (a) the amplitude of microwave $H_{mw} = 0.05H_K$, the thickness of the soft layer $t_s = 1.5w$; in (b) $H_{mw} = 0.07H_K$, $t_s = 0.75w$. Gray areas represent the conditions under which the reversal occurs.

From the Figure 33, we can also see the reduction of resonance frequency when the thickness of the soft layer increases. This is due to the different resonant mechanism in respective types of media. For homogeneous elements, the FMR frequencies are determined mainly by the anisotropy field H_K, but for composite elements, the FMR frequencies are determined by the properties of the soft layer and the inter-layer coupling field, which is smaller than H_K, thus leading to FMR frequency reduction. For thick composite elements with layers thicker than the domain wall length, the reversal in the soft layer is incoherent. The reversal starts in the top part of the soft section and

then a domain wall is formed in the soft section. The domain wall propagates though the soft and subsequently through the hard section. The resonant frequency in this case is mainly determined by the external field with the exchange field. These two fields are much smaller than the anisotropy field H_K thus leading to a significant FMR frequency reduction. In addition, since the influence of ferromagnetic coupling through common interface is weak on the spins in upper part of soft layer, they can be easily switched under a weak bias field resulting in a lower reversal field. Time evolution of the spins in the two regimes is shown schematically in Figure 34 (a), (b), and (c).

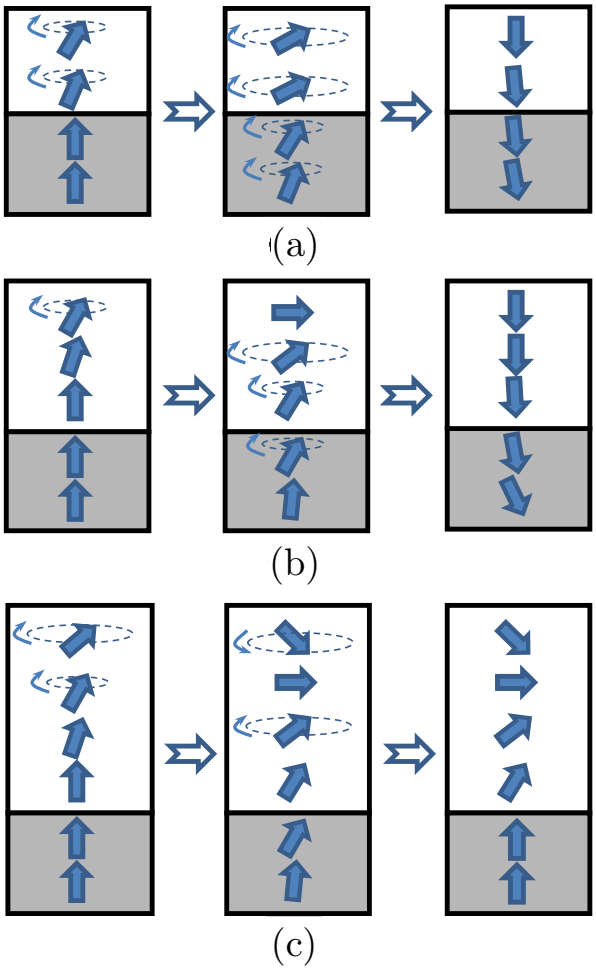


Figure 34 Schematic representation of the spin time evolution in the regime of (a) uniform and (b) non-uniform (microwave assisted domain wall) reversal. In (c), the thickness of soft layer is too large that the domain wall stops before move into hard layer.

7.2.3 Reversal mechanism for homogeneous, composite media

The performance of the MAMR system may be restricted not only by the limitation on maximally achievable head fields and microwave frequencies but also by deviations of the reversal field H_r caused by random distributions of the element

parameters. Among them, random distributions of the anisotropy field H_K can have a crucial influence as they may lead to significant deviations of f_{mw}^{res} and H_r. For homogeneous elements, deviations of f_{mw}^{res} scale proportionally with deviations of H_K , but for composite elements, this deviation of f_{mw}^{res} with respect to H_K might be alleviated.

Figure 35 compares the dependence of H_r versus f_{mw} and H_K for composite elements of different t_s and a homogeneous element. For the homogenous element (shown in Figure 35(b)), f_{mw}^{res} is proportional to H_K, e.g. 10% deviations of H_K lead to about 10% deviations of f_{mw}^{res}. Deviations of H_r are substantially more significant, e.g. 10% deviations of H_K lead to more than 50% deviations of H_r. The situation is very different for composite elements, where deviations of H_r and f_{mw}^{res} are substantially reduced and the area of reversal of these two cases overlap with each other for a major part on the phase graphs. For the composite element with $t_s = 1.5w$, deviations of f_{mw}^{res} are only 3% for 10% deviations of H_K, which represents a five-fold improvement over the homogeneous element. The reduction of the deviations of f_{mw}^{res} has a physical source similar to that leading to the reduction of f_{mw}^{res} itself, i.e. f_{mw}^{res} are significantly affected by the soft section where the field is mostly given by the external and exchange fields but not by H_K. This significant improvement correlates with results obtained for conventional domain wall assisted reversal. Due to the potential improvements to bit error rates, this insensitivity to the anisotropy field distribution is a crucial advantage of composite elements over homogeneous elements.

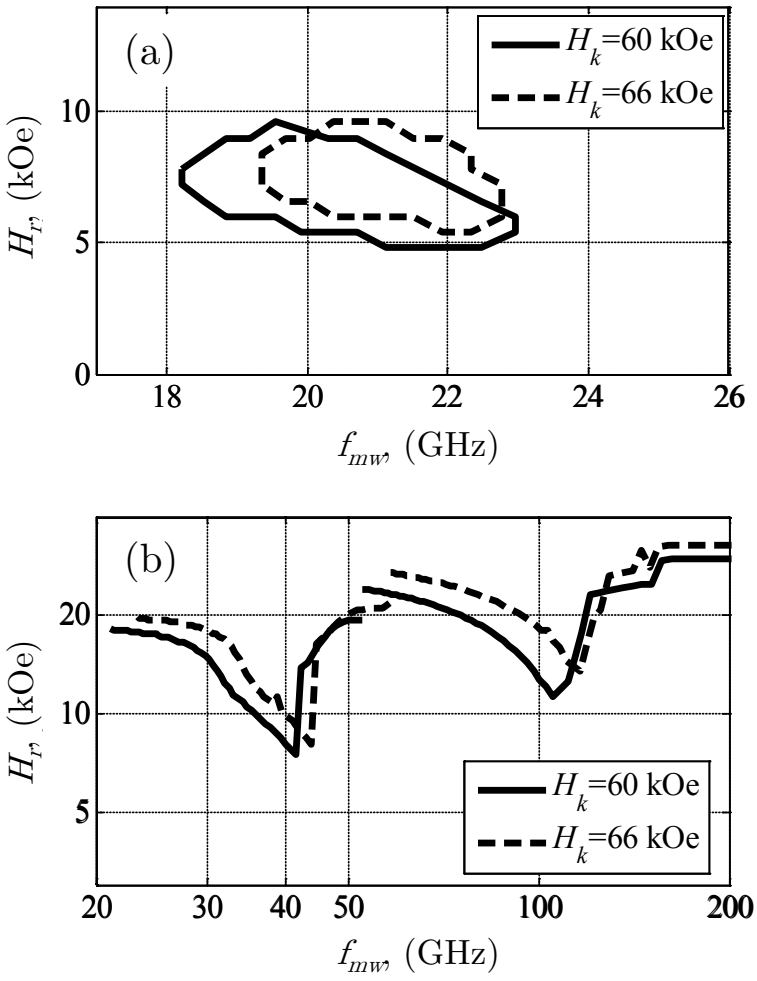


Figure 35 Reversal field vs. f_{mw} for different H_K for composite and homogeneous elements. The damping constant α is always 0.1. In (a) $H_{mw} = 3\text{kOe}$, the thickness of the soft layer is $t_s = 1.5w$; In (b), the right curves are homogeneous element with $t = 1.5w$, under the microwave strength $H_{mw} = 8.4\text{kOe}$, and the left curves are composite elements with $t_s = 0.75w$, $t_h = 1.5w$, under the microwave strength $H = 4.2\text{kOe}$.

It is important to mention that the advantages of the composite elements are obtained without compromising the thermal stability. The maximally achievable energy barrier for the given cross-section is determined by the hard section height t_h related to

the domain wall length t_{dw}. The horizontal domain wall length and energy in the hard sections is given by $t_{dw} = 4\sqrt{A/K_h} = 4l_{ex}\sqrt{2M_s/H_K}$ and $E_{dw} = 4w^2\sqrt{AK_h} = t_{dw}w^2K_h$., respectively. For the elements used to generate in Figure 1, $t_{dw} \approx 6.5\,\text{nm} \approx 1.3w$, which means that the maximal barrier is obtained for approximately $t_h = w$. For this height, the energy barrier is estimated as $E_{dw} \approx 88k_BT$ (with $T = 400K$) and this value approximately matches that obtained numerically via the elastic band method [46]. Further increase of t_h in the composite or homogeneous elements has minor effect on the barrier.

7.2.4 MAMR for multilevel recording

From the results shown in Figure 36, it is clear that the FMR frequencies can be tuned in a wide range by either changing the anisotropy field in the case of homogeneous elements or by changing the anisotropy field, coupling, and geometrical parameters in the case of composite elements. The possibility to tune the FMR and reduce the reversal field near this frequency suggests a novel multilevel recording scheme.

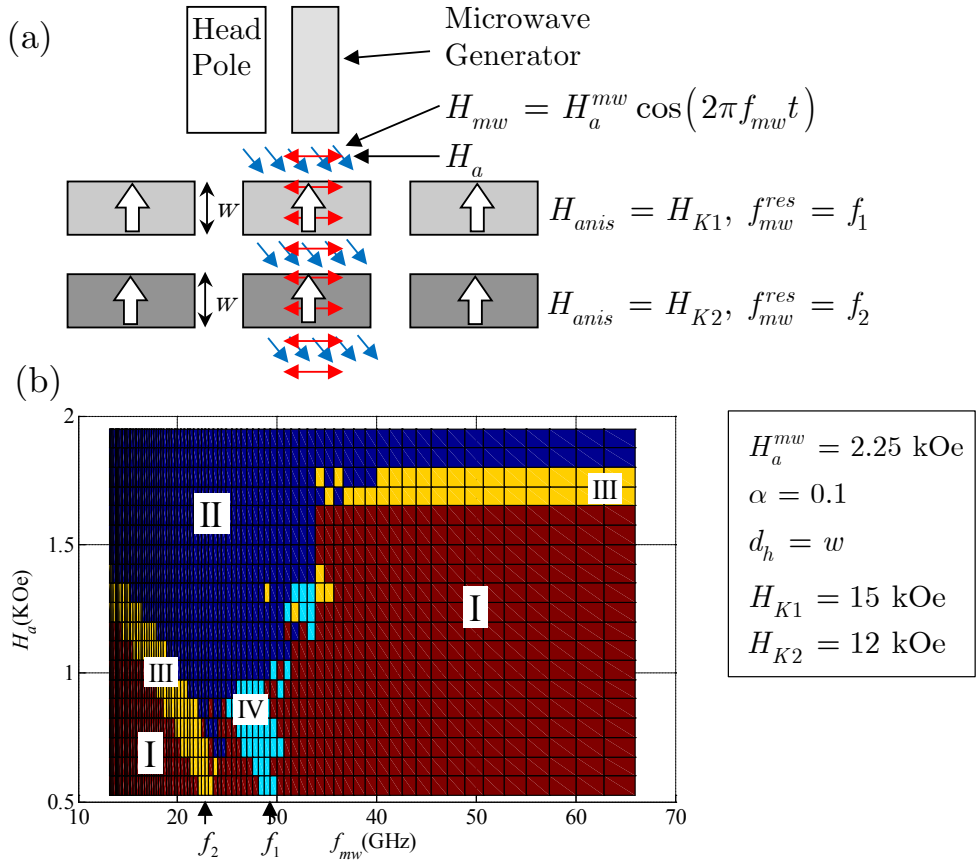


Figure 36 (a) Schematic representation of a multi-layer microwave-assisted magnetic recording system; (b) A reversal pattern of double layer recording system. Four different areas represent different magnetization states of in a two-layer structure comprising homogeneous elements for different microwave frequencies. Area I corresponds to no-switching of any layer. Area II corresponds to switching of both layers. Area III corresponds to switching of the lower layer only. Area IV corresponds to switching of the upper layer only.

The proposed media comprise several layers, where each layer has a different FMR frequency (Figure 36 (a)). The microwave field is used to assist reversing elements in different levels by tuning the microwave frequency to the FMR frequency of the layer being recorded. This method is anticipated to lead to a reliable multilevel

recording scheme with a number of advantages over currently considered multilevel recording methods. For example, there expected to be no need in multi-pass recording since every level can be addressed independently. This scheme does not require addressing the elements in different layers by different strength of the reversal field and can allow for a smaller separation between the layers. In addition, a recording system that can generate microwave fields at several frequencies potentially can address several levels simultaneously thus increasing the recording speed.

To demonstrate the possibility of recording elements with different FMR frequencies independently, we consider an example of a two-level system comprised of homogeneous elements (Figure 36 (a)). In this system, the element in Layer 1 and Layer 2 have anisotropy $H_{K1} = 15\text{KOe}\,\text{and.}\,H_{K2} = 12\text{KOe}$, respectively. All elements are of size $w \times w \times w$ with $w = 10\text{nm}$ and have $M_s = 500\,\text{emu/cm}^3$. The separation between the layers is w. The microwave and bias fields are applied simultaneously to both layers. Figure 36 (b) shows the final magnetization states in the two layers as a function of microwave frequency and the bias field. Area I and II respectively represent regimes of non-reversal and reversal of both layers. Area III and Area IV respectively represent regimes where Layer 1 and Layer 2 can be reversed individually. In practical systems, due to the gradient of head fields, the field is weaker on the layer farther from the head pole and it is reasonable to put the layer with lower anisotropy farther than the layer with higher anisotropy. From Figure 36, it is shown that the

field and element parameters can be found that lead to individual switching of the layers with different resonant frequency. Various media elements can be used. As for the generation of local microwave fields with sufficiently high frequency and strength generated, devices such as the spin-torque driven oscillators might fulfill the requirement. Combined with a conventional recording head, they can result in a system that generates both switching fields and assisting local microwave fields.

7.2.5 Summary

In this section, we investigated reversal properties of homogenous hard magnetic elements and exchange-coupled composite elements with different soft layer thickness under the influence of a local microwave field. Composite elements allow for a significant reduction of the reversal field, the microwave field, and the FMR frequency as compared to homogeneous elements. MAMR behavior in composite and homogeneous is found to be completely different due to the phenomena associated with domain wall formation and propagation. In addition, the reversal field for composite elements can be much less sensitive to the element anisotropy field distributions than for homogeneous elements, which is crucial to allow reducing bit error rates. However, there are also several obstacles that may complicate practical implementations of MAMR schemes and the most critical one is the microwave source. High anisotropy materials may require microwave field strength and frequency may be very high. Such strong fields

and high frequencies are hard to achieve in practical recording systems even using spin-torque driven devices [76, 134, 180].

7.3 Acknowledgement

Chapter 7 consists of results and discussions from papers:

- S. Li, B. Livshitz, H. N. Bertram, E. E. Fullerton, and V. Lomakin, "Microwave-assisted magnetization reversal and multilevel recording in composite media," Journal of Applied Physics, vol. 105, no. 7, pp. 07B909-07B909-3, 2009.
- S. Li, B. Livshitz, H. N. Bertram, A. Inomata, E. E. Fullerton, and V. Lomakin, "Capped bit patterned media for high density magnetic recording," Journal of Applied Physics, vol. 105, no. 7, pp. 07C121-07C121-3, 2009.

8 Summary and future directions

8.1 Summary

The contributions of this thesis are mostly in three different areas: a) the development of fast methods for field evaluations in computational electromagnetics and micromagnetics on massively parallel GPU architectures; b) highly efficient micromagnetic and electromagnetic solvers built up on those fast methods; c) analysis and design of complex magnetic systems using the fast solvers.

The thesis reviewed the current status of newly emerged massively parallel processors and their impacts on scientific computing. Massively parallel processors, GPGPUs being one of such type, are serious contenders in the field of high performance computing recently. Though different in architecture and slightly more difficult to program than traditional CPUs, they provide extremely high computational and memory access throughput. GPUs also have higher performance-power ratio and lower cost. As many real world phenomena are intrinsically parallel, most scientific models are adaptable to these highly parallel processor architectures.

Several fast algorithms for field evaluations in the computational electromagnetic and micromagnetic simulations, targeting the GPGPUs are described and analyzed. Though built based on similar mathematical principles, the GPU version of these

algorithms adopt different approaches for data arrangement as well as the flow of the operations. After a substantial effort of redesigning, redistributing and rerouting the computational tasks, the achieved speed-ups over sequential versions are significant. Over 100x of speed-ups are common for all three algorithms, namely NGIM, B-AIM and FPIM, for problems started from several thousand of degrees of freedom. Moreover, the memory usage is merely 1% to 2% of the sequential version.

Built on the fast methods described in Chapter 4, a high-performance micromagnetic solver, FastMag are developed as a collaborative work. FastMag significantly extends the range of micromagnetic problems that researchers can solver using regular desktop computers. Ultra-large magnetic systems such as a complete recording head uniformly discretized are simulated, showing unobserved physical phenomena that might affect the future head design. FastMag is currently used by many internal and external users to simulate highly complex magnetic systems.

In chapter 7, two novel magnetic recording mechanisms are discussed. They aim at helping the next generation ultra-high density magnetic recording systems. The capped bit pattered media (CBPM) is thought to have better switching field distribution properties as well as lower reversal field and these hypotheses have been proven by simulations and further by follow-up experiments from other research groups. The microwave-assisted magnetic recording (MAMR) is a potentially revolutionary magnetic recording method. It is found, by computer simulations, that many factors, including

the geometric design of the media and the coupling strength might affect the reversal mechanism inside the recording media while exposed to high frequency microwaves.

8.2 Future directions

8.2.1 Further development of NGIM

The NGIM algorithm described in the Section 4.2 is highly efficient in handling scalar as well as vector fields generated by a randomly distributed set of sources. However, it still has room to be improved to meet the theoretically efficiency on handling extremely non-uniform source/observer distribution. Unbalanced tree structures are required to accelerate the computation of fields generated by highly non-uniform sources. This task might be more feasible on GPUs when the next generation NVIDIA's Kepler GK110 GPUs become available. The new "dynamic parallelism" technology is proposed to make the GPU handle multiple layers of parallelism in more efficient manner.

8.2.2 FastMag on GPUs

Even though the field evaluation, which is the most time consuming part of FastMag, has been successfully implemented on GPUs, currently FastMag still relies on CPUs to handle several essential tasks such as the time integration. Those once negligible computational processes now cost significant portion of overall time as the

original bottlenecks are already removed by GPUs. According to the famous Amdahl's Law [75], even if only 10% of the original sequential code are not efficiently parallelized, the upper limit of overall acceleration would be less than 10x. Therefore, porting as many operations as possible to GPUs is critical for further improvement of the efficiency of the FastMag simulator.

8.2.3 Parallelization across multiple computing nodes

The computational results presented in the thesis are obtained on a single computing node with single or multiple GPUs. Though NGIM and B-AIM are designed to be easily scalable to multiple computing nodes with multi-million or even billions of threads, multi-node parallel versions are still under development. One significant challenge that can be anticipated is the adverse impact of the much slower inter-node communication. Under these circumstances, exploring the opportunities of overlapping the communication and computation would be critical for further scaling the fast algorithms as well as the solvers as a whole.

NGIM is particularly attractive for multi-GPU systems, especially for high-frequency problems when only outgoing NG grids are contracted and each such grid is independent of each other at the same level. Moreover, the memory consumption is of $O\left(N\right)$ as only two levels are needed to be kept in memory at any moment. These properties can eliminate the need for any data exchange between different GPUs in a multi-node computing cluster, which is crucial for achieving high-performance.

The B-AIM is generally harder to be extended to multi-GPU systems. This is because the FFT used in B-AIM has relative low arithmetic density and are prone to slow inter-node communication.

Appendix A The big-O notation

The big-O notation is usually used in the computer science and specifically the algorithm design and analysis field to indicate the asymptotic behavior of a computer algorithm when the problem size grows large. The big-O notation is called the "asymptotic upper bound" and is defined as follows: [40]

For a given function $g(n)$, we denote by $f(x) = O(g(n))$ if and only if there is a positive constant c such that for all sufficiently large values of x that $|f(x)| \leq c|g(x)|$.

There are several points that worth mentioning for using this big-O notation.

a) The big-O notation does not tell the exact running time of any given algorithm. It only shows how the running time of a given algorithm, with all other conditions kept the same, responds to the change of the problem size

b) When write equation $f(x) = O(g(n))$ to reflect the statement "$f(x)$ is $O(g(n))$", we should be aware that it is an abuse of notation and this equation cannot be swapped. One simple example is that $O(x) = O(x^2)$ is true, but $O(x^2) = O(x)$ is not.

c) The asymptotic complexity of an algorithm is determined by the most "complex" part of it. For example, if a function $f(n)$ that represents the number of operations required to accomplish a certain task can be express as

$$f\left(n\right) = C_0 n + C_1 n^3 + C_2 n \log\left(n\right) \ , \tag{8.1}$$

The "big-O" of $f\left(n\right)$ would be $f\left(n\right) = O\left(n^3\right)$.

There are several other related but different notations that are commonly used by people to describe the complexity of an algorithm, such as the theta-notation and the omega-notation. They are used to represent the *asymptotically tight bound* and *asymptotic lower bound* of functions. Detailed explanation can be found in Ref. [40].

Appendix B Periodic Green's function

The PGFs for 1D, 2D, and 3D infinite arrays residing in free-space can be calculated via single, double, and triple spatial summations, respectively, over the integers i_x, i_y, and i_z, which represent the general index $\mathbf{i}$ in Section 5.2:

$$\begin{aligned} G^{1D}(\mathbf{r},0) &= \sum_{i_x=-\infty}^{\infty} \frac{e^{-jk_{x0}i_xL_x}e^{-jk|\mathbf{r}-i_xL_x\mathbf{x}|}}{4\pi \mid \mathbf{r}-i_xL_x\mathbf{x} \mid}, \\ G^{2D}(\mathbf{r},0) &= \sum_{i_x,i_y=-\infty}^{\infty} \frac{e^{-j\left(k_{x0}i_xL_x+k_{y0}i_yL_y\right)}e^{-jk|\mathbf{r}-i_xL_x\mathbf{x}-i_yL_y\mathbf{y}|}}{4\pi \mid \mathbf{r}-i_xL_x\mathbf{x}-i_yL_y\mathbf{y} \mid}, \\ G^{3D}(\mathbf{r},0) &= \sum_{i_x,i_y,i_z=-\infty}^{\infty} \frac{e^{-j\left(k_{x0}i_xL_x+k_{y0}i_yL_y+k_{z0}i_zL_z\right)}e^{-jk|\mathbf{r}-i_xL_x\mathbf{x}-i_yL_y\mathbf{y}-i_zL_z\mathbf{z}|}}{4\pi \mid \mathbf{r}-i_xL_x\mathbf{x}-i_yL_y\mathbf{y}-i_zL_z\mathbf{z} \mid}. \end{aligned} \tag{8.2}$$

Alternatively, the PGF's may be found via the following Floquet mode expansions

$$\begin{aligned} G^{1D}(\mathbf{r},0) &= \frac{1}{L_x}\sum_{m=-\infty}^{\infty} \frac{1}{4j}e^{-jk_{xm}x}H_0^{(2)}\left(k_{\rho m}\sqrt{y^2+z^2}\right), \\ G^{2D}(\mathbf{r},0) &= \frac{1}{L_xL_y}\sum_{m,n=-\infty}^{\infty} \frac{e^{-jk_{xm}x-jk_{yn}y-jk_{zmn}|z|}}{2jk_{zmn}}, \\ G^{3D}(\mathbf{r},0) &= \frac{1}{L_xL_y}\sum_{m,n=-\infty}^{\infty} \frac{e^{-jk_{xm}x-jk_{yn}y}}{2jk_{zmn}} \times \\ &\times\left(e^{-jk_{zmn}|z|}+\frac{e^{-j(k_{zmn}-k_{z0})L_z}e^{-jk_{zmn}z}}{(1-e^{-j(k_{zmn}-k_{z0})L_z})}+\frac{e^{-j(k_{z0}+k_{zmn})L_z}e^{jk_{zmn}z}}{(1-e^{-j(k_{zmn}+k_{z0})L_z})}\right), \end{aligned} \tag{8.3}$$

where $k_{xm} = k_{x0} + 2\pi m / L_x$, $k_{ym} = k_{y0} + 2\pi n / L_y$, $k_{\rho m} = \left(k^2 - {k_{xm}}^2\right)^{1/2}$, and $k_{zmn} = \left(k^2 - {k_{xm}}^2 - {k_{ym}}^2\right)^{1/2}$. The square root for $k_{\rho m}$ and k_{zmn} is defined with $\mathrm{Im}\left\{k_{\rho m}, k_{zmn}\right\} \le 0$ on the top Riemann sheet of the complex k_{x0} plane for most m; however, a certain (typically small) number of square roots can be defined on the lower Riemann sheet with $\mathrm{Im}\left\{k_{\rho m}\right\} > 0$. The expression for the 3D PGF is obtained from

that for the 2D PGF using $G^{3D}(\mathbf{r},0)=\sum_{i_z=-\infty}^{\infty} G^{2D}(\mathbf{r},i_z L_z \mathbf{z})e^{-jk_{z0}i_z L_z}$ with G^{2D} found by Floquet expansion.

These expressions are rapidly convergent provided z is not too small. Specifically, the number of terms required to achieve a prescribed error of ε is estimated as $L_x \log(\varepsilon^{-1})\big/(2\pi z)$ for 1D arrays and $L_x L_y \log^2(\varepsilon^{-1})\big/(2\pi z)^2$ for 2D and 3D arrays (for the case $L_x = L_y = L_z$).

References

[1] M. Albrecht, C. T. Rettner, A. Moser, M. E. Best, and B. D. Terris, "Recording performance of high-density patterned perpendicular magnetic media," *Applied Physics Letters,* vol. 81, no. 15, pp. 2875-2877, 2002.

[2] AMD Inc., *AMD Accelerated Parallel Processing OpenCL programming guide*, 2012.

[3] AMD Inc., *AMD Accelerated Processing Units*, 2012.

[4] AMD Inc., *Southern Islands series instruction set architecture*, 2012.

[5] P. L. Arlett, A. K. Bahrani, and O. C. Zienkiewicz, "Application of finite elements to the solution of Helmholtz's equation," *Proceedings of the Institution of Electrical Engineers,* vol. 115, no. 12, pp. 1762-1766, 1968.

[6] S. S. Baghsorkhi, I. Gelado, M. Delahaye, and W.-m. W. Hwu, "Efficient performance evaluation of memory hierarchy for highly multithreaded graphics processors," in *Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, New Orleans, Louisiana, USA, Year, pp. 23-34.

[7] J. M. Bahi, R. Couturier, and L. Z. Khodja, "Parallel GMRES implementation for solving sparse linear systems on GPU clusters," in *Proceedings of the 19th High Performance Computing Symposia*, Boston, Massachusetts, Year, pp. 12-19.

[8] J. Barnes, and P. Hut, "A hierarchical *O(NlogN)* force-calculation algorithm," *Nature,* vol. 324, no. 6096, pp. 446-449, 1986.

[9] J. Bedorf, E. Gaburov, and S. P. Zwart, "A sparse octree gravitational N-body code that runs entirely on the GPU processor," *Journal of Computational Physics,* vol. 231, no. 7, pp. 2825-2839, 2012.

[10] N. Bell, and M. Garland, "Implementing sparse matrix-vector multiplication on throughput-oriented processors," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, Portland, Oregon, Year, pp. 1-11.

[11] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz, "AIM: adaptive integral method for solving large-scale electromagnetic scattering and radiation problems," *Radio Science,* vol. 31, no. 5, pp. 1225-1251, 1996.

[12] D. Blythe, "Rise of the graphics processor," *Proceedings of the IEEE,* vol. 96, no. 5, pp. 761-778, 2008.

[13] A. Boag, and Y. Leviatan, "Analysis of electromagnetic scattering from linear periodic arrays of perfectly conducting bodies using a cylindrical-current model," *Antennas and Propagation, IEEE Transactions on,* vol. 39, no. 9, pp. 1332-1337, 1991.

[14] A. Boag, and B. Livshitz, "Adaptive nonuniform-grid (NG) algorithm for fast capacitance extraction," *IEEE Transactions on Microwave Theory and Techniques,* vol. 54, no. 9, pp. 3565-3570, 2006.

[15] A. Boag, V. Lomakin, and E. Michielssen, "Nonuniform grid time domain (NGTD) algorithm for fast evaluation of transient wave fields," *IEEE Transactions on Antennas and Propagation,* vol. 54, no. 7, pp. 1943-1951, 2006.

[16] A. Boag, E. Michielssen, and A. Brandt, "Nonuniform polar grid algorithm for fast field evaluation," *IEEE Antennas and Wireless Propagation Letters,* vol. 1, pp. 142-145, 2002.

[17] J. Bolz, I. Farmer, E. Grinspun, and P. Schrooder, "Sparse matrix solvers on the GPU: conjugate gradients and multigrid," *ACM Transactions on Graphics,* vol. 22, no. 3, pp. 917-924, 2003.

[18] K. Bondalapati, and V. K. Prasanna, "Reconfigurable computing systems," *Proceedings of the IEEE,* vol. 90, no. 7, pp. 1201-1217, 2002.

[19] C. Boone, J. A. Katine, E. E. Marinero, S. Pisana, and B. D. Terris, "Microwave-assisted magnetic reversal in perpendicular media," *IEEE Magnetics Letters,* vol. 3, 2012.

[20] A. Breitling, T. Bublat, and D. Goll, "Exchange-coupled L_{10}-FePt/Fe composite patterns with perpendicular magnetization," *physica status solidi (RRL) – Rapid Research Letters,* vol. 3, no. 5, pp. 130-132, 2009.

[21] A. R. Brodtkorb, C. Dyken, T. R. Hagen, J. M. Hjelmervik, and O. O. Storaasli, "State-of-the-art in heterogeneous computing," *Scientific Programming,* vol. 18, no. 1, pp. 1-33, 2010.

[22] W. F. Brown, Jr., *Micromagnetics*: Interscience Publishers, 1963.

[23] R. Buchty, V. Heuveline, W. Karl, and J.-P. Weiss, "A survey on hardware-aware and heterogeneous computing on multicore processors and accelerators," *Concurrency and Computation: Practice and Experience,* vol. 24, no. 7, pp. 663-675, 2012.

[24] M. Burtscher, and K. Pingali, "Chapter 6 - An efficient CUDA implementation of the tree-based Barnes Hut n-body algorithm," *GPU Computing Gems Emerald Edition*, W. H. Wen-mei, ed., pp. 75-92, Boston: Morgan Kaufmann, 2011.

[25] F. Casoli, F. Albertini, L. Nasi, S. Fabbrici, R. Cabassi, F. Bolzoni, C. Bocchi, and P. Luches, "Role of interface morphology in the exchange-spring behavior of FePt/Fe perpendicular bilayers," *Acta Materialia,* vol. 58, no. 10, pp. 3594-3601, 2010.

[26] M. F. Catedra, R. F. Torrs, J. Basterrechea, and E. Gago, *The CG-FFT Method: application of signal processing techniques to electromagnetics*, Norwood, MA: Artech House, Inc., 1994.

[27] F. T. Celepcikay, D. R. Wilton, D. R. Jackson, and F. Capolino, "Choosing splitting parameters and summation limits in the numerical evaluation of 1-D and 2-D periodic Green's funcstions with the Ewald method," *Radio Science,* vol. 43, pp. RS6S01, 2008.

[28] A. Cevahir, A. Nukada, and S. Matsuoka, "Fast conjugate gradients with multiple GPUs computational science – ICCS 2009," in, Year, pp. 893-903.

[29] A. Chandramowlishwaran, S. Williams, L. Oliker, I. Lashuk, G. Biros, and R. Vuduc, "Optimizing and tuning the fast multipole method for state-of-the-art multicore architectures," in *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, Year, pp. 1-12.

[30] R. Chang, M. A. Escobar, S. Li, M. V. Lubarda, and V. Lomakin, "Accurate evaluation of exchange fields in finite element micromagnetic solvers," *Journal of Applied Physics,* vol. 111, no. 7, pp. 07D129-123, 2012.

[31] R. Chang, S. Li, M. Lubarda, B. Livshitz, and V. Lomakin, "FastMag: Fast micromagnetic simulator for complex magnetic structures," *Journal of Applied Physics,* vol. 109, no. 7, pp. 07D358-307D358-356, 2011.

[32] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using CUDA," *Journal Parallel and Distributed Computing,* vol. 68, no. 10, pp. 1370-1380, 2008.

[33] Y. Chen, X. Cui, and H. Mei, "Large-scale FFT on GPU clusters," in *Proceedings of the 24th ACM International Conference on Supercomputing*, Tsukuba, Ibaraki, Japan, Year, pp. 315-324.

[34] H. W. Cheng, W. Y. Crutchfield, Z. Gimbutas, L. F. Greengard, J. F. Ethridge, J. F. Huang, V. Rokhlin, N. Yarvin, and J. S. Zhao, "A wideband fast multipole method for the Helmholtz equation in three dimensions," *Journal of Computational Physics,* vol. 216, no. 1, pp. 300-325, 2006.

[35] W. C. Chew, B. Hu, Y. C. Pan, and J. S. Zhao, "Fast algorithm for complex structures," in *Asia-Pacific Microwave Conference, 2001*, Year, pp. 75-78 vol.71.

[36] W. C. Chew, J.-M. Jin, C.-C. Lu, E. Michielssen, and J. M. Song, "Fast solution methods in electromagnetics," *Antennas and Propagation, IEEE Transactions on,* vol. 45, no. 3, pp. 533-543, 1997.

[37] S. Cohen, and C. Hindmarsh, "CVODE, A Stiff/nonstiff ODE Solver In C," *Computers in Physics,* vol. 10, no. 2, pp. 138-143, 1996.

[38] K. Compton, and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys,* vol. 34, no. 2, pp. 171-210, 2002.

[39] J. W. Cooley, and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation,* vol. 19, pp. 297-301, 1965.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*: MIT Press, 2009.

[41] M. Cwikla, J. Aronsson, and V. Okhmatovski, "Low-frequency MLFMA on graphics processors," *Antennas and Wireless Propagation Letters, IEEE,* vol. 9, pp. 8-11, 2010.

[42] C. P. da Silva, L. F. Cupertino, D. Chevitarese, M. A. C. Pacheco, and C. Bentes, "Exploring data streaming to improve 3D FFT implementation on multiple GPUs," in *Computer Architecture and High Performance Computing Workshops (SBAC-PADW), 2010 22nd International Symposium on*, Year, pp. 13-18.

[43] E. Darve, C. Cecka, and T. Takahashi, "The fast multipole method on parallel clusters, multicore processors, and graphics processing units," *Comptes Rendus Mécanique,* vol. 339, no. 2–3, pp. 185-193, 2011.

[44] D. De Donno, A. Esposito, G. Monti, and L. Tarricone, "Parallel efficient method of moments exploiting graphics processing units," *Microwave and Optical Technology Letters,* vol. 52, no. 11, pp. 2568-2572, 2010.

[45] D. De Donno, A. Esposito, G. Monti, and L. Tarricone, "Efficient acceleration of sparse MPIE/MoM with graphics processing units," in *41st European Microwave Conference (EuMC), 2011*, Year, pp. 175-178.

[46] R. Dittrich, T. Schrefl, D. Suess, W. Scholz, H. Forster, and J. Fidler, "A path method for finding energy barriers and minimum energy paths in complex micromagnetic systems," *Journal of Magnetism and Magnetic Materials,* vol. 250, no. 0, pp. 12-19, 2002.

[47] B. Duan, W. Wang, X. Li, C. Zhang, P. Zhang, and N. Sun, "Floating-point mixed-radix FFT core generation for FPGA and comparison with GPU and CPU," in *Field-Programmable Technology (FPT), 2011 International Conference on*, Year, pp. 1-6.

[48] J. W. Eastwood, R. W. Hockney, and D. N. Lawrence, "P3M3DP-the three-dimensional periodic particle-particle/particle-mesh program," *Computer Physics Communications,* vol. 35, no. 0, pp. C-618-C-619, 1984.

[49] M. A. Escobar, M. V. Lubarda, S. Li, R. Chang, B. Livshitz, and V. Lomakin, "Advanced micromagnetic analysis of write head dynamics using fastmag," *IEEE Transactions on Magnetics*, no. 99, pp. 1-1, 2012.

[50] M. A. Francavilla, E. A. Attardo, F. Vipiana, and G. Vecchi, "A GPU acceleration for FFT-based fast solvers for the integral equation," in *Proceedings of the Fourth European Conference on Antennas and Propagation (EuCAP), 2010*, Year, pp. 1-4.

[51] M. A. Francavilla, F. Vipiana, and G. Vecchi, "FFT-based solvers for the EFIE on graphics processors," in *IEEE Antennas and Propagation Society International Symposium (APSURSI), 2010*, Year, pp. 1-4.

[52] F. Franchetti, M. Puschel, Y. Voronenko, S. Chellappa, and J. M. F. Moura, "Discrete fourier transform on multicore," *IEEE Signal Processing Magazine,* vol. 26, no. 6, pp. 90-102, 2009.

[53] E. E. Fullerton, J. S. Jiang, M. Grimsditch, C. H. Sowers, and S. D. Bader, "Exchange-spring behavior in epitaxial hard/soft magnetic bilayers," *Physical Review B,* vol. 58, no. 18, pp. 12193-12200, 1998.

[54] N. Galoppo, N. K. Govindaraju, M. Henson, and D. Manocha, "LU-GPU: Efficient algorithms for solving dense linear systems on graphics hardware," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, Year, pp. 3.

[55] P. Gepner, and M. F. Kowalik, "Multi-core processors: New way to achieve high system performance," in *Parallel Computing in Electrical Engineering, 2006. PAR ELEC 2006. International Symposium on*, Year, pp. 9-13.

[56] T. L. Gilbert, "A phenomenological theory of damping in ferromagnetic materials," *IEEE Transactions on Magnetics,* vol. 40, no. 6, pp. 3443-3449, 2004.

[57] T. L. Gilbert, and J. M. Kelly, "Anomalous rotational damping in ferromagnetic sheets," in *Magnetism and Magnetic Materials Conference*, Pittsburgh, PA, Year, pp. 253 - 263.

[58] D. Goll, and S. Macke, "Thermal stability of ledge-type L_{10}-FePt/Fe exchange-spring nanocomposites for ultrahigh recording densities," *Applied Physics Letters,* vol. 93, no. 15, pp. 152512-152513, 2008.

[59] N. K. Govindaraju, B. Lloyd, Y. Dotsenko, B. Smith, and J. Manferdelli, "High performance discrete Fourier transforms on graphics processors," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, Austin, Texas, Year, pp. 1-12.

[60] A. Greenbaum, *Iterative methods for solving linear systems*, Philadelphia, PA: Society of Industrial and Applied Mathematics, 1987.

[61] L. Greengard, and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of Computational Physics,* vol. 73, no. 2, pp. 325-348, 1987.

[62] N. A. Gumerov, and R. Duraiswami, "Fast multipole methods on graphics processors," *Journal of Computational Physics,* vol. 227, no. 18, pp. 8290-8313, 2008.

[63] E. Gutierrez, S. Romero, M. Trenas, and E. Zapata, "Memory locality exploitation strategies for FFT on the CUDA architecture high performance computing for computational science - VECPAR 2008," in, Year, pp. 430-443.

[64] W. Hackbusch, and B. Khoromskij, "A sparse matrix arithmetic based on H-matrices. Part I: Introduction to H-matrices," *Computing,* vol. 62, pp. 89-108, 1999.

[65] W. Hackbusch, and B. Khoromskij, "A sparse H-matrix arithmetic. Part II: Application to multi-dimensional problems," *Computing,* vol. 64, no. 21 - 47, 2000.

[66] T. Hamada, T. Narumi, R. Yokota, K. Yasuoka, K. Nitadori, and M. Taiji, "42 TFlops hierarchical N-body simulations on GPUs with applications in both astrophysics and turbulence," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, Portland, Oregon, Year, pp. 1-12.

[67] D. J. Hardy, J. E. Stone, K. L. Vandivort, D. Gohara, C. Rodrigues, and K. Schulten, "Chapter 4 - Fast molecular electrostatics algorithms on GPUs," *GPU Computing Gems Emerald Edition*, W. H. Wen-mei, ed., pp. 43-58, Boston: Morgan Kaufmann, 2011.

[68] R. F. Harrington, *Field Computation by Moment Methods*: Wiley-IEEE Press, 1993.

[69] M. J. Harris, "Real-time cloud simulation and rendering", University of North Carolina at Chapel Hill, Chapel Hill, 2003.

[70] M. J. Harris, G. Coombe, T. Scheuermann, and A. Lastra, "Physically-based visual simulation on graphics hardware," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Saarbrucken, Germany, Year, pp. 109-118.

[71] R. Helfenstein, and J. Koko, "Parallel preconditioned conjugate gradient algorithm on GPU," *Journal of Computational and Applied Mathematics,* vol. 236, no. 15, pp. 3584-3590, 2012.

[72] O. Hellwig, A. Berger, T. Thomson, E. Dobisz, Z. Z. Bandic, H. Yang, D. S. Kercher, and E. E. Fullerton, "Separating dipolar broadening from the intrinsic switching field distribution in perpendicular patterned media," *Applied Physics Letters,* vol. 90, no. 16, pp. 162516-162513, 2007.

[73] G. Henkelman, and H. Jonsson, "Improved tangent estimate in the nudged elastic band method for finding minimum energy paths and saddle points," *Journal of Chemical Physics,* vol. 113, no. 22, pp. 9978-9985, 2000.

[74] J. Hennessy, and D. Patterson, *Computer Architecture: A Quantitative Approach, 5th Edition*, Waltham, MA: Morgan Kaufmann, 2011.

[75] M. D. Hill, and M. R. Marty, "Amdahl's Law in the multicore era," *Computer,* vol. 41, no. 7, pp. 33-38, 2008.

[76] D. Houssameddine, U. Ebels, B. Delaet, B. Rodmacq, I. Firastrau, F. Ponthenier, M. Brunet, C. Thirion, J. P. Michel, L. Prejbeanu-Buda, M. C. Cyrille, O. Redon, and B. Dieny, "Spin-torque oscillator using a perpendicular polarizer and a planar free layer," *Nature Materials,* vol. 6, no. 6, pp. 447-453, 2007.

[77] Q. Hu, N. A. Gumerov, and R. Duraiswami, "Scalable fast multipole methods on distributed heterogeneous architectures," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, Seattle, Washington, Year, pp. 1-12.

[78] W.-m. Hwu, K. Keutzer, and T. G. Mattson, "The concurrency challenge," *IEEE Design & Test of Computers,* vol. 25, no. 4, pp. 312-320, 2008.

[79] Intel, "Many Integrated Core (MIC) Architecture - Advanced," *http://www.intel.com/content/www/us/en/architecture-and-technology/many-integrated-core/intel-many-integrated-core-architecture.html,* vol. 2012, no. 8/19, 2012.

[80] P. Jetley, L. Wesolowski, F. Gioachin, Kale, x, L. V., and T. R. Quinn, "Scaling hierarchical n-body simulations on GPU clusters," in *High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for*, Year, pp. 1-11.

[81] J.-m. Jin, *The Finite Element Method in Electromagnetics*, Second Edition ed.: Wiley-IEEE Press, 2002.

[82] Khronos Group, "The OpenCL Specification - Khronos Group," 2011.

[83] G. Kobidze, B. Shanker, and D. P. Nyquist, "Efficient integral-equation-based method for accurate analysis of scattering from periodically arranged nanostructures," *Physical Review E,* vol. 72, no. 5, pp. 056702, 2005.

[84] C. Kraemer, N. Nikseresht, J. O. Piatek, N. Tsyrulin, B. D. Piazza, K. Kiefer, B. Klemke, T. F. Rosenbaum, G. Aeppli, C. Gannarelli, K. Prokes, A. Podlesnyak, T. Strässle, L. Keller, O. Zaharko, K. W. Krämer, and H. M.

Rønnow, "Dipolar antiferromagnetism and quantum criticality in $LiErF_4$," *Science,* vol. 336, no. 6087, pp. 1416-1419, 2012.

[85] K. S. Kunz, and R. J. Luebbers, *The Finite Difference Time Domain Method for Electromagnetics*, pp. 464 Pages: CRC Press, 1993.

[86] L. D. Landau, and E. M. Lifshitz, "On the theory of the dispersion of magnetic permeability in ferromagnetic bodies," *Phys. Z. Sowietunion,* vol. 8, no. 153, 1935.

[87] I. Lashuk, A. Chandramowlishwaran, H. Langston, T.-A. Nguyen, R. Sampath, A. Shringarpure, R. Vuduc, L. Ying, D. Zorin, and G. Biros, "A massively parallel adaptive fast-multipole method on heterogeneous architectures," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, Portland, Oregon, Year, pp. 1-12.

[88] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, and P. Dubey, "Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU," *ACM SIGARCH Computer Architecture News,* vol. 38, no. 3, pp. 451-460, 2010.

[89] E. Lezar, "GPU acceleration of matrix-based methods in computational electromagnetics", Stellenbosch University, Stellenbosch, 2011.

[90] E. Lezar, and D. B. Davidson, "GPU-Accelerated method of moments by example: monostatic scattering," *Antennas and Propagation Magazine, IEEE,* vol. 52, no. 6, pp. 120-135, 2010.

[91] E. Lezar, and D. B. Davidson, "GPU-based LU decomposition for large method of moments problems," *Electronics Letters,* vol. 46, no. 17, pp. 1194-1196, 2010.

[92] E. Lezar, and D. B. Davidson, "GPU acceleration of electromagnetic scattering analysis using the method of moments," in *Electromagnetics in Advanced Applications (ICEAA), 2011 International Conference on*, Year, pp. 452-455.

[93] S. Li, R. Chang, and V. Lomakin, "Chapter 19 - Fast electromagnetic integral equation solvers on graphics processing units," *GPU Computing Gems Jade Edition*, W. H. Wen-mei, ed., pp. 243-266, Boston: Morgan Kaufmann, 2012.

[94] S. Li, R. Chang, and V. Lomakin, "Fast integral equation solvers on Graphics Processing Units for Electromagnetics," *Ieee Antennas and Propagation Magazine*, to appear in 2013.

[95] S. Li, B. Livshitz, H. N. Bertram, E. E. Fullerton, and V. Lomakin, "Microwave-assisted magnetization reversal and multilevel recording in composite media," *Journal of Applied Physics,* vol. 105, no. 7, pp. 07B909-907B909-903, 2009.

[96] S. Li, B. Livshitz, H. N. Bertram, A. Inomata, E. E. Fullerton, and V. Lomakin, "Capped bit patterned media for high density magnetic recording," *Journal of Applied Physics,* vol. 105, no. 7, pp. 07C121-107C121-123, 2009.

[97] S. Li, B. Livshitz, H. N. Bertram, M. Schabes, T. Schrefl, E. E. Fullerton, and V. Lomakin, "Microwave assisted magnetization reversal in composite media," *Applied Physics Letters,* vol. 94, pp. 202509, 2009.

[98] S. Li, B. Livshitz, and V. Lomakin, "Fast evaluation of Helmholtz potential on graphics processing units (GPUs)," *Journal of Computational Physics,* vol. 229, no. 22, pp. 8463-8483, 2010.

[99] S. Li, B. Livshitz, and V. Lomakin, "Graphics Processing Unit Accelerated O(N) Micromagnetic Solver " *IEEE Transactions on Magnetics,* vol. 46, no. 6, pp. 2373-2375, 2010.

[100] S. Li, D. A. Van Orden, and V. Lomakin, "Fast periodic interpolation method for periodic unit cell problems," *Antennas and Propagation, IEEE Transactions on,* vol. 58, no. 12, pp. 4005-4014, 2010.

[101] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "NVIDIA Tesla: a unified graphics and computing architecture," *IEEE Micro,* vol. 28, no. 2, pp. 39-55, 2008.

[102] B. Livshitz, A. Boag, H. N. Bertram, and V. Lomakin, "Non-uniform grid algorithm for fast magnetostatic interactions calculation in micromagnetics," *Journal of Applied Physics,* vol. 105, 2009.

[103] B. Livshitz, A. Boag, H. N. Bertram, and V. Lomakin, "Nonuniform grid algorithm for fast calculation of magnetostatic interactions in micromagnetics," in *Proceedings of the 53rd Annual Conference on Magnetism and Magnetic Materials*, Austin, Texas (USA), Year, pp. 07D541-543.

[104] B. Livshitz, R. Choi, A. Inomata, H. N. Bertram, and V. Lomakin, "Fast precessional reversal in perpendicular composite patterned media," *Journal of Applied Physics,* vol. 103, no. 7, pp. 07C516-513, 2008.

[105] B. Livshitz, A. Inomata, H. N. Bertram, and V. Lomakin, "Precessional reversal in exchange-coupled composite magnetic elements," *Applied Physics Letters,* vol. 91, no. 18, pp. 182502-182503, 2007.

[106] V. Lomakin, R. Choi, B. Livshitz, S. Li, A. Inomata, and H. N. Bertram, "Dual-layer patterned media "ledge" design for ultrahigh density magnetic recording," *Applied Physics Letters,* vol. 92, no. 2, pp. 022502-022503, 2008.

[107] V. Lomakin, S. Li, B. Livshitz, A. Inomata, and H. N. Bertram, "Patterned media for 10 Tb/in^2 utilizing dual-section "ledge" elements," *IEEE Transactions on Magnetics,* vol. 44, no. 11, pp. 3454-3459, 2008.

[108] M. López-Portugués, J. A. López-Fernández, J. Ranilla, R. G. Ayestarán, and F. Las-Heras, "Parallelization of the FMM on distributed-memory GPGPU systems for acoustic-scattering prediction," *The Journal of Supercomputing*, pp. 1-11, 2012.

[109] I. Lorentz, M. Malita, and R. Andonie, "Fitting FFT onto an energy efficient massively parallel architecture," in *Proceedings of the Second International Forum on Next-Generation Multicore/Manycore Technologies*, Saint-Malo, France, Year, pp. 1-11.

[110] C. C. Lu, and W. C. Chew, "Fast algorithm for solving hybrid integral equations [EM wave scattering]," *Microwaves, Antennas and Propagation, IEE Proceedings H,* vol. 140, no. 6, pp. 455-460, 1993.

[111] M. Lubarda, M. Escobar, S. Li, R. Chang, E. Fullerton, and V. Lomakin, "Domain wall motion in magnetically frustrated nanorings," *Physical Review B,* vol. 85, no. 21, pp. 214428, 2012.

[112] M. Lubarda, S. Li, B. Livshitz, E. Fullerton, and V. Lomakin, "Reversal in Bit Patterned Media With Vertical and Lateral Exchange," *IEEE Transactions on Magnetics,* vol. 47, no. 1, 2011.

[113] M. V. Lubarda, S. Li, B. Livshitz, E. E. Fullerton, and V. Lomakin, "Antiferromagnetically coupled capped bit patterned media for high-density magnetic recording," *Applied Physics Letters,* vol. 98, no. 1, pp. 012513-012513-012513, 2011.

[114] J. C. Maxwell, "A dynamical theory of the electromagnetic field," *Philosophical Transactions of the Royal Society of London,* vol. 155, pp. 459-512, 1865.

[115] T. W. McDaniel, W. A. Challener, and K. Sendur, "Issues in heat-assisted perpendicular recording," *IEEE Transactions on Magnetics,* vol. 39, no. 4, pp. 1972-1979, 2003.

[116] S. A. McKee, "Reflections on the memory wall," in *Proceedings of the 1st conference on Computing frontiers*, Ischia, Italy, Year, pp. 162.

[117] Microsoft, "C++ AMP : Language and Programming Model," 2012.

[118] R. Mittra, "Integral equation methods for transient scattering," *Transient Electromagnetic Fields*, L. B. Felsen, ed., New York: Springer Verlag, 1976.

[119] K. Moreland, and E. Angel, "The FFT on a GPU," in *Proceedings of SIGGRAPH/Eurographics Workshop on Graphics Hardware*, San Diego, California, Year, pp. 112-119.

[120] A. Moroz, "Quasi-periodic Green's functions of the Helmholtz and Laplace equations," *Journal of Physics A,* vol. 39, no. 36, 2006.

[121] A. Munshi, B. Gaster, T. G. Mattson, J. Fung, and D. Ginsburg, *OpenCL Programming Guide*: Pearson Education, 2012.

[122] H. Muraoka, Y. Sonobe, K. Miura, A. M. Goodman, and Y. Nakamura, "Analysis on magnetization transition of CGC perpendicular media," *IEEE Transactions on Magnetics,* vol. 38, no. 4, pp. 1632-1636, 2002.

[123] A. Nukada, and S. Matsuoka, "Auto-tuning 3-D FFT library for CUDA GPUs," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, Portland, Oregon, Year, pp. 1-10.

[124] A. Nukada, Y. Ogata, T. Endo, and S. Matsuoka, "Bandwidth intensive 3-D FFT kernel for GPUs using CUDA," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, Austin, Texas, Year, pp. 1-11.

[125] NVIDIA Corporation, *NVIDIA's Next Generation CUDA Compute Architecture: Fermi*, v1.1 ed., 2010.

[126] NVIDIA Corporation, *CUDA C Best Practice Guide*, 2012.

[127] NVIDIA Corporation, *CUDA Compute Unified Device Architecture Programming Guide, V4.2*, 2012.

[128] NVIDIA Corporation, *CUFFT Library V4.2*, 2012.

[129] NVIDIA Corporation, *GPU Computing SDK 4.2*, 2012.

[130] NVIDIA Corporation, "NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110 - The Fastest, Most Efficient HPC Architecture Ever Built," pp. 24, 2012.

[131] OpenACC-Standard.org, "The OpenACC™ Application Programming Interface," 2011.

[132] S. Peng, and Z. Nie, "Acceleration of the method of moments calculations by using graphics processing units," *Antennas and Propagation, IEEE Transactions on,* vol. 56, no. 7, pp. 2130-2133, 2008.

[133] S. Peng, and C.-F. Wang, "Hardware accelerated MoM-PFFT method using graphics processing units," in *Antennas and Propagation (APSURSI), 2011 IEEE International Symposium on*, Year, pp. 3152-3153.

[134] J. Persson, Y. Zhou, and J. Akerman, "Phase-locked spin torque oscillators: Impact of device variability and time delay," *Journal of Applied Physics,* vol. 101, no. 9, pp. 09A503-503, 2007.

[135] A. F. Peterson, S. L. Ray, and R. Mittra, *Computational Methods for Electromagnetics*, New York: IEEE Press, 1998.

[136] J. R. Phillips, and J. K. White, "A precorrected-FFT method for electrostatic analysis of complicated 3-D structures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 16, no. 10, pp. 1059-1072, 1997.

[137] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes*: Cambridge University Press, 2007.

[138] M. Ranjbar, S. N. Piramanayagam, R. Sbiaa, and T. C. Chong, "Magnetic properties of antidots in conventional and spin-reoriented antiferro-magnetically coupled layers," *Journal of Applied Physics,* vol. 111, no. 7, pp. 07B921-923, 2012.

[139] M. Ranjbar, S. N. Piramanayagam, S. K. Wong, R. Sbiaa, and T. C. Chong, "Anomalous Hall effect measurements on capped bit-patterned media," *Applied Physics Letters,* vol. 99, no. 14, pp. 142503-142503, 2011.

[140] S. Rao, D. Wilton, and A. Glisson, "Electromagnetic scattering by surfaces of arbitrary shape," *Antennas and Propagation, IEEE Transactions on,* vol. 30, no. 3, pp. 409-418, 1982.

[141] S. M. Rao, and D. R. Wilton, "Transient scattering by conducting surfaces of arbitrary shape," *IEEE Transactions on Antennas and Propagation,* vol. 39, pp. 56-61, 1991.

[142] H. J. Richter, "The transition from longitudinal to perpendicular recording," *Journal of Physics D: Applied Physics,* vol. 40, no. 9, pp. R149, 2007.

[143] V. Rokhlin, "Rapid solution of integral equations of scattering theory in two dimensions," *Journal of Computational Physics,* vol. 86, no. 2, pp. 414-439, 1990.

[144] S. Romero, M. A. Trenas, E. Gutierrez, and E. L. Zapata, "Locality-improved FFT implementation on a graphics processor," in *Proceedings of the 7th WSEAS International Conference on Signal Processing, Computational Geometry & Artificial Vision*, Athens, Greece, Year, pp. 58-63.

[145] R. E. Rottmayer, S. Batra, D. Buechel, W. A. Challener, J. Hohlfeld, Y. Kubota, L. Lei, L. Bin, C. Mihalcea, K. Mountfield, K. Pelhos, P. Chubing, T. Rausch, M. A. Seigler, D. Weller, and Y. XiaoMin, "Heat-assisted magnetic recording," *IEEE Transactions on Magnetics,* vol. 42, no. 10, pp. 2417-2421, 2006.

[146] M. Rumpf, and R. Strzodka, "Nonlinear diffusion in graphics hardware," *Proceedings of EG/IEEE TCVG Symposium on Visualization (VisSym '01)*, pp. 75-84, 2001.

[147] Y. Saad, *Iterative methods for sparse linear systems*, Philadelphia, PA: Society for Industrial and Applied Mathematics., 2003.

[148] Y. Saad, and M. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing,* vol. 7, no. 3, pp. 856-869, 1986.

[149] W. Scholz, and S. Batra, "Micromagnetic modeling of ferromagnetic resonance assisted switching," *Journal of Applied Physics,* vol. 103, no. 7, pp. 07F539-533, 2008.

[150] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan, "Larrabee: a many-core x86 architecture for visual computing," *ACM Transactions on Graphics,* vol. 27, no. 3, pp. 1-15, 2008.

[151] V. Skumryev, S. Stoyanov, Y. Zhang, G. Hadjipanayis, D. Givord, and J. Nogues, "Beating the superparamagnetic limit with exchange bias," *Nature,* vol. 423, no. 6942, pp. 850-853, 2003.

[152] J. M. Song, and W. C. Chew, "Multilevel fast-multipole algorithm for solving combined field integral equations of electromagnetic scattering," *Microwave and Optical Technology Letters,* vol. 10, no. 1, pp. 14-19, 1995.

[153] Y. Sonobe, K. K. Tham, T. Umezawa, C. Takasu, J. A. Dumaya, and P. Y. Leo, "Effect of continuous layer in CGC perpendicular recording media," *Journal of Magnetism and Magnetic Materials,* vol. 303, no. 2, pp. 292-295, 2006.

[154] G. Stantchev, W. Dorland, and N. Gumerov, "Fast parallel Particle-To-Grid interpolation for plasma PIC simulations on the GPU," *Journal of Parallel and Distributed Computing,* vol. 68, no. 10, pp. 1339-1349, 2008.

[155] J. E. Stone, J. C. Phillips, P. L. Freddolino, D. J. Hardy, L. G. Trabuco, and K. Schulten, "Accelerating molecular modeling applications with graphics processors," *Journal of Computational Chemistry,* vol. 28, no. 16, pp. 2618-2640, 2007.

[156] D. Suess, T. Schrefl, S. Fahler, M. Kirschner, G. Hrkac, F. Dorfbauer, and J. Fidler, "Exchange spring media for perpendicular recording," *Applied Physics Letters,* vol. 87, no. 1, pp. 012504-012503, 2005.

[157] Z. Z. Sun, and X. R. Wang, "Magnetization reversal through synchronization with a microwave," *Physical Review B,* vol. 74, no. 13, pp. 132401, 2006.

[158] C. Thirion, W. Wernsdorfer, and D. Mailly, "Switching of magnetization by nonlinear resonance studied in single nanoparticles," *Nature Materials,* vol. 2, no. 8, pp. 524-527, 2003.

[159] D. A. Thompson, and J. S. Best, "The future of data storage technology," *IBM Journal of Research and Development,* vol. 44, pp. 311-322, 2000.

[160] T. Topa, A. Noga, and A. Karwowski, "Adapting MoM with RWG basis functions to GPU technology using CUDA," *Antennas and Wireless Propagation Letters, IEEE,* vol. 10, pp. 480-483, 2011.

[161] A. Y. Toukmaji, and J. A. Board Jr, "Ewald summation techniques in perspective: a survey," *Computer Physics Communications,* vol. 95, no. 2–3, pp. 73-92, 1996.

[162] G. Valerio, P. Baccarelli, P. Burghignoli, and A. Galli, "Comparative analysis of acceleration techniques for 2-D and 3-D Green's functions in periodic structures along one and two directions," *IEEE Transactions on Antennas and Propagation,* vol. 55, no. 6, pp. 1630 - 1643 2007.

[163] G. Valerio, P. Baccarelli, S. Paulotto, F. Frezza, and A. Galli, "Regularization of mixed-potential layered-media Green's functions for efficient interpolation procedures in planar periodic structures," *IEEE Transactions on Antennas and Propagation,* vol. 57, no. 1, pp. 122-134, 2009.

[164] D. Van Orden, and V. Lomakin, "Rapidly convergent representations for 2D and 3D Green's functions for a linear periodic array of dipole sources," *Antennas and Propagation, IEEE Transactions on,* vol. 57, no. 7, pp. 1973-1984, 2009.

[165] S. Velamparambil, S. MacKinnon-Cormier, J. Perry, R. Lemos, M. Okoniewski, and J. Leon, "GPU accelerated Krylov subspace methods for computational electromagnetics," in *Microwave Conference, 2008. EuMC 2008. 38th European*, Year, pp. 1312-1314.

[166] R. H. Victora, and S. Xiao, "Composite media for perpendicular magnetic recording," *IEEE Transactions on Magnetics,* vol. 41, no. 2, pp. 537-542, 2005.

[167] V. Volkov. "Better performance at lower occupany", http://www.cs.berkeley.edu/~volkov/volkov10-GTC.pdf, accessed on 08/25/2012;

[168] V. Volkov, and J. W. Demmel, "Benchmarking GPUs to tune dense linear algebra," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, Year, pp. 1-11.

[169] D. Weller, and A. Moser, "Thermal effect limits in ultrahigh-density magnetic recording," *IEEE Transactions on Magnetics,* vol. 35, no. 6, pp. 4423-4439, 1999.

[170] G. Winkler, D. Suess, J. Lee, J. Fidler, M. A. Bashir, J. Dean, A. Goncharov, G. Hrkac, S. Bance, and T. Shrefl, "Microwave-assisted three-dimensional multilayer magnetic recording," *Applied Physics Letters,* vol. 94, no. 23, pp. 232501-232503, 2009.

[171] G. Woltersdorf, and C. H. Back, "Microwave assisted switching of single domain $Ni_{80}Fe_{20}$ elements," *Physical Review Letters,* vol. 99, no. 22, pp. 227207, 2007.

[172] H. Wong, M. M. Papadopoulou, M. Sadooghi-Alvandi, and A. Moshovos, "Demystifying GPU microarchitecture through microbenchmarking," in

Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on, Year, pp. 235-246.

[173] R. W. Wood, "XLII. On a remarkable case of uneven distribution of light in a diffraction grating spectrum," *Philosophical Magazine,* vol. 4, no. 21, 1902.

[174] K. Yang, and A. E. Yilmaz, "Comparison of precorrected FFT/adaptive integral method matching schemes," *Microwave and Optical Technology Letters,* vol. 53, no. 6, pp. 1368-1372, 2011.

[175] K. Yee, "Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media," *Antennas and Propagation, IEEE Transactions on,* vol. 14, no. 3, pp. 302-307, 1966.

[176] D. Yeh, L.-S. Peh, S. Borkar, J. Darringer, A. Agarwal, and W.-m. Hwu, "Thousand core chips," *IEEE Design & Test of Computers,* vol. 25, no. 3, pp. 272-278, 2008.

[177] A. E. Yilmaz, J. Jian-Ming, and E. Michielssen, "Time domain adaptive integral method for surface integral equations," *IEEE Transactions on Antennas and Propagation,* vol. 52, no. 10, pp. 2692-2708, 2004.

[178] R. Yokota, and L. A. Barba, "Chapter 9 - Treecode and Fast Multipole Method for n-body simulation with CUDA," *GPU Computing Gems Emerald Edition*, W. H. Wen-mei, ed., pp. 113-132, Boston: Morgan Kaufmann, 2011.

[179] R. Yokota, J. P. Bardhan, M. G. Knepley, L. A. Barba, and T. Hamada, "Biomolecular electrostatics using a fast multipole BEM on up to 512 GPUs and a billion unknowns," *Computer Physics Communications,* vol. 182, no. 6, pp. 1272-1283, 2011.

[180] J.-G. Zhu, and Y. Wang, "Microwave assisted magnetic recording utilizing perpendicular spin torque oscillator with switchable perpendicular electrodes," *IEEE Transactions on Magnetics,* vol. 46, no. 3, pp. 751-757, 2010.

[181] J.-G. Zhu, X. Zhu, and Y. Tang, "Microwave assisted magnetic recording," *IEEE Transactions on Magnetics,* vol. 44, no. 1, pp. 125-131, 2008.