December 2014

The Unified-FFT Method for Fast Solution of Integral Equations as Applied to Shielded-Domain Electromagnetics

Brian J. Rautio
Syracuse University

Abstract

Electromagnetic (EM) solvers are widely used within computer-aided design (CAD) to improve and ensure success of circuit designs. Unfortunately, due to the complexity of Maxwell's equations, they are often computationally expensive. While considerable progress has been made in the realm of speed-enhanced EM solvers, these fast solvers generally achieve their results through methods that introduce additional error components by way of geometric approximations, sparse-matrix approximations, multilevel decomposition of interactions, and more. This work introduces the new method, Unified-FFT (UFFT). A derivative of method of moments, UFFT scales as $O(N \log N)$, and achieves fast analysis by the unique combination of FFT-enhanced matrix fill operations (MFO) with FFT-enhanced matrix solve operations (MSO).

In this work, two versions of UFFT are developed, UFFT-Precorrected (UFFT-P) and UFFT-Grid Totalizing (UFFT-GT). UFFT-P uses precorrected FFT for MSO and allows the use of basis functions that do not conform to a regular grid. UFFT-GT uses conjugate gradient FFT for MSO and features the capability of reducing the error of the solution down to machine precision. The main contribution of UFFT-P is a fast solver, which utilizes FFT for both MFO and MSO. It is demonstrated in this work to not only provide simulation results for large problems considerably faster than state of the art commercial tools, but also to be capable of simulating geometries which are too complex for conventional simulation. In UFFT-P these benefits come at the expense of a minor penalty to accuracy.

UFFT-GT contains further contributions as it demonstrates that such a fast solver can be accurate to numerical precision as compared to a full, direct analysis. It is shown to provide even more algorithmic efficiency and faster performance than UFFT-P. UFFT-GT makes an additional contribution in that it is developed not only for planar geometries, but also for the case of multilayered dielectrics and metallization. This functionality is particularly useful for multi-layered printed circuit boards (PCBs) and integrated circuits (ICs). Finally, UFFT-GT contributes a 3D planar solver, which allows for current to be discretized in the *z*-direction. This allows for similar fast and accurate simulation with the inclusion of some 3D features, such as vias connecting metallization planes.

The Unified-FFT Method for Fast Solution of Integral Equations as Applied to Shielded-Domain Electromagnetics

by
Brian Rautio

B.S.E.E., Rensselaer Polytechnic Institute, May 2009
M.S.E.E., Syracuse University, July 2011

Dissertation
Submitted in partial fulfillment for the degree of Doctor of Philosophy in Electrical and Computer Engineering in the Graduate School of Syracuse University

December 2014

Acknowledgements

The author would, first and foremost, like to thank his girlfriend, Ms. Patricia Leusner not only for all of her support, but also for bearing with his vast workload, idiosyncrasies, and meltdowns during this project. His parents, Dr. Jim and Mrs. Jean Rautio for giving him the resources and encouragement to get this far. His advisors, Prof. Jay Kyoon Lee and Prof. Vladimir Okhmatovski, for answering endless questions, for reviewing countless documents, and for putting up with just a bit too much work being done in the 11th Hour. The committee members, Prof. Thong Quoc Dang, Prof. Jun H. Choi, Prof. Tomislav Bujanovic, and Prof. Prasanta K. Ghosh, for their time, effort, and willingness to help. Mr. Dan Ferguson and Mr. Matt Thelen of Sonnet Software, for their hard work adding the necessary hooks and data outputs into Sonnet to make this work possible. The anonymous MTT Transactions reviewers—the detailed process absolutely made for better results. And finally, Mr. Petey Peterson, the author's tabby cat, for all the times he made good company whilst working far into the night.

Table of Contents

List of Figures

List of Tables

1. Introduction

Numerical methods within electromagnetics have existed for several decades [1], even in commercial form [2]. They are often used for microwave filter design, antenna design, electromagnetic compatibility checking, and many others. Unfortunately, their usefulness is often limited by the performance of the algorithm. As they generally fill and solve a dense system of equations, such techniques may scale poorly. For example Method of Moments (MoM) often scales in terms of operations as $O(N^3)$ and in terms of memory as $O(N^2)$. In order to extend the usefulness of numerical methods within electromagnetics, a number of better-scaling algorithms have been developed, however there is still great commercial need for more specific and powerful algorithms in various applications. This work investigates accelerated solution of integral equations by enhancing two of the most cumbersome components of such techniques, Matrix Fill Operations (MFO) and Matrix Solve Operations (MSO), with Fast Fourier Transforms (FFT). It is applied to create a Galerkin Method of Moments (MoM) numerical electromagnetics algorithm, which reduces the complexity of operations from the often prohibitively high $O(N^3)$, down to $O(N \log N)$. As it is the first to combine FFT usage for both MFO and MSO, the new solver is termed Unified-FFT (UFFT). Two variants of UFFT are introduced, UFFT-Precorrected (UFFT-P) and UFFT-Grid Totalizing (UFFT-GT). The first variant, UFFT-P uses the pre-corrected FFT (PFFT) algorithm [3] for MSO, which includes a geometry projection to allow for basis functions that do not conform to a regular grid. The second variant, UFFT-GT features the conjugate-gradient FFT (CG-FFT) [4] algorithm for MSO, and uses a single grid for both MFO and MSO. This allows for fast calculation near to the numerical precision of the

computer. This chapter establishes some general principles within modeling of electromagnetics and then other principles more specific to computational electromagnetics. Additionally, objectives and contributions of the work are identified.

1.1 Electromagnetics Modeling Overview

Electrical, electronics, and computer engineering have benefited from modeling since their inception as fields. Modeling is often done analytically, numerically, and by measurement, and generally is inclusive of electrical properties such as circuit theory, full-wave electromagnetics, and optics, depending on desired frequency range and available resources. Increasingly, physical factors such as heat dissipation, thermal expansion, and even fluid flow are being modeled in relation to electronic circuits [5].

This work focuses on accelerated modeling of time harmonic (i.e., sinusoidal wave), full-wave electromagnetic behavior of circuits that are some significant fraction or some small multiple of wavelength in size. There are a number of methods with which these problems can be solved, as depicted in Fig. 1.1-1.

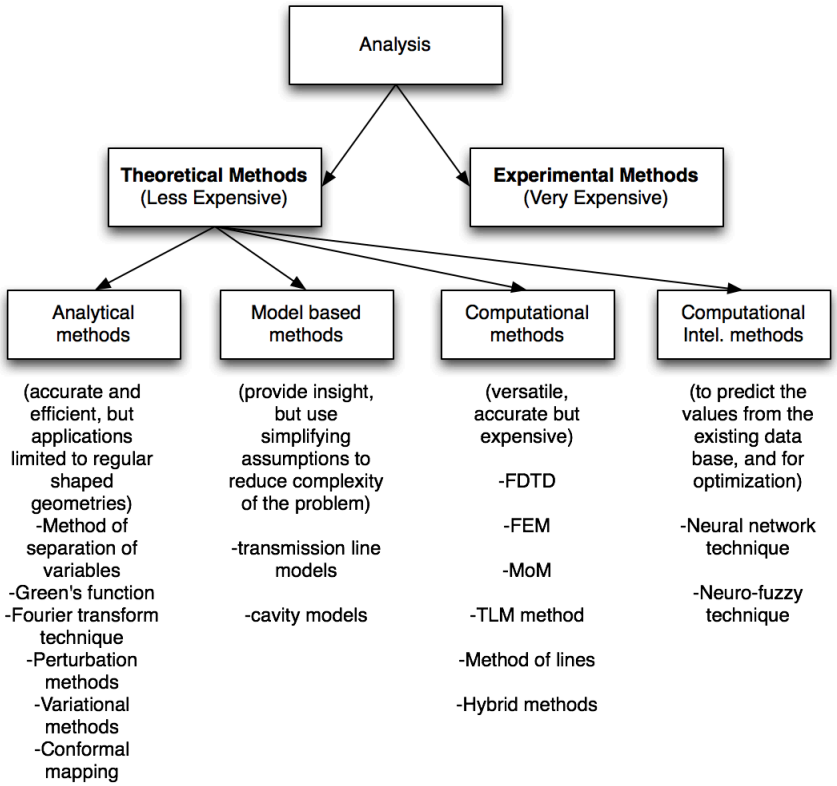

Fig. 1.1-1. Breakdown of electromagnetics solution methods, from [7].

While some electromagnetic problems can be approximated with optical theory or circuit theory, those of interest to this work require Maxwell's Equations to be solved to arrive at a precise solution. There are numerous methods for this, which can be further broken down into *theoretical* and *experimental* methods. In early application of electromagnetic theory, experimental methods saw considerable use, notably the development of wireless communications [6]. However, with theoretical methods becoming increasingly easier to implement, experimental methods are generally limited to verification of devices that have already been produced [7].

Theoretical methods, again as shown in Fig. 1.1-1, can be subdivided into purely analytical methods, model-based methods, computational methods, and computational intelligence methods. Analytical methods are very accurate and efficient, however they require (both mathematically and colloquially) complex derivations, e.g., [8]. This generally limits usage to more easily integrable geometries, e.g., spheres and planes, however their usage can be combined with other theoretical methods in part as will be seen in Section 2.1.

Model based methods are also useful, whereby a set of governing equations is established for a particular problem set, and new problems can be solved by setting the appropriate parameter values within these equations. This is particularly useful for problems involving cavities/waveguides and/or transmission lines. A comprehensive example as applied to cavities is found in [9].

Computational methods, one of which is the focus of this work, involve discretizing the problem geometry and solving a set of simultaneous equations of order N, where N is the number of discretized elements, thus calculating a solution based on Maxwell's equations. These methods are very versatile and generally suffer far fewer limitations as to problems that can be solved. Accuracy varies from solver to solver but they can be quite accurate when such is an emphasis of the method [10]. However, as all methods are representing continuous physics with discrete numbers, they intrinsically suffer from limitations in numerical precision, typically 64-bit in modern computers without incurring penalties to the efficiency of the hardware. Computational methods also tend to be expensive, with the order of operations for modern algorithms often scaling as $O(N^3)$.

Finally, there is growing development in computational intelligence methods [7]. These methods attempt to "train" an artificial intelligence about the nature of the problem and physics allowing it to "predict" future results. Often this takes the form of an optimization engine driving repeated analysis of a prior method [11]. These methods are of growing interest as research progresses; however, they are beyond the scope of this work.

1.2 Full-Wave Numerical Electromagnetics Overview

While there are a number of methods within full-wave numerical electromagnetics, the nature of the problem yields some attributes shared by all methods. First, a governing equation must be determined, generally some form (integral or differential, continuous or discrete, etc.) of Maxwell's equations. Second, a discretization of the geometry needs to be done. As the solver is of the numerical type, even continuous equations cannot be solved as such with a symbolic equation for a solution. This discretization applied to the governing equation results in a system of simultaneous linear equations in the form of a matrix equation,

$$[A][x]=[b] \quad . \tag{1.2-1}$$

The width and length of the matrix (and thus number of unknowns, *N*) is equal to the number of elements into which the circuit has been *discretized* (or, interchangeably, *subsectioned*). Third, a matrix equation solver must be applied to the system. Finally, post-

processing is done on the resulting solution. This represents, e.g., converting current distribution data into S-parameters or RADAR cross-sections (RCS) [7].

Within numerical electromagnetics there are three main methods, a number of additional methods, and countless sub-divisions and derivatives of these. We consider Method of Moments (MoM), Finite Difference Time Domain (FDTD), and Finite Element Method (FEM), to be foremost.

1.2a Method of Moments (MoM)

Early examples of the Method of Moments as applied to electromagnetics include the efforts of E. N. Vasil'ev [12] and R. F. Harrington [1], and it has since become one of the leading computational methods within electromagnetics [13]. MoM is generally implemented as a time-harmonic, frequency domain technique, for application exclusively to linear problems. It is dependent on derivation of a dyadic Green's function (DGF) (i.e., the field due to an impulse current source), which is analogous to a spatial impulse response of a linear system, and thus relates electric field to current distribution through a convolution integral, e.g. in [14],

$$\overline{E}\left(\overline{r}\right) = j\omega\mu \iiint d\overline{r}\,'\overline{\overline{\mathrm{G}}}\left(\overline{r},\overline{r}\,'\right)\cdot\overline{J}\left(\overline{r}\,'\right) \ . \qquad (1.2\text{-}2)$$

Although numerical calculation of the DGF values is often difficult, time consuming, and subject to error [10], it allows for the creation of an impedance matrix [Z]. The impedance matrix relates a voltage array [v] and a current array [i] as a system of linear equations,

with one for each subsection of the discretized geometry. For a 2D implementation, this results in a system of the form,

$$\begin{bmatrix} \mathbf{Z}^{xx} & \mathbf{Z}^{xy} \\ \mathbf{Z}^{yx} & \mathbf{Z}^{yy} \end{bmatrix} \begin{bmatrix} \mathbf{I}^{x} \\ \mathbf{I}^{y} \end{bmatrix} = \begin{bmatrix} \mathbf{V}^{x} \\ \mathbf{V}^{y} \end{bmatrix}, \tag{1.2-3}$$

where individual matrix elements can be calculated as

$$Z_{ij}^{\alpha\beta} = \int_{S_i} t_i^{\alpha}(\vec{r}) \int_{S_j} G^{\alpha\beta}(\vec{r},\vec{r}\,')b_j^{\beta}(\vec{r}\,')\,ds', \quad V_i^{\alpha} = \int_{S_i} t_i^{\alpha}(\vec{r})E_{inc}^{\alpha}(\vec{r})\,ds, \tag{1.2-4}$$

$$\alpha,\beta = x,y; \quad i,j = 1,\ldots,N^{\alpha,\beta}$$

This is very convenient, as it means that only boundary elements of the metallization need to be discretized and not entire volumes, radically reducing the number of unknowns. Indeed, some call MoM the boundary element method (BEM) [13]. However, as this matrix must include interactions from every element to every other element (and vice versa), this unfortunately results in a dense matrix. Due to reciprocity [14], the dense matrix is symmetric which may be exploited.

MoM implementations often feature rectangular patches for subsections, as the derivations (detailed in Chapters 2 and 3) are more straightforward. However, it is possible to subsection with triangle patches as well for geometries that are not as conveniently discretized, at additional computational cost [15]. The current for each patch is one element of the current array [*i*] which may alternatively be viewed as an array of *weighted residuals.* Indeed, MoM is sometimes referred to outside of electromagnetics as the method of weighted residuals [16].

1.2b Time Domain Methods

Finite-difference-time-domain (FDTD), initially proposed in [17], is one of the leading time-domain methods. It greatly contrasts MoM. Instead of solving the frequency domain version of Maxwell's equations, it is self-evidently solving the time domain version. Instead of solving the integral equivalent of Maxwell's equations, FDTD solves the latter directly in their native form of coupled partial differential equations. Moreover, as it approximates the differential operator discretely, no DGF and indeed, no matrix, is required when Maxwell's equations are solved explicitly [7], [13]. When they are solved implicitly, a sparse matrix is constructed which must be solved directly or iteratively.

FDTD is implemented by discretizing the structure into a grid of nodes. The interaction between each node is calculated at a number of "time steps." In explicit schemes, timestep size is limited by the *Courant Limit*, which in part drives scaling [13], though implicit schemes also exist which are not subject to the Courant Limit. These methods are particularly useful for problems where time-based "animation" of physics is necessary, as well as those featuring sources that are not time-harmonic, e.g., transients. FDTD schemes, unlike MoM, are able to handle non-linear problems. However, as calculations are not done in the frequency domain, dispersive materials (i.e., those with real and/or imaginary permittivity that vary with frequency) are particularly difficult or impossible to handle with FDTD methods.

A widely used variant of FDTD is the finite integration technique (FIT) originally shown in [18]. FIT solves all four of Maxwell's equations in a consistent way, the combination of

which yields a unique solution, which is more stable. Although still dependent on the FDTD gridding, FIT has also been extended to subgridding [19].

Also among time-domain methods is the finite volume time domain method (FVTD), allowing for geometries that do not conform to a regular grid, e.g., hexahedrals and tetrahedrons. It functions by casting Maxwell's equations into the conservation law, and then solving the resulting equations. An early use in electromagnetics can be found in [20].

Another branch of time-domain methods includes discontinous Galerkin (DG) method, originally applied in [21] for solution of the neutron transport equation. It has since been applied to electromagnetics in, including work such as [22]. The method takes the form of a combination of Finite Element Method (FEM) in Section 1.2c and FVTD previously detailed.

1.2c Finite Element Method (FEM)

Finite Element Method, while previously extant, was first applied to electromagnetics by Arlett et al. in [23]. Like MoM, the Finite Element Method can involve a matrix and arrays representing a linear system to solve Maxwell's equations. Like FDTD, the FEM solves them in differential form. FEM is widely used in other computational methods for other branches of physics, making it more easily incorporated into multiphysics environments. As it is generally used to mesh entire volumes as opposed to surface boundaries, FEM can easily handle materials that are inhomogeneous in nature. This volume mesh then solves for the fields using either a MoM-like weighted residual matrix (albeit a sparse one) or with a matrix-free variational method [13]. However, discretizing the entire volume of the

problem geometry greatly increases the number of unknowns vs. surface-meshing methods such as MoM/BEM, which only have to discretize the boundaries—this is demonstrated with an example problem between surface meshing and volume meshing tools in Section 4.2. This is amortized, as the matrix equation that is generated is sparse and thus requires less memory and processing (FLOPS) to achieve a solution. Further, FEM can mesh the structure as nodes similar to FDTD, or as edge-based, as is commonly done in MoM [7].

1.2d Transmission Line Matrix (TLM), Method of Lines (MoL), and Generalized Multipole Technique (GMT)

While the previous *big three* methods make up a large body of conventional EM solvers, the difficult nature of Maxwell's equations allows for development of a large number of more specialized methods and solvers, including Transmission Line Matrix (TLM), which discretizes circuits as a set of transmission lines, Method of Lines (MoL), which specializes in waveguide structures, and Generalized Multipole Technique (GMT), which uses multipoles as basis functions [13].

1.2e Methods behind Numerical Electromagnetics

While the previous subsections review several algorithms used for numerical electromagnetics at a macroscopic level, it is worth noting some methods that are used within them. For example, in order to convert continuous operator problems to discrete ones, Galerkin and point matching methods are often used. For FDTD methods, simple finite differences are implemented at multiple time steps. For matrix solutions, Gaussian

elimination and LU factorization with back-substitution are frequently used for direct solution, and for iterative solution, conjugate gradient and GMRES methods are often utilized. Fourier transforms can help with time-domain/frequency-domain conversions, and FFTs are often employed to accelerate fast solvers [7].

1.3 Previous Work in Speed Enhanced Electromagnetics

As full matrix inverse solution of the dense matrices resulting from the methods in Section 1.2 is often computationally prohibitive, there has been considerable development in other techniques that benefit from enhanced speed and scaling. For example, within MoM solvers there has been application of iterative conjugate gradient (CG) methods [4], [24] which reduce processing requirements in solving the system of equations by repeating matrix vector products in an optimization loop (see Section 3.2), however the dense nature of the MoM matrix results in memory and operations that unfortunately both scale as $O(N^2)$, and memory requirements actually scale worse than a full solution as preconditioner storage is added to memory requirements. However, these works enabled a new class of iterative solvers that employ further enhancements.

For electromagnetic characterization of high-density and/or electrically large planar circuits there have been several classes of fast algorithms developed over the last two decades. These methods become increasingly more unique and specific as opposed to the general methods listed in Section 1.2.

1.3a Kernel Dependent Methods

Among these fast methods are kernel dependent iterative fast methods, such as multi-level fast multipole algorithm (MLFMA) [25]-[26]. MLFMA uses a hierarchical representation of interactions within a circuit to speed matrix-vector multiplies. Those interactions that are nearer (or at a *lower level* within the multi-level hierarchy) are computed together. Interactions that are futher away (or at a *higher level*) are calculated at another stage, with lower level interactions being inclusive.

CG-FFT algorithm [4], [27]-[30], accelerates the matrix-vector multiplies with implicit matrix representation and FFT solve (see Section 3.3a for more detail). Geometries must conform to a regular grid but can allow higher precision as shown by UFFT-GT in this work. Another kernel dependent method, Pre-corrected FFT (PFFT) algorithm (a.k.a. Adaptive Integral Method (AIM) [3],[31]-[35]), is similar to CG-FFT, albeit notably allows for off-grid discretization and is utilized in the UFFT-P version of this work. It achieves this by projecting the non-grid-conforming geometry onto a grid, interpolating the results back, and pre-correcting based on calculations from explicit matrix elements.

Other methods worth noting include the integral equation FFT (IE-FFT) method [36] and the sparse-matrix/canonical grid (SM/CG) Method [37]. Additionally, speed-enhanced direct solvers have been extended to finite-element analysis in a deterministic fashion [38].

1.3b General Kernel Independent Methods

There are several kernel-independent algorithms, such as iterative QR- and SVD-based compression techniques [39]-[41], iterative [42]-[43], and direct [44] H-matrix based methods, as well as iterative adaptive-cross-approximation (ACA) method [45]-[46]. Interestingly, ACA methods were originally shown in [47], albeit in a less application specific and more general mathematical sense. Other methods include iterative wavelet-based compression method [48].

1.3c Comparisons of Previous Work to This Work

Though the previous group of hierarchical-matrix algorithms can be applied to acceleration of MoM solutions in conjunction with complex DGFs of layered media and/or perfect shielding enclosures as in this work, special implementations are required to produce sufficient accuracy of results for full wave physics [49] and so they are considered distinct. Conversely, the full-wave algorithms detailed previously, such as MLFMA, PFFT and CG-FFT, natively retain both accuracy and efficiency in capturing full-wave physics, require significant and rarely implemented modification for complex media. MLFMA has been extended to handle full-wave layered kernels in its modification, fast inhomogeneous plane wave algorithm (FIPWA) [50]-[51], however it does not demonstrate rigorous error control beyond two to three significant figures. MLFMA has also been extended for the case of static layered kernels [52]-[53], and rectangular enclosures in the absence of layered media [54]. This is all in contrast to this work, which can handle full wave physics, multilayered media, and a shielding enclosure while still exhibiting computational and memory complexity as a fast solver.

In the class of FFT-based algorithms, which are most similar to this work, several modifications of algorithms have been developed for unshielded layered media [26], [55]-[57]. The attempt of generalizing the PFFT method to the case of shielded stratified media supported by this work has been made before [58]. This method, however, fell short of expectations due to the large computational complexity associated with matrix fill operations (MFO), with only moderate size problems having been analyzed. Issues in [58] are remedied here through application of an FFT-enhanced algorithm for MFO developed by J. Rautio and implemented for rapid MFO in Sonnet® em® [2]. This MFO algorithm is shown to scale as $O(N \log N)$, where N is approximately the number of unknowns associated with the MoM solution. Additionally, the previously developed PFFT generalization for shielded stratified media [59] has been further advanced through introduction of a more accurate method for basis function projection onto the FFT grid utilizing waveguide mode matching instead of standard multipole reproduction criteria [3]. The PFFT algorithm for fast handling of the matrix solve operations (MSO) has also been generalized to the handling of both PEC and mixed PEC/PMC types of enclosures. Additionally, the CG-FFT algorithm may also be used to ensure high accuracy.

Further, FFTs have been used to increase parallelization of MLFMA with the MLFMA-FFT method [60], and a similar application to GPU clusters [61], though the use of FFT does not increase the accuracy of the multi-level decomposition. [62] has extended an FFT-based approach for cavities, similar to this work, albeit for scattering problems as opposed to circuits.

1.4 Research Contributions

While there are many previously developed $O(N \log N)$ solvers, both research-based and commercially available, as detailed in Section 1.2, they are either not optimized for use in shielded-domain electromagnetic problems, incur a significant and at times unsuitable accuracy penalty, or are subject to some other mitigating factor.

With this work, a speed-enhanced method is developed for shielded domain, 3D planar, full-wave solution of Maxwell's equations. It is based on a combination of PFFT or CG-FFT algorithms applied for MSO and an existing FFT-enhanced MFO algorithm. When paired together, as shown in the block diagram of Fig. 1.4-1, this is termed Unified-FFT (UFFT). It is to be proven that this method can perform error-controllable analysis with respect to the direct MoM solution that scales as $O(N \log N)$ for operations and $O(N)$ for memory.

As MFO and MSO are the two main time consuming operations in the MoM solution of the Electric Field Integral Equation (EFIE), the novel use of FFT-accelerated versions of both in conjunction creates the Unified-FFT methodology for the expedient MoM solution of planar electromagnetic analysis in the shielded layered environment.

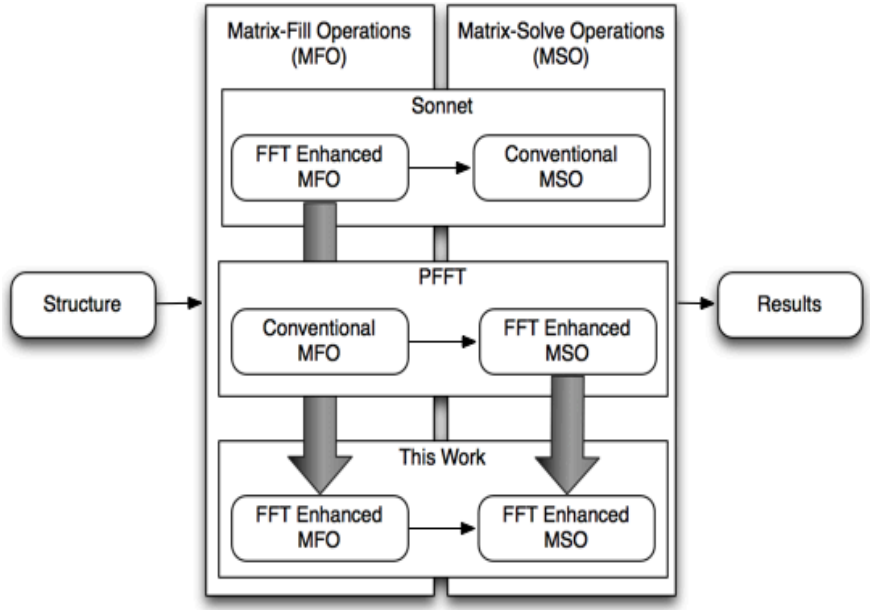

Fig. 1.4-1. A breakdown of the components that make up the UFFT-P implementation and the previous projects from which the components are sourced.

For clarity it should be explained that this is expressly a 3D-planar solver as opposed to an arbitrary-3D solver. It accepts limitation of simulation capability to predominantly planar structures, such as those printed on PCBs and ICs, in exchange for more efficient meshing of the structure. Note that while this generally reduces the number of unknowns significantly (as demonstrated in Section 4.2), it does not reduce the complexity of the algorithm per unknown.

Notably, this work develops two methods of UFFT. UFFT-precorrected (UFFT-P) and UFFT-grid-totalizing (UFFT-GT). These methods allow for more precise control of flexibility vs. accuracy and differences are shown briefly in Fig. 1.4-2 and detailed in Section 3.1. Note that the maximum unknowns for UFFT-GT (500 000) are in a simulation example not

further detailed in this work. Fig. 1.4-1 also applies to UFFT-GT, with the notable exception of the PFFT block being replaced with an equivalent CG-FFT block. UFFT-P is developed with a FORTRAN- [63] and MATLAB- [64] coded software package that makes calls to the existing commercial solver, Sonnet [2]. UFFT-GT is developed completely independently from UFFT-P as both a new algorithm and new solver. It is written entirely in MATLAB- [64], Sonnet [2], and SonnetLAB [65] as an interface between the two.

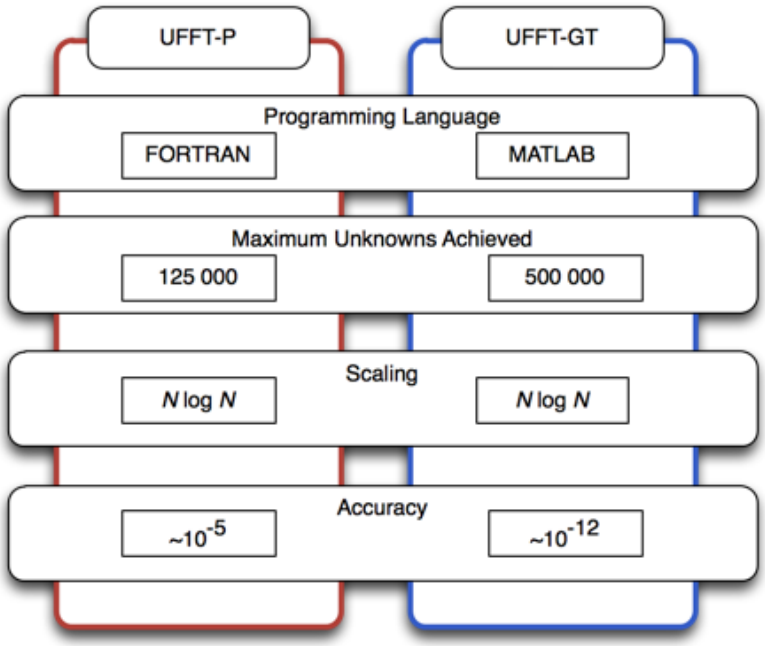

Fig. 1.4-2. A basic comparison of the two version of the new UFFT methodology.

Chapter 2 of this dissertation explains how FFT is used to enhance the MFO of the solver, including the **K**-Matrix formulation. It is detailed first in a general sense, and then in detail for both UFFT-P and UFFT-GT, as there are small but important differences. Chapter 3 explains how FFT is used to enhance the MSO of the solver. It starts with an overview of the differences between PFFT and CG-FFT, then it overviews GMRES, the iterative algorithm on which both are based, and follows that with a detailed description of how the MSO works

for both solvers. Chapter 4 includes a number of example circuits, which are simulated and benchmarked against various commercial solvers. Chapter 5 discusses directions that future research efforts can take which are based upon this work, including some features that are necessary to be developed for the commercialization of this research. Finally, Chapter 6 concludes the dissertation.

2. FFT-Enhancement of Matrix Fill Operations

In this chapter the FFT-Enhancement of the MFO is detailed, as well as a method to perform the DGF calculation for UFFT in an FFT-enhanced manner.

2.1 Overview of the K-Matrix

This section introduces the **K**-Matrix, a mathematical device which is used to rapidly construct matrix elements through the use of FFT. Notably, while some work has been done with high performance FFTs that sample at non-uniform intervals [66], the fastest and most reliable FFT algorithms require sampling at uniform intervals. In this case, the FFT takes place in 2D space across the circuit geometry. As such, a uniform grid is introduced for basis and testing functions on the geometry, as shown in Fig. 2.1-1.

To better explain the algorithm and usage of the **K**-matrix, steps are described to calculate an example matrix element, Z_{pq}^{xx}, in a Galerkin MoM for roof-top basis and testing functions $b_q^x(\vec{r})$ and $t_p^x(\vec{r})$ [67] that conform to the aforementioned regular grid. The grid-spanning waveguide cross-section with N_b divisions from *0* to *a* (along the *x*-axis) and M_b divisions from *0* to *b* (along the *y*-axis of the box) and is introduced. Analytic evaluation of the integrals of basis and testing functions in (1.2-4), followed by reordering of summation as in [68], produces the following form,

$$Z_{pq}^{xx} = \sum_{m=1}^{M_b/2-1} \sum_{n=1}^{N_b/2-1} \sin\frac{n\pi n_p}{N_b} \sin\frac{n\pi n_q}{N_b} \cos\frac{m\pi m_p}{M_b} \cos\frac{m\pi m_q}{M_b} k_{mn}^{xx}, \qquad (2.1\text{-}1)$$

where $(m_p a / M_b, n_p b / N_b)$ and $(m_q a / M_b, n_q b / N_b)$ are the centers of the test and basis functions, respectively.

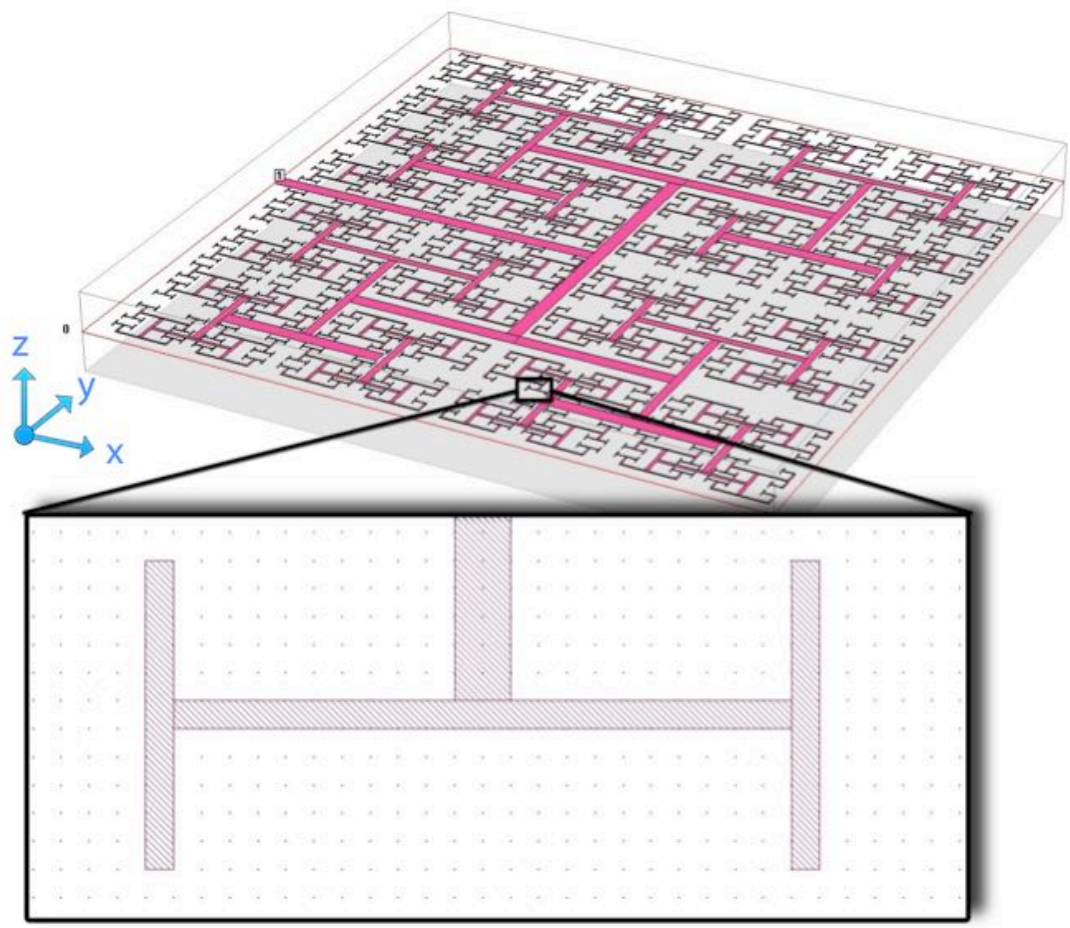

Fig. 2.1-1. The clock network example geometry (see Section 4.1) with a close-up view of the uniform grid.

Coefficients k_{mn}^{xx} in (2.1-1) are defined as an infinite double series that can be calculated to desired precision [68]. This is extremely important for UFFT-GT, as calculating these elements to double precision is an obvious but necessary step to solving the system of equations to double precision as compared to the reference MoM solution. These

coefficients are independent of the locations of test and basis functions and are solely defined by dimensions of the waveguide and the layered substrate. Using identities $\sin x = [\exp(jx) - \exp(-jx)]/2j$ and $\cos x = [\exp(jx) + \exp(-jx)]/2$ for each of the sine and cosine functions in (2.1-1) we obtain the matrix element in the form

$$
\begin{aligned}
Z_{pq}^{xx} = & \\
& \sum_{m=-M_b/2-1}^{M_b/2+1} \sum_{n=-N_b/2}^{N_b/2} \exp\left[j\frac{n\pi(n_p - n_q)}{N_b} \right] \exp\left[j\frac{m\pi(m_p - m_q)}{M_b} \right] k_{mn}^{xx} \\
& + \sum_{m=-M_b/2-1}^{M_b/2+1} \sum_{n=-N_b/2}^{N_b/2} \exp\left[j\frac{n\pi(n_p - n_q)}{N_b} \right] \exp\left[j\frac{m\pi(m_p + m_q)}{M_b} \right] k_{mn}^{xx} \\
& + \sum_{m=-M_b/2-1}^{M_b/2+1} \sum_{n=-N_b/2}^{N_b/2} \exp\left[j\frac{n\pi(n_p + n_q)}{N_b} \right] \exp\left[j\frac{m\pi(m_p - m_q)}{M_b} \right] k_{mn}^{xx} \\
& + \sum_{m=-M_b/2-1}^{M_b/2+1} \sum_{n=-N_b/2}^{N_b/2} \exp\left[j\frac{n\pi(n_p + n_q)}{N_b} \right] \exp\left[j\frac{m\pi(m_p + m_q)}{M_b} \right] k_{mn}^{xx}
\end{aligned}
\quad . \qquad (2.1\text{-}2)
$$

The matrix element Z_{pq}^{xx} in (2.1-2) is seen to be the sum of the four terms, each of which is in the form of a 2D discrete Fourier transform (DFT). Further, each of the DFTs have convolution/correlation dependence on the indices defining spatial positions of the basis and testing functions. Thus, any matrix element can be obtained by first calculating matrix K^{xx} using FFT as

$$
K_{\mu\nu}^{xx} = \sum_{m=-M_b/2}^{M_b/2-1} \sum_{n=-N_b/2}^{N_b/2-1} \exp\left[j\frac{n\pi\nu}{N_b} \right] \exp\left[j\frac{m\pi\mu}{M_b} \right] k_{mn}^{xx} = FFT\{k_{mn}^{xx}\}, \qquad (2.1\text{-}3)
$$

for $\mu = -M_b/2,\ldots,M_b/2-1$, $\nu = -N_b/2,\ldots,N_b/2-1$.

2.2 General MFO Formulation

Calculating the matrix element values from the **K**-matrix in the previous chapter is simply

$$Z_{pq}^{xx} = K_{m_p-m_q,n_p-n_q}^{xx} + K_{m_p-m_q,n_p+n_q}^{xx} + K_{m_p+m_q,n_p-n_q}^{xx} + K_{m_p+m_q,n_p+n_q}^{xx} . \qquad (2.2\text{-}1)$$

Note that for indices $m_p \pm m_q$ and $n_p \pm n_q$ outside the range $-M_b / 2,\ldots,M_b / 2 - 1$ and $-N_b / 2,\ldots N_b / 2 - 1$, values are calculated according to periodicity of the DGF which is reflected in the matrix K^{xx}. This calculation scales as $O[M_bN_b\log(M_bN_b)]$ operations due to FFT, where N_b and M_b are the number of discretization cells on the x and y axis, respectively.

Subsequently, all Z_{pq}^{xx} matrix elements can be calculated from K^{xx} via (2.2-1). In the case of UFFT-P and other matrix-free methods only $O(N)$ matrix elements $\mathbf{Z}_{MoM}^{near}$ corresponding to the near interactions need to be computed, as opposed to $O(N^2)$ in dense methods, where the matrix is explicitly stored. Thus, calculation of $\mathbf{Z}_{MoM}^{near}$ requires $O[M_bN_b\log(M_bN_b)]$ operations with the proposed FFT enhanced approach. In a similar manner, matrices K^{xy}, K^{yx}, and K^{yy} are computed for evaluation of the remaining three blocks $\mathbf{Z}^{xy}$, $\mathbf{Z}^{yx}$, and $\mathbf{Z}^{yy}$ of the impedance matrix **Z** in the general form

$$\begin{aligned} Z_{pq}^{\alpha\beta} = {} & s_1^{\beta} K_{m_p-m_q,n_p-n_q}^{\alpha\beta} + s_2^{\beta} K_{m_p-m_q,n_p+n_q}^{\alpha\beta} \\ & + s_3^{\beta} K_{m_p+m_q,n_p-n_q}^{\alpha\beta} + s_4^{\beta} K_{m_p+m_q,n_p+n_q}^{\alpha\beta} , \end{aligned} \qquad (2.2\text{-}2)$$

where for the case of PEC walls the factors s_n^{β} are given by $s_1^x = s_1^y = 1;\ s_2^x = -1,\ s_2^y = 1;$ $s_3^x = 1,\ s_3^y = -1;\ s_4^x = s_4^y = -1$, and for the case with walls made of PMC the sign factors are $s_1^x = s_1^y = 1;\ s_2^x = -1,\ s_2^y = 1;\ \ s_3^x = -1,\ s_3^y = 1;\ s_4^x = s_4^y = 1,$ where β denotes the source basis

function orientation, *x* or *y*. In the case of UFFT-GT, no matrix elements are explicitly calculated from the **K**-matrix, however it is still used to generate the implicit matrix as detailed in Section 2.4.

2.3 UFFT-P Matrix Fill

UFFT-P is the first fast solver algorithm that combines FFT-enhanced MFO with MSO operations. It uses conventional Sonnet for MFO and the pre-corrected FFT (PFFT) algorithm for MSO. The PFFT algorithm uses a projection and interpolation to put geometries which may or may not apply to a regular grid onto a regular grid, which allows for fast solution with FFTs. This projection and interpolation, however, introduces additional error to the simulation. While the general MFO algorithm from Section 2.2 is directly applicable to the evaluation of the matrix elements in the MoM formulation with geometry conforming to a regular grid, it can also be used for a more general DGF calculation. For example, in implementation of MoM with off-grid meshes, as used in UFFT-P, it allows for the DGF to be calculated in an FFT-enhanced manner as opposed to slower methods such as discrete complex image method (DCIM), detailed in Appendix B. In this case, taking as basis and testing functions in (1.2-4) the delta-functions located at the nodes of the regular grid, i.e.,

$$b_q^x(\vec{r}) = \delta\left[x - \frac{m_q a}{M}\right]\delta\left[y - \frac{n_q b}{N}\right], t_p^x(\vec{r}) = \delta\left[x - \frac{m_p a}{M}\right]\delta\left[y - \frac{n_p b}{N}\right],$$

and doing a summation reordering according to [68], we obtain the representation for the Green's function samples on the regular grid in the same form as (2.1-1)

$$G_{pq}^{xx} = \sum_{m=1}^{M/2-1} \sum_{n=1}^{N/2-1} \sin\frac{n\pi n_p}{N} \sin\frac{n\pi n_q}{N} \cos\frac{m\pi m_p}{M} \cos\frac{m\pi m_q}{M} g_{mn}^{xx} . \tag{2.3-1}$$

By casting (2.3-1) into the DFT based representation similar to (2.1-2) we can calculate the Green's function for any pair of observation and source points on the grid as

$$G_{pq}^{xx} = \Gamma_{m_p-m_q,n_p-n_q}^{xx} + \Gamma_{m_p-m_q,n_p+n_q}^{xx} + \Gamma_{m_p+m_q,n_p-n_q}^{xx} + \Gamma_{m_p+m_q,n_p+n_q}^{xx} , \tag{2.3-2}$$

where the generating matrix Γ^{xx} is defined on the regular grid as

$$\Gamma^{xx} = FFT\{g^{xx}\}$$

similar to (2.2-2). From a matrix of regular grid samples Γ^{xx} the values of function $\tilde{\Gamma}^{xx}(x,y)$ at any arbitrary location on $x \in [0,2a]$, $y \in [0,2b]$ can be obtained via 2D interpolation [69]. Following similar procedure for other components of the DGF we obtain the following continuous dependence of the DGF on the source and observation point coordinates

$$\begin{aligned} G^{\alpha\beta}(x,y;x',y') = {} & s_1^{\beta}\Gamma^{\alpha\beta}(x-x',y-y') + s_2^{\beta}\Gamma^{\alpha\beta}(x-x',y+y') \\ & + s_3^{\beta}\Gamma^{\alpha\beta}(x+x',y-y') + s_4^{\beta}\Gamma^{\alpha\beta}(x+x',y+y') \end{aligned} . \tag{2.3-3}$$

Equation (2.3-3) can be utilized in evaluation of the impedance matrix elements in MoM defined on non-uniform meshes using standard quadrature rules. Sign factors s in (2.3-3) are the same as in (2.2-2). The proposed approach is appropriate for evaluation of the matrix elements in pairs of source and basis functions which do not encounter the DGF singularity. The matrix elements involving integration of the DGF singularity in the case of non-uniform mesh based MoM can be evaluated via direct summation over waveguide modes in (1.2-4) after the surface integrals are evaluated analytically. Alternatively, the

static component of the DGF can be first extracted from the waveguide DGF spectrum using Discrete Complex Image Method (DCIM) [70], [71] prior to its casting to the form (2.3-1). Subsequently, the infinite series of the static contribution can be calculated as described in Appendix B [72]. Another related PFFT technique is found in [73].

2.4 UFFT-GT Matrix Fill

UFFT-GT, as previously described briefly, is a variant of UFFT which uses the CG-FFT algorithm for MSO. This allows for fast analysis with no additional error beyond the direct MoM solution. As mentioned, a uniform grid is first overlain on the circuit geometry situated in the general multilayered medium transverse to the box formed by the side-walls of a rectangular enclosure, as shown in Fig. 2.1-1. There are a total of M_b cells across the height of the grid and N_b cells across the length of grid, which lies in an *XY*-plane at a point on the *Z*-axis where two of the stratified dielectrics interface, with the plane containing the metallization of the circuit. Note that grid dimensions may be defined arbitrarily in both dimensions, even for cells with wide aspect ratios, though they must sample at uniform intervals for proper FFT usage. This uniform grid is the same as in the MoM implemention of Sonnet's EM engine [2]. This grid is used for FFT-application to perform MFO to machine precision in $O(N \log N)$ operations. The MoM impedance matrix fill and storage, however, is considerably different from that of a conventional MoM impedance matrix.

Rather than filling the traditional MoM for given circuits, the UFFT-GT method relies on the translational invariance property of the DGF previously noted. It is quite similar to the

K-matrix based conventional MFO in Sonnet, however it is used to store the matrix in an implicit fashion. Indeed, all possible interactions within the totalized grid are represented indirectly, as opposed to just those for where metallization is present. By storing *all* interactions, (considerably more information than is present in even a dense $O(N^2)$ Sonnet matrix, despite considerably less storage space), and separately storing each of the four convolution and correlation terms from the translational invariance of the DGF, one arrives at a set of large 4D Toeplitz and Hankel matrices, for which 2D analogs (as are used in this work), and their 1D representations (as are stored) are of the form

$$\boldsymbol{A}_{n,m} = \boldsymbol{a}_{n+m-1} \tag{2.4-1a}$$

$$\boldsymbol{B}_{n,m} = \boldsymbol{b}_{n+m-1} \tag{2.4-1b}$$

$$\boldsymbol{C}_{n,m} = \boldsymbol{c}_{n-m+1} \tag{2.4-1c}$$

$$\boldsymbol{D}_{n,m} = \boldsymbol{d}_{n-m+1} \tag{2.4-1d}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are Hankel matrices, and $\boldsymbol{C}$ and $\boldsymbol{D}$ are Toeplitz matrices, and $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$ are the arrays that represent them implicitly.

Calculating and storing all elements of these 4D matrices would indeed yield far higher storage and computational requirements vs. a traditional MFO, however, as this abstraction has composed matrices into strictly 2D Toeplitz and Hankel ordered structures, all matrices are both calculated and stored merely as a single row. This, and (2.4-1), are

demonstrated in Fig. 2.4-1 for a 2D Hankel type matrix, where elements beneath the anti-diagonal are unimportant and not shown.

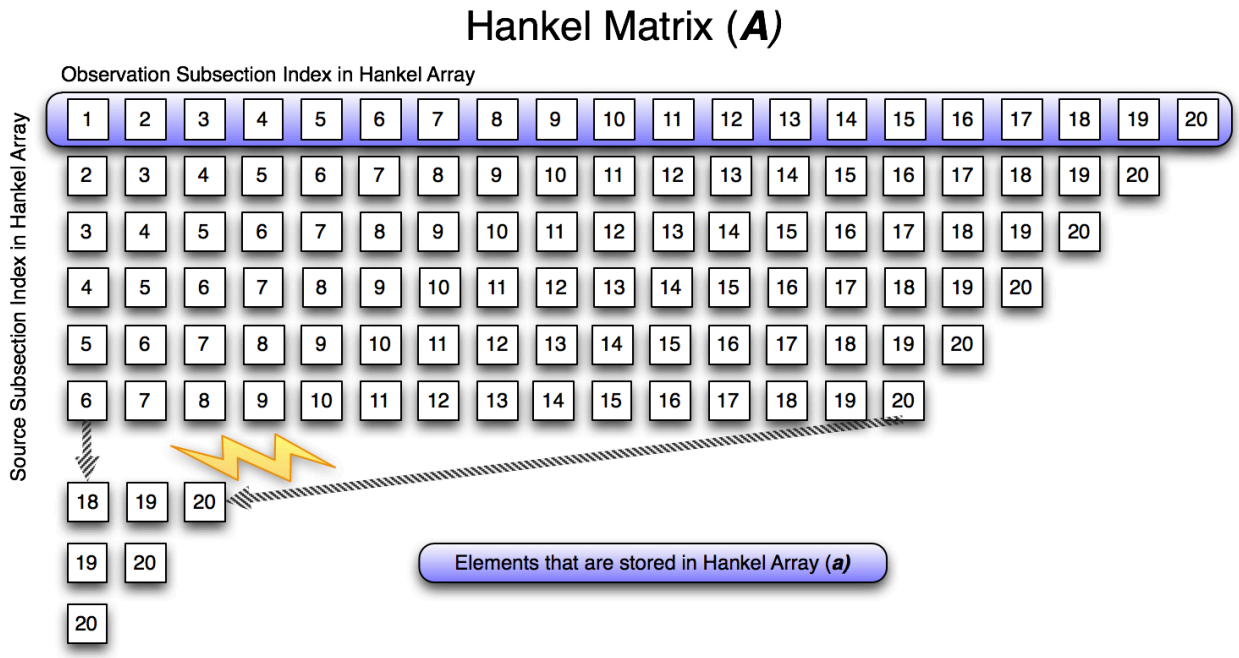

Fig. 2.4-1. Example of the implicit storage of one of the 2D Hankel-type matrices, A.

Conventionally the matrix takes the 2D Toeplitz form natively for circuits that are one-dimensional and discretized along the entire length, e.g., the case of a dipole wire antenna. For two-dimensional circuits, this translates to 4D Toeplitz and Hankel type matrices. For this work, however, matrices are down-converted to 2D, with its unique elements stored in a single dimension as in Fig. 2.4-1. The physical distribution of the indexing of the 1D generating array (the first row of Fig. 2.4-1, with four forms referred to as in (2.4-1) as ***a***, ***b***, ***c***, or ***d***, depending on term)—is depicted in Fig. 2.4-2; all interactions between a single source subsection and all possible observation subsections across twice the box cross-section are shown. Book notation is used for the matrices with operators that feature convolutions or correlations exclusively and reverse is used for matrices featuring both

operators. Note that while this ordering shows the general form, there is some padding and rearranging necessary to ensure Toeplitz and Hankel form in this scenario. The specific ordering is beyond the scope of this document, however, it can be recreated by generating all possible interactions (and not just those of a single source subsection), with each new source subsection position filling an additional row of the set of matrices as depicted by Fig. 2.4-1. Areas of the matrix that do not adhere to the Toeplitz or Hankel format, for example those caused from box-wall half-subsections, vias, etc., can at this point be reordered until they do.

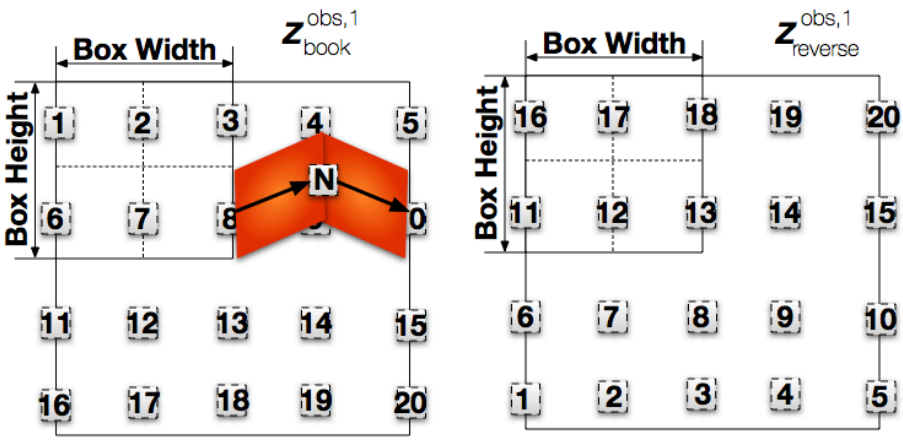

Fig. 2.4-2. Ordering of the Toeplitz and Hankel matrices converted to 1D.

A total of $N_{arrays} = 4DP^2$ of these matrices, (stored, calculated, and henceforth referred to as 'arrays'), are obtained, where D represents the number of tensor elements in the DGF and P represents the number of planes. Note that 4 represents the number of Toeplitz and Hankel arrays into which the impedance matrix elements are decomposed, where one of each of the aforementioned types is stored in Z_{book} and $Z_{reverse}$ format to ensure proper matrix structure. Note that for 2D problems, D = 4, representing x-directed source and x-directed observation interactions, x-directed source and y-directed observation

interactions, *y*-directed source and *x*-directed observation interactions and *y*-directed source and *y*-directed observation interactions. For 3D problems, $D = 9$ to account for the additional tensor elements. It is also theoretically possible to make the reduction $D = 3$ for 2D and $D = 6$ for 3D by way of reciprocity theorem [1]. Unfortunately, while this does reduce memory requirements, due to the way indexing is performed when translating current and voltage data into the Toeplitz/Hankel format (which must be done at every iteration of the simulation), this memory savings greatly increases computational requirements and is not practical. These arrays are each of length

$$x_{length} = 2M_b(2N_b + 1)$$
$$y_{length} = 2N_b(2M_b + 1) \qquad (2.4\text{-}2)$$
$$z_{length} = 2M_b 2N_b,$$

where *x*, *y*, and *z* represent the direction of the observation basis function, and with the overall lengths being theoretically irrespective of source observation basis function direction. The additional +1 term is due to the presence of subsections that must cross the box wall. In practice, the arrays stored are slightly longer and vary with a small amount with respect to source observation direction; this is so as to allow padding for optimum performance accessing memory during FFTs.

2.5 Matrix Element Reconstruction for UFFT-GT Preconditioning

Preconditioning, a process used in conjunction with GMRES (see Section 3.2), is used to help provide expedient convergence and solution of the simulator. Preconditioning generally involves taking an inverse, an LU decomposition or otherwise, of a sparse representation of the MoM impedance matrix. Sparse elements are selected based on geometric proximity or other physical and numerical criteria, and might represent 0.1-2% of the content of the full matrix. Unfortunately, with the matrix stored implicitly, none of these elements are directly available to be used. This has two implications:

1. Matrix elements for the preconditioner must be selected based on criteria other than their respective complex values; as such data is not available during the selection process. Instead, criteria such as geometry proximity must be used.
2. The matrix elements must be calculated.

The solution to (1) is straightforward—elements are selected based on the geometry proximity of the two basis functions interacting, which can be calculated from the available circuit geometry data. For (2), there are two potential solutions. It is possible to do one of the following (A and B):

A. Calculate the impedance matrix elements directly from the **K**-matrix (See Section 2.1).
B. Reconstruct matrix elements from the implicitly stored matrix, i.e., the Toeplitz and Hankel arrays.
C. In UFFT-P, the pre-correction *near part* may be used as the preconditioner.

For UFFT-P, (C) is the clear choice as it reuses data already stored. For UFFT-GT this is not an option as there is no pre-correction. Neither (A) nor (B) is without complication,

however, as the **K**-matrix is not directly accessible in MATLAB, the development time for (B) was predicted to be lower. For UFFT-GT, (B) is implemented, however it is found to be easier to develop at a penalty to runtime performance.

Recalling that the matrix is stored as in Fig. 2.4-1, the process of element reconstruction is essentially finding the equivalent 1D index in the Toeplitz and Hankel implicitly stored matrices that corresponds to the 2D index in the explicitly stored matrix. An example is shown in Fig. 2.5-1 for matrix element $A_{4,3}$.

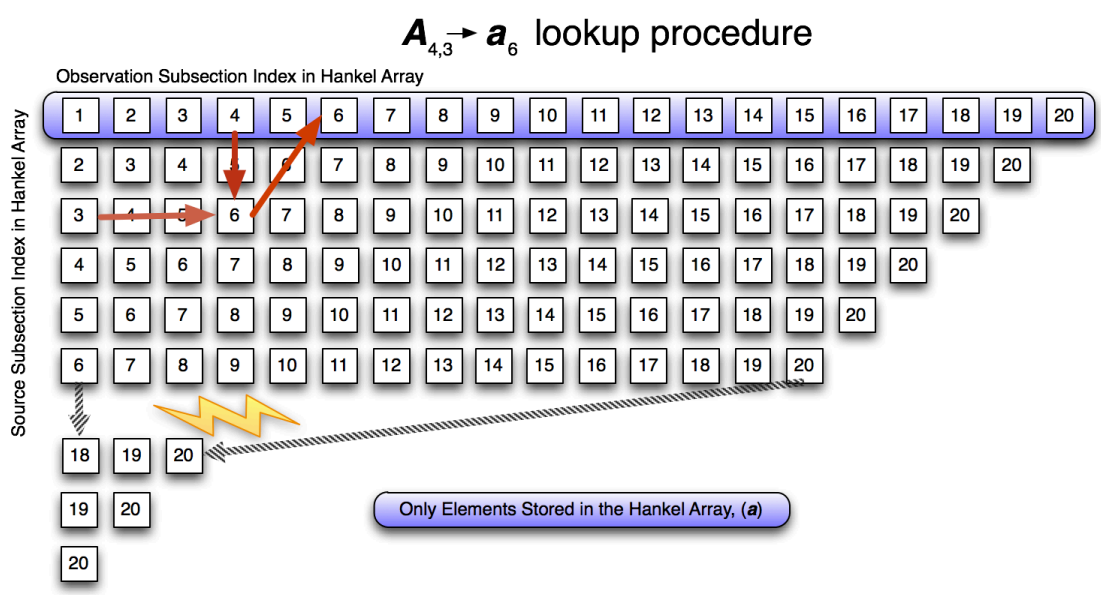

Fig. 2.5-1. Partial reconstruction of example matrix element $Z_{4,3}$ within the associated Hankel implicit matrix, A.

Note that the indices 4 and 3 each have a second mapping between Toeplitz and Hankel matrices (representing a 1D subsection array converted via the *book* and *reverse* notations) and the original dense matrix, as detailed previously in Fig. 2.4-1. This operation is performed for both Hankel matrices, ***A*** and ***B***, corresponding to the appropriate

plane/plane and source-direction/observation-direction set, and then a similar operation is performed for both Toeplitz matrices, $\boldsymbol{C}$ and $\boldsymbol{D}$, in the same set. The resulting four values are summed to reconstruct the matrix element, which for the case of $\boldsymbol{Z}_{n,m} = \boldsymbol{Z}_{4,3}$ yields,

$$\boldsymbol{Z}_{4,3} = \boldsymbol{a}_{\mathbf{6}} + \boldsymbol{b}_{\mathbf{6}} + \boldsymbol{c}_{\mathbf{2}} + \boldsymbol{d}_{\mathbf{2}}, \tag{2.5-1}$$

where 4 and 3 are indices in a two dimensional matrix of Toeplitz or Hankel form, and $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$, are the associated 1D arrays representative of the appropriate axis-axis arrangement (e.g., $\boldsymbol{Z}^{xx}$) and $\boldsymbol{Z}_{n,m}$—which contains the same content but has indexing independent of $\boldsymbol{Z}_{p,q}$—is the $\boldsymbol{Z}$ matrix indexed in the positional fashion. Indexing for *n* and *m* correspond to indices in Fig. 2.4-2, where *n* and *m* are the observation and source indices, respectively, (for example, *m* = 1,…,20, and *n* = 1,…,20 in the case of Fig. 2.4-2). A direct relation between *n,m* and *p,q* can be determined. In the general case,

$$\boldsymbol{Z}_{n,m} = \boldsymbol{a}_{n+m-1} + \boldsymbol{b}_{n+m-1} + \boldsymbol{c}_{n-m+1} + \boldsymbol{d}_{n-m+1}. \tag{2.5-2}$$

The equivalent implicit matrix index is found based on the recurring structure of the Hankel matrices. The source subsection represents the column in the matrix, and the field subsection represents the row (or vice versa where reciprocity is applicable). The intersection point of the two is an implicitly stored element which is not directly in memory, however, in the case of Hankel, it is on the anti-diagonal of an element that is stored, and the anti-diagonal is essentially traced back to find the element. In this case, this component of the interaction between subsections 4 and 3 is the same as 5 and 2, is the same as 6 and 1, etc. This is repeated for all Hankel matrices and the elements are summed. It is then repeated for all Toeplitz matrices (with the exception of tracing the diagonal and not the anti-diagonal) with these terms further summed, at which point the matrix element has been calculated. Unfortunately, while this process is more simple to develop,

performing these indexing and lookup operations in interpreted MATLAB code was found to be low performance. This is because significant flow control and single-threaded operations are required, which performs slowly in MATLAB as they must be interpreted, as opposed to FFT routines which MATLAB links to compiled routines.

In summary, Chapter 2 has described the mathematics and logistics necessary to implement FFT-enhanced MFO within the context of both variants of the UFFT algorithm. This includes introducing the **K**-matrix, a convenient way to store and access Green's function data that is calculated with FFT. Also introduced is the MFO formulation for the technique for the implicit matrix storage, with a detailed mathematical description including PEC and PMC shielding enclosures. Further, the specifics as MFO relates to both UFFT-P and UFFT-GT are discussed. For UFFT-GT, it is convenient to explain the format of the implicit matrix with some details of matrix-vector products and FFTs, which are included. For UFFT-P, as there is a projection onto an additional grid of delta sources, the mathematics are included for this as well. Finally, a method to reconstruct full, explicit, matrix elements (impedances) from the implicitly stored matrix is explained.

3. FFT-Enhancement of Matrix Solve Operations

In this chapter, the two methods for FFT-enhanced MSO used in this work are detailed. These methods are UFFT-P, with MSO powered by PFFT, and UFFT-GT, with MSO powered by CG-FFT. UFFT-P was developed first in this work, and the improvements made with the transition to UFFT-GT are detailed in Sections 3.1a-c. Notably, both methods of FFT-enhancement are dependent on the solver being of the iterative type. Section 3.2 contains an overview of one of the leading iterative solvers, generalized minimum residual (GMRES), which is used in this work.

3.1 Conjugate Gradient (CG-FFT) vs. Pre-Corrected (PFFT) FFT.

It is perhaps most illustrative to start by comparing the similarities of PFFT and CG-FFT. First, both are ultimately fast, matrix-free methods designed to solve matrix-vector products within iterative solvers. They are both able to compute the result of a matrix-vector product without explicitly storing the entire matrix, and whilst performing fewer operations than a conventional $O(N^2)$ matrix vector product. Indeed, they both scale as approximately $O(M_b N_b log\ M_b N_b)$. For EM method of moments (MoM) application with PFFT and CG-FFT (the algorithms theoretically can be used outside of electromagnetics given an appropriate problem set), the *implicitly stored* matrix for both is the impedance matrix, a.k.a., moment matrix. The vector is explicitly stored and contains current information about the circuit. Thus, the result of the matrix-vector product is simply a voltage array, which is solved in an optimization loop, in this case, GMRES, until port excitations are achieved.

Functionally, the main distinction of CG-FFT is that it successfully replaces the $O(N^2)$ matrix vector products of traditional GMRES implementations with $O(M_b log\ M_b\ N_b log\ N_b)$ matrix-free-vector-products, where M_b and N_b are the number of grid elements across the box, without adding any significant error beyond machine precision (typically 10^{-12} or so for MoM implementations). Given that GMRES itself allows for user-defined precision up to nearly machine precision levels, this means that CG-FFT introduces virtually no error as compared with a full, $O(N^3)$ LU-decomposition solution, while being several orders of magnitude faster. Notably, CG-FFT must operate, *in effect*, with more unknowns for the matrix vector product, in that $M_b N_b > N$, sometimes substantially. This is because of the way that CG-FFT algorithms break down the impedance matrix into a set of smaller Toeplitz and Hankel matrices, each of which is defined by only a single row and column. However, in terms of preconditioning and optimization space, the number of unknowns remains N.

PFFT was developed as a solution to allow for off-grid basis functions such as those in [15]. This allows for more efficient meshing of the structure and an associated reduction in unknowns, and for potentially increased speedup for some geometries even as compared with CG-FFT, although, it is unfortunately not possible to do so without conceding some degree of accuracy of the result. PFFT accomplishes this capability for off-grid basis functions by *projecting* the geometry onto another FFT grid and performing a traditional CG-FFT-style matrix vector product, then *interpolating* the results back onto the original geometry. While this is a good approximation for far interactions, it is not a good

approximation for near interactions. Thus, it is necessary to do a "pre-correction," (the namesake of the technique), whereby interpolation results for near interactions are replaced by accurately computed results, as

$$\mathbf{Z}\cdot\mathbf{I}=\mathbf{Z}_{MoM}^{near}\cdot\mathbf{I}+\hat{\mathbf{Z}}^{far}\cdot\mathbf{I}=\mathbf{Z}_{MoM}^{near}\cdot\mathbf{I}+(\hat{\mathbf{Z}}\cdot\mathbf{I}-\hat{\mathbf{Z}}^{near}\cdot\mathbf{I}). \tag{3.1-1}$$

In this sense, PFFT can more accurately be viewed as being $O(M_b N_b log\ M_b N_b)$ + $O(N^2{}_{pre-corr.})$, thus, it is actually less efficient (in addition to being less accurate) for geometries which do not require non-grid conforming subsections.

An important distinction between UFFT-GT and UFFT-P is that the UFFT-P codebase was originally written in FORTRAN, while the newer UFFT-GT codebase is written in MATLAB for coding efficiency. Thus, when comparing the two, there is an additional factor to consider. Porting the UFFT-P codebase to MATLAB would provide more meaningful results for performance, however, that is a large and difficult task beyond the scope of this work. Moreover, the main comparison between the two is the accuracy of the solvers, to which the difference in language contributes significantly less than the algorithm itself.

3.1a Accuracy Improvements of UFFT-GT

The most notable improvement made transitioning from UFFT-P to UFFT-GT is accuracy. Notably, full precision as compared to Sonnet [2], an extremely accurate numerical electromagnetics tool [10] is desired, as this is fundamentally the most accuracy possible while using Sonnet's FFT-accelerated MFO. While it is certainly possible to make a UFFT implementation based on other MFO techniques, there are no other FFT-accelerated

methods available for use, and developing or getting access to another FFT-accelerated MFO would significantly delay the development of a unified technique. Sonnet cedes only two components to accuracy—the first, the discretization, is endemic to most or all forms of numerical electromagnetics and well controlled by subsection density. The second is numerical precision or quantization, which is endemic to the machine on which computation is done, typically 64-bit.

UFFT-P has these limitations but introduces a further set of three limitations:

1. Iterative solver convergence (the residual between the calculated voltage array of the last iteration and the initial port excitation voltage array).
2. Impedance matrix element error due to projection of the geometry.
3. Impedance matrix element error due to interpolation of the geometry.

These are all removed in part or in whole with the UFFT-GT algorithm, which eliminates the projection entirely and can benefit from better convergence due to the removal of noise.

3.1b Memory Usage Improvements

While memory usage of UFFT-P performs better than conventional Sonnet as well as an MLFMA solver as demonstrated in Chapter 4, there is room for improvement. The pre-correction of matrix elements necessitates that the matrix be stored implicitly (in the set of Toeplitz/Hankel matrices detailed in Chapter 2), but also the pre-correction sparse matrix needs to be stored. This matrix typically has a density that is a small percentage of the $O(N^2)$ matrix—typically around 1% in the examples of Chapter 4—however, this can be a

large percentage of the implicitly stored matrix and thus leaves a large amount of room for improvement. UFFT-GT does not need to store any pre-correction matrix thus reducing memory requirements, although this advantage is somewhat mitigated by the fact that UFFT-GT still does require storage of a preconditioner sparse matrix.

3.1c Multilayer and 3D Planar Support

The UFFT-P implementation is limited in that it only supports simulation of planar geometries. This is a limitation of the developed codes as opposed to theory. Given the many other benefits of researching UFFT-GT, research on UFFT-P was halted for UFFT-GT. As such, the ability to simulate many layered structures with interconnecting vias was not added to the UFFT-P code, though there is still value in such a development as UFFT-P allows for off-grid discretization. 3D planar support has been added to the UFFT-GT code, however, and is detailed in Sections 3.3c-d.

3.2 Generalized Minimum Residual Method (GMRES) Specifics

It is noteworthy that both PFFT and CG-FFT are methods to perform fast matrix products. As such, to form a complete solution to the EFIE, it is necessary to perform these accelerated matrix vector products inside of an iterative framework that converges to a solution.

Typically, numerical electromagnetics solvers have been *direct*. This is to say that, e.g., for MoM where matrix equation $[A][x]=[b]$ is cast into $[Z][i]=[v]$, the matrix $[Z]$ is

calculated and inverted. $[Z]^{-1}$ is *directly* calculated, which allows for the solution $[i]=[Z]^{-1}[v]$ to be calculated easily.

By contrast, *iterative* solvers calculate $[Z][i_{iter}]=[v_{iter}]$ inside of an optimization loop, where many $[Z][i_{iter}]$ matrix vector products are calculated with $[i_{iter}]$ being adjusted until $[Z][i_{iter}]\cong[v]$.

3.2a Overview

The Generalized Minimum Residual method (GMRES) is an extremely popular iterative solver algorithm. It is a Krylov subspace method that can be used to solve systems of linear equations. It is most often applied to sparse matrices, however it can be applied to dense matrices, like those from MoM, as well. As applied to MoM, it calculates $[Z][i_{iter}]=[v_{iter}]$ many times, and at each iteration, produces a residual, $r_{iter}=[Z][i_{iter}]-[v]$, which is minimized throughout the process. The solver is driven to continue until r_{iter}, essentially an error term, reaches some prescribed level [74]. A common setup within electromagnetics might be $r_{iter}=10^{-5}$ to achieve accuracy of about 1% in interactions at a level of -60dB. However, useable high error results can often be obtained with a residual as high as $r_{iter}=10^{-3}$. With UFFT-GT, it is commonly possible utilize results of converge to $r_{iter}=10^{-12}$ or better, as both MFO and MSO are accurate to this precision.

Maxwell's equations can easily give rise to matrices which are not *well conditioned*. This means that reaching a tightly defined tolerance may require excessive iterations, or worse, could be impossible to reach. As GMRES convergence is monotonic, any iteration that does not improve the residual denotes a stagnation—a failure which halts the algorithm [74]. So as to ensure good convergence, it is often necessary to improve the condition of the matrix through use of a preconditioner.

3.2b Preconditioners

Preconditioners are essentially sub-algorithms, which generate a sparse (or more sparse) representation of the matrix that is applied to the main matrix so as to attempt to improve the condition number of the matrix. During iteration, they are applied to the matrix, attempting to move eigenvalues to locations where GMRES can more easily converge [74], i.e., narrowing the spectrum of the matrix, although there is no guarantee of efficacy for any given preconditioner type. Preconditioners are often applied as matrices, M, filled up as the solutions of sparse representations of the impedance matrix $[Z]$. They are multiplied by the matrix vector product, as

$$[M]^{-1}[Z][i] = [M]^{-1}[v]. \tag{3.2-1}$$

As the solution of the preconditioner itself must be calculated, stored, and applied to the matrix at every iteration, the complexity of the preconditioner inversion must be weighed against the benefit provided to convergence for maximum speed increase. This often results in not only sparse matrix representations, but also methods to solve sparse matrices that themselves produce sparse matrices—the full inverse of a sparse matrix itself

is often far denser. Note that there are multiple methods for the solution of [*M*]. These include direct LU factorization, incomplete-LU factorization, sparse-approximate inverse, and more [74].

The most simplistic yet still effective preconditioner for MoM is generally the Jacobi preconditioner,

$$\left[M_{Jacobi}\right] = diag\left(\left[Z\right]\right). \tag{3.2-2}$$

The Jacobi preconditioner is a very sparse matrix, where inverse is easy to calculate and is equally sparse. Further, it includes all self-impedance terms of the moment matrix as they are all on the diagonal. However, for large and complicated problems, it is seen in this work to not be sufficient for good convergence, as the number of iterations required to converge approaches the number of unknowns, and thus impacts scaling too greatly. In general for this work, a preconditioner is considered effective if it can converge in fewer than 50 iterations. Unfortunately, no examples in this work meet this criterion with no preconditioner or even a Jacobi preconditioner.

A more effective preconditioner, particularly for problems that are on the order of a wavelength or more, is the near-interaction preconditioner. The near-interaction preconditioner identifies all subsection interactions that are less than a threshold wavelength, e.g., $\frac{\lambda}{10}$, apart. All impedance matrix elements that meet this criterion become part of a sparse matrix, from which a solution is generated. The near part preconditioner is

used most often in this work, and unless explicitly stated otherwise can be assumed for use in all examples. The density of an example near-interaction preconditioner is seen in Fig. 3.2-1 for the 4 000 unknown variant of the example demonstrated in Section 4.1. Preconditioner density for examples considered in this work ranges from approximately 0.1% to 2%.

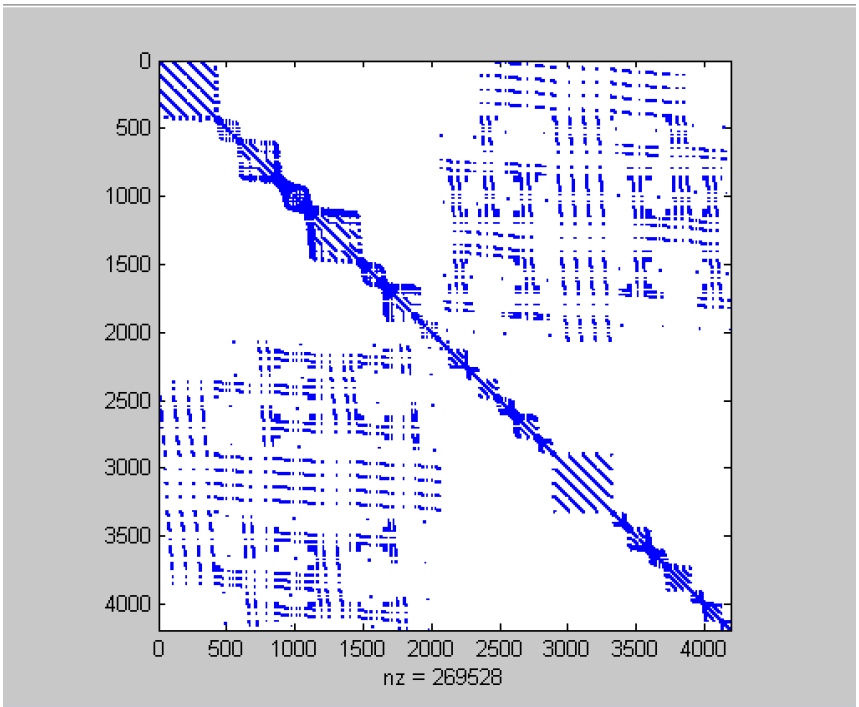

Fig. 3.2-1. The density of the near part preconditioner for the example in Section 4.1.

Convergence of the resultant matrix equation generally is faster as the radius of near subsections is increased. Ultimately the near radius can be always made sufficiently large to ensure convergence in any desired number of iterations. Indeed, taking the entire matrix as preconditioner ensures convergence in a single iteration. Even though the general trend is for the larger radius to produce a faster convergence of the iterative solver, in some particular cases this trend does not hold for reasonable near radius values and becomes

apparent only when near radius becomes excessively too large to remain practical. In addition to storage considerations, larger preconditioners can make the time spent applying the preconditioner—which must be done at each iteration—surpass that of the matrix vector product. Thus, it is best to use as small a radius as possible that will still achieve convergence.

The condition of the matrix, with subsectioning constant, generally gets worse as frequency goes down, as shown in Fig. 3.2-2. As the geometrical size of subsections is unimportant when compared to the electrical size, this is effectively demonstrating performance with frequency as well as performance with discretization density, without influence from the number of unknowns. Note that this data is for the capacitor example in Section 4.2, and that data is a courtesy of Matt Thelen of [2]. The figure contains data for multiple simulations across a broad frequency band and is not the convergence of a single simulation. Note that the flat line at 122 total iterations represents an iteration cutoff limit; convergence is not achieved at these frequencies.

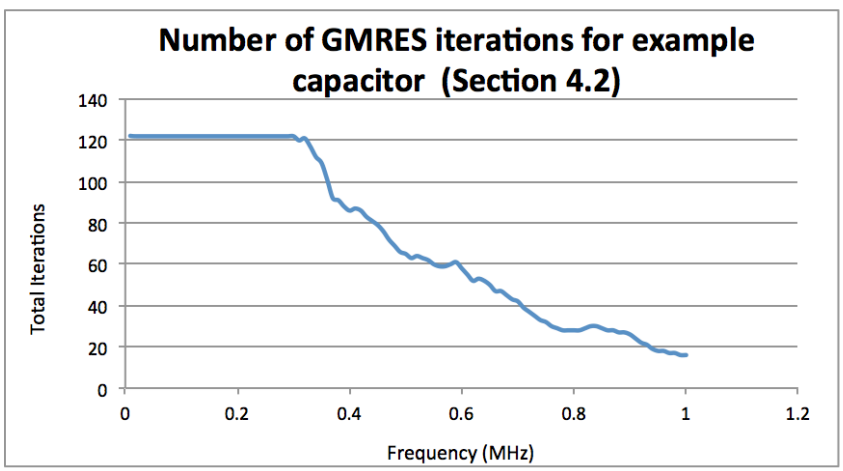

Fig. 3.2-2. Convergence of the example in Section 4.2 vs. frequency. Data courtesy Matt Thelen of [2].

Small electrical size of the structure inevitably causes small electrical size of the basis functions. A key idea of [75] is leveraging the Calderon identity to turn the unbounded spectrum of the EFIE to the bounded one. This eliminates the low frequency breakdown described here, which is not mitigated by the near-interaction preconditioner. As such, it is also generally advisable to use as coarse a grid as possible to achieve the desired accuracy. In practice, it is not a suitable solution to increase the frequency of simulation to achieve better convergence. In this work, however, it is used sometimes as a short-term solution to improve convergence for more finely discretized examples. This is considered acceptable only because this work focuses mainly on algorithmic efficiency as opposed to the efficacy of preconditioners. To mitigate the effect of low frequency breakdown for simulation in practice standard techniques based on loop-tree decomposition [83] of the MoM basis functions and the Calderon Multiplicative Preconditioner can be applied for this work.

It is further noteworthy that, depending on the numbering of the subsections, the form of the sparse matrix can be radically shifted. Important and close interactions do not necessarily end up close to the self-impedance diagonal of the matrix. However, sparse matrix algorithms generally perform better when the sparse matrix density is closer to the diagonal [76]. As mentioned previously, the inverse of a sparse matrix is often denser than the matrix itself when, e.g., the first row or column is very dense. Thus, for maximum performance of the preconditioner, it is often beneficial to sort the matrix to achieve a more sparse solution, particularly if the density of the matrix exceeds approximately 15% as has been found in this work. Fig. 3.2-3 shows the same 269 628 non-zero element

preconditioner as Fig. 3.2-1, albeit sorted such that elements are closer to the diagonal. This is sometimes called a "banded" version of the matrix. A popular algorithm for such matrix banding is found in [77].

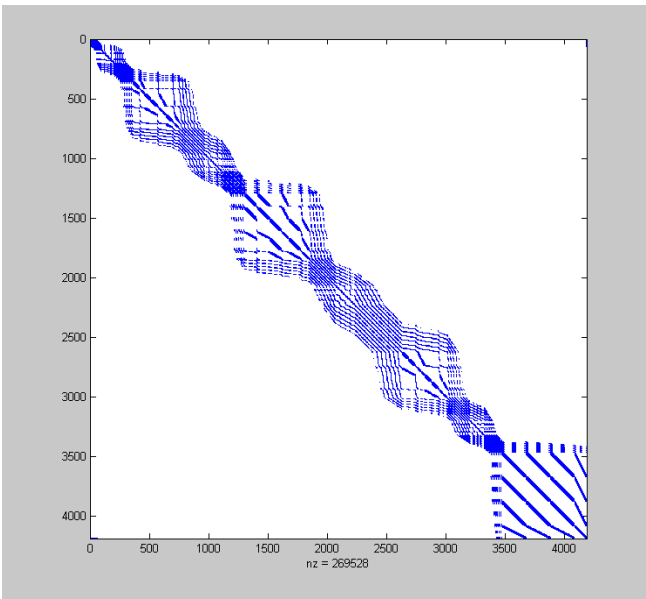

Fig. 3.2-3. The same preconditioner as Fig. 3.2-1, however, it is sorted to produce a banded matrix.

There are any number of possible types of preconditioners that can be made specific to the type of problem. When creating such preconditioners, it should be possible to use singular value decomposition [76] (a.k.a., principal component analysis), essentially, a matrix decomposition that gives information about the respective eigenvalues, to gauge the potential usefulness.

3.3 Conjugate Gradient Method and UFFT-GT

It is notable that, as iterative methods require only the result of the matrix-vector product, $[v]=[Z][i]$, it is possible to calculate this result without ever storing $[Z]$ explicitly. This type

of method is called a "matrix-free" algorithm. Through utilization of the periodicity of the DGF as applied to MoM, it is possible to decompose the matrix into a set of Toeplitz and Hankel ordered arrays. These arrays can be vector-vector multiplied with [*i*], along with some signal processing, to produce the desired result in $O(N_b M_b \log N_b M_b)$ operations [76]. Although presented first due to the more simplistic nature of the algorithm, it is noteworthy that UFFT-GT was developed secondary to UFFT-P (Section 3.4) as a vehicle to reduce simulation error. Additionally, it was implemented as a completely new codebase.

3.3a Implicit Matrix Setup, Structure, and Multiplies for MSOs

The electric field integral equation (EFIE) for this implementation of UFFT-GT starts in the form

$$\hat{\mathbf{t}} \cdot \int_{y'=0}^{b} \int_{x'=0}^{a} \overline{\overline{G}}(x,x',y,y') \cdot \overline{J}(x',y') dx' dy' = \hat{\mathbf{t}} \cdot \overline{E}_{inc}(x,y), \tag{3.3-1}$$

where $\hat{\mathbf{t}}$ is the tangential unit vector to the surface of the structure, $x \in [0,a]$, $y \in [0,\mathrm{b}]$, and $\overline{\overline{G}}(x,x',y,y')$ is the dyadic Green's function of the multilayered medium situated transversally to the axis of a rectangular waveguide. The Green's function can be represented as a sum of four terms according to the image theory [24] as follows;

$$\begin{aligned}\overline{\overline{G}}(x,x',y,y') &= \overline{\overline{G}}^{--}(x-x',y-y') + \overline{\overline{G}}^{-+}(x-x',y+y') \\ &+ \overline{\overline{G}}^{+-}(x+x',y-y') + \overline{\overline{G}}^{++}(x+x',y+y').\end{aligned} \tag{3.3-2}$$

where the dyadic product representation of $\overline{\overline{G}}$ for planar examples is

$$\overline{\overline{G}} = \hat{\mathbf{x}}\hat{\mathbf{x}} G_{xx} + \hat{\mathbf{x}}\hat{\mathbf{y}} G_{xy} + \hat{\mathbf{y}}\hat{\mathbf{x}} G_{yx} + \hat{\mathbf{y}}\hat{\mathbf{y}} G_{yy} + \ldots, \tag{3.3-3}$$

noting that there are five terms involving *z* components not shown as they are unused and not calculated or stored for planar examples. The current discretization is of the form

$$\overline{J}\left(x',y'\right)=\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1}J_{m'n'}^{x}\overline{B}^{x}\left(x'-x_{m'},y'-y_{n'}\right)+\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1}J_{m'n'}^{y}\overline{B}^{y}\left(x'-x_{m'},y'-y_{n'}\right), \qquad (3.3\text{-}4)$$

with the circuit being discretized over the entire M_b x N_b box, where $J_{m'n'}$ is the weighted residual within MoM, and $\overline{B}$ are the basis functions. While $J_{m'n'}$ is inherently zero at indices where there is no metallization, all possible basis functions ($\overline{B}$) are used at this level so as to leverage their periodicity. "Empty" basis functions where there is no current, however, are not treated as unknowns with respect to the overall solution. Combining (3.3-4) with (3.3-1), effectively applying our linear operator (EFIE) to the discretization, yields a discretized form of the EFIE featuring a dyadic Green's function (DGF),

$$\hat{\mathbf{t}}\cdot\int_{y'=0}^{b}\int_{x'=0}^{a}\overline{\overline{G}}\cdot\left[\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1}\left(J_{m'n'}^{x}\overline{B}_{m'n'}^{x}+J_{m'n'}^{y}\overline{B}_{m'n'}^{y}\right)\right]dx'\,dy'=\hat{\mathbf{t}}\cdot\overline{E}_{inc}\left(x,y\right). \qquad (3.3\text{-}5)$$

The MoM is implemented as a Galerkin method [1] so as to benefit from electromagnetic reciprocity and resultant matrix symmetry, and applying (3.3-5) as an inner product with the same function used for basis function results, for x and y,

$$\int_{y=0}^{b}\int_{x=0}^{a}\left(\overline{B}_{mn}^{x}\right)\cdot\int_{y'=0}^{b}\int_{x'=0}^{a}\overline{\overline{G}}\cdot\left[\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1}\left(J_{m'n'}^{x}\overline{B}_{m'n'}^{x}+J_{m'n'}^{y}\overline{B}_{m'n'}^{y}\right)\right]dx'dy'dx\,dy$$
$$=\int_{y=0}^{b}\int_{x=0}^{a}\left(\overline{B}_{mn}^{x}\right)\cdot\overline{E}_{inc}\left(x,y\right)dxdy$$
$$\int_{y=0}^{b}\int_{x=0}^{a}\left(\overline{B}_{mn}^{y}\right)\cdot\int_{y'=0}^{b}\int_{x'=0}^{a}\overline{\overline{G}}\cdot\left[\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1}\left(J_{m'n'}^{x}\overline{B}_{m'n'}^{x}+J_{m'n'}^{y}\overline{B}_{m'n'}^{y}\right)\right]dx'dy'dx\,dy \quad , \qquad (3.3\text{-}6)$$
$$=\int_{y=0}^{b}\int_{x=0}^{a}\left(\overline{B}_{mn}^{y}\right)\cdot\overline{E}_{inc}\left(x,y\right)dxdy$$

where the right-hand-side (RHS) of (3.3-6) reduces simply to the voltage V_{mn}^{x}, and V_{mn}^{y}, respectively. In order to arrive at a linear operator that may be applied in the fashion of an impedance matrix, (3.3-6) is reordered as

$$\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1}\left(J_{m'n'}^{x}\int_{y=0}^{b}\int_{x=0}^{a}\overline{B}_{mn}^{x}\cdot\int_{y'=0}^{b}\int_{x'=0}^{a}\overline{\overline{G}}\cdot\overline{B}_{m'n'}^{x}\,dx'dy'dx\,dy\right)$$
$$+\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1}\left(J_{m'n'}^{y}\int_{y=0}^{b}\int_{x=0}^{a}\overline{B}_{mn}^{x}\cdot\int_{y'=0}^{b}\int_{x'=0}^{a}\overline{\overline{G}}\cdot\overline{B}_{m'n'}^{y}\,dx'dy'dx\,dy\right)=V_{mn}^{x}$$
$$\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1}\left(J_{m'n'}^{x}\int_{y=0}^{b}\int_{x=0}^{a}\overline{B}_{mn}^{y}\cdot\int_{y'=0}^{b}\int_{x'=0}^{a}\overline{\overline{G}}\cdot\overline{B}_{m'n'}^{x}\,dx'dy'dx\,dy\right) \quad . \qquad (3.3\text{-}7)$$
$$+\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1}\left(J_{m'n'}^{y}\int_{y=0}^{b}\int_{x=0}^{a}\overline{B}_{mn}^{y}\cdot\int_{y'=0}^{b}\int_{x'=0}^{a}\overline{\overline{G}}\cdot\overline{B}_{m'n'}^{y}\,dx'dy'dx\,dy\right)=V_{mn}^{y}$$

The integrals in (3.3-7) are more conveniently represented as impedance matrices, *Z*, as in

$$\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1}\left[Z_{mnm'n'}^{xx}J_{m'n'}^{x}+Z_{mnm'n'}^{xy}J_{m'n'}^{y}\right]=V_{mn}^{x}$$
$$\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1}\left[Z_{mnm'n'}^{yx}J_{m'n'}^{x}+Z_{mnm'n'}^{yy}J_{m'n'}^{y}\right]=V_{mn}^{y} \quad . \qquad (3.3\text{-}8)$$

Due to decomposition of the Green's function according to (3.3-3) each of the '*xx*', '*xy*', '*yx*', and '*yy*' blocks the above system of linear algebraic equations (SLAE) can be cast into the form of 2D correlations/convolutions as demonstrated below for the '*xx*' component of the SLAE:

$$V_{mn}^{xx} = \sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1} Z_{m,n,m'n'}^{xx} J_{m'n'}^{x} =$$
$$\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1} \mathrm{Z}_{m-m',n-n'}^{xx} J_{m'n'}^{x} + \sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1} \mathrm{Z}_{m-m',n+n'}^{xx} J_{m'n'}^{x} +$$
$$\sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1} \mathrm{Z}_{m+m',n-n'}^{xx} J_{m'n'}^{x} + \sum_{m'=0}^{M_b-1}\sum_{n'=0}^{N_b-1} \mathrm{Z}_{m+m',n+n'}^{xx} J_{m'n'}^{x} \qquad (3.3\text{-}9)$$

It is now beneficial to examine the terms for the impedance matrix in detail, from which the periodicity necessary to perform the proposed CG-FFT algorithm hails. As an example, we find elements which couple *x*-directed sources to *x*-directed observation points corresponding to the $G_{xx}\left(x-x',y-y'\right)$ component of the dyadic Green's function,

$$Z_{m-m',n-n'}^{xx} = \int_{y_n-\triangle y}^{y_n+\triangle y}\int_{x_m-\triangle x}^{x_m+\triangle x}\left(\overline{B}^{x}\left(x-x_m,y-y_n\right)\right)\cdot$$
$$\int_{y_{n'}-\triangle y}^{y_{n'}+\triangle y}\int_{x_{m'}-\triangle x}^{x_{m'}+\triangle x}\overline{\overline{G}}\left(x-x',y-y'\right)\cdot\overline{B}^{x}\left(x'-x_{m'},y'-y_{n'}\right)dx'\,dy'\,dx\,dy \quad . \qquad (3.3\text{-}10)$$

Letting $\xi = x - x_m$, $\eta = y - y_n$, $\xi' = x' - x_{m'}$, $\eta' = y' - y_{n'}$, and $dx = d\xi$, $dy = d\eta$, $dx' = d\xi'$, $dy' = d\eta'$, (3.3-10) can be rewritten as

$$\mathrm{Z}^{xx}_{m-m',n-n'} = \int_{-\triangle y}^{\triangle y}\int_{-\triangle x}^{\triangle x} d\xi\,\mathrm{d}\eta\,\overline{B}^{x}\left(\xi,\eta\right)\cdot$$
$$\int_{-\triangle y}^{\triangle y}\int_{-\triangle x}^{\triangle x} \overline{\overline{G}}\left[\left(\xi+x_m\right)-\left(\xi'+x_{m'}\right),\left(\eta-y_n\right)-\left(\eta'-y_{n'}\right)\right]\cdot\overline{B}^{x}\left(\xi',\eta'\right)d\xi'\mathrm{d}\eta', \qquad (3.3\text{-}11)$$

which is further rewritten so as to produce convolution-correlation pairs as

$$Z^{xx}_{m-m',n-n'} = \int_{-\triangle y}^{\triangle y}\int_{-\triangle x}^{\triangle x} d\xi\,\mathrm{d}\eta\,\overline{B}^{x}\left(\xi,\eta\right)\cdot$$
$$\int_{-\triangle y}^{\triangle y}\int_{-\triangle x}^{\triangle x} \overline{\overline{G}}\left[\xi-\xi'+\Delta x(m-m'),\eta-\eta'+\Delta y(n-n')\right]\cdot\overline{B}^{x}\left(\xi',\eta'\right)d\xi'\mathrm{d}\eta'. \qquad (3.3\text{-}12)$$

With a complete $Z^{xx}_{\mu\nu}$ calculated and stored, it is conveniently possible to exploit the periodicity of $\xi = x - x_m$, $\xi' = x' - x_{m'}$ $\eta = y - y_n$ and $\eta' = y' - y_{n'}$, and as a result produce any element of $\mathrm{Z}^{xx}_{m-m',n-n'}$ with three arithmetic operations and four memory accesses from data which is not sequentially indexed. Further, this process is only necessary for sparse preconditioner elements that need to be explicitly stored. The dense matrix need not be stored with the CG-FFT algorithm as the periodic storage of the data can further be exploited to accelerate matrix vector products via FFT. Note that this precludes the use of full matrix inverse with this technique, however it does allow for the use of iterative methods. To explain, first it is beneficial to demonstrate full $O(N_b^2M_b^2)$ matrix vector product which explicitly calculates each matrix element and performs directly the 2D convolutions and correlations below,

$$\mathrm{Z}_{++}^{xx} * J_{book}^{x} = \sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} Z_{m+m',n+n'}^{xx} J_{m'n'}^{x}$$

$$\mathrm{Z}_{+-}^{xx} * J_{reverse}^{x} = \sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} Z_{m+m',n-n'}^{xx} J_{m'n'}^{x}$$

$$\mathrm{Z}_{-+}^{xx} * J_{reverse}^{x} = \sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} Z_{m-m',n+n'}^{xx} J_{m'n'}^{x}$$

$$\mathrm{Z}_{--}^{xx} * J_{book}^{x} = \sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1} Z_{m-m',n-n'}^{xx} J_{m'n'}^{x}\ , \qquad (3.3\text{-}13)$$

where $Z_{--}^{xx}, Z_{+-}^{xx}, Z_{-+}^{xx}$, and Z_{++}^{xx} are simply 1D re-orderings of $Z_{m-m',n-n'}^{xx}$,$Z_{m+m',n-n'}^{xx}$, $Z_{m-m',n+n'}^{xx}$, and $Z_{m+m',n+m'}^{xx}$, respectively, with appropriate and selective padding so as to preserve the Toeplitz and Hankel structures for the 1D ordering. J_{book}^{x} and $J_{reverse}^{x}$ are both 1D re-orderings of $J_{m'n'}^{x}$, where the former re-orders basis functions as though they are lines in a book and the later organizes them as though the lines are reversed down the page, as shown in Fig. 3.3-1, which shows an ordering of the basis functions for an extremely coarse 2 by 2 grid with 6 potential basis function locations—at indices 1, 2, 3, 6, 7, and 8. They also feature a second type of padding, doubling the length with zeros so as to allow matrix-vector products with the circulant form of the Toeplitz and Hankel matrices. Each numbered rectangle represents the center of an *x*-directed rooftop basis function on the 2D grid, with the number being its associated index in each of the 1D current arrays. Note that the array spans twice the cross-section of the box to ensure Toeplitz and Hankel structure of the implicit matrix.

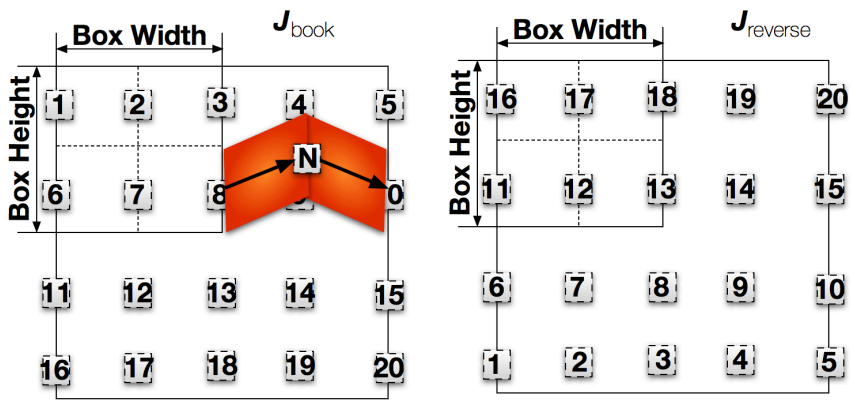

Fig. 3.3-1. Indexing for the two 1D current arrays. Each number represents a rooftop basis function.

Matrix vector products are enhanced to $O(N_b M_b \log N_b M_b)$ by performing a series of 1D discrete convolutions through the use of digital signal processing [61], with the *x* source and *x* observation coupling producing,

$$
\begin{aligned}
&V^{xx} = \\
&IFFT\left\{FFT\left\{Z_{++}^{xx}\right\} \circ FFT\left\{J_{book}^{x}\right\}\right\} \\
&+IFFT\left\{FFT\left\{Z_{+-}^{xx}\right\} \circ FFT\left\{J_{reverse}^{x}\right\}\right\} \\
&+IFFT\left\{FFT\left\{Z_{-+}^{xx}\right\} \circ FFT\left\{J_{reverse}^{x}\right\}\right\} \\
&+IFFT\left\{FFT\left\{Z_{--}^{xx}\right\} \circ FFT\left\{J_{book}^{x}\right\}\right\}
\end{aligned}
\quad , \qquad (3.3\text{-}14)
$$

Using 1D orderings and FFTs is unconventional, typically 2D orderings are used, which implement four terms for J^x as opposed to two. The increased time to perform 1D FFTs is negated by the need to do fewer of them, whilst preserving memory. Note that, unlike UFFT-P, the MFO is no longer split between a gridded set of point source basis and testing functions (which require the geometry projection and interpolation to use), and the set of

full rectangular rooftop basis functions (necessary for pre-correction due to error from the projection and interpolation). With this work, it is made possible to fill the implicit matrix directly with full rectangular rooftop basis functions, negating the need for projection and interpolation as discussed at the beginning of Section 2.3. An explicitly-defined *near-interaction* matrix is still generated for pre-conditioning purposes.

The generalized minimum residual (GMRES) iterative method [74] is used to provide a solution in a fashion most similar to existing CG-FFT implementations, e.g., [24], though very often Conjugate Gradient method may be used.

Note that although no traditional matrix elements (i.e., impedances) are directly calculated or stored, for the *near-interaction* matrix used for pre-conditioning purposes they are available through the summation of four array elements as discussed in Section 2.5. As such, solving a sparse matrix representation of geometrically close interactions in the circuit forms the *near-interaction* pre-conditioner.

As the implicitly stored matrix is a set of Toeplitz and Hankel arrays encompassing all interactions possible on the grid, and the current array that is generated by GMRES is dense and represents only interactions between elements where metallization is present, a sparse transform of the current array is made. The Toeplitz and Hankel matrices are then converted to circulant form, which is shown in [26] to allow matrix vector multiply calculations with the use of FFT.

Thus, it is possible to compute the matrix vector multiplies $\boldsymbol{V} = \boldsymbol{Z}_{full} \cdot \boldsymbol{I}$ with implicitly stored matrix elements by calculating and summing

$$\boldsymbol{V}_{intermediate} = FFT^{-1}\big(FFT(\boldsymbol{Z}_{circ}) \cdot FFT(\boldsymbol{J}_{circ})\big), \tag{3.3-15}$$

for each of the $N_{arrays} = 4DP^2$ arrays, where $\boldsymbol{Z}_{circ}$ are circulant representations of the matrices $\boldsymbol{Z}_{++}$,$\boldsymbol{Z}_{++}$, $\boldsymbol{Z}_{++}$, or $\boldsymbol{Z}_{++}$ from (3.3-14), and $\boldsymbol{J}_{circ}$ are padded versions of either $\boldsymbol{J}_{book}$ or $\boldsymbol{J}_{reverse}$ from (3.3-14). This process is repeated for a new full current vector $\boldsymbol{I}$ at every iteration of GMRES. Once all matrix-vector multiplies are completed, the sparse representation of the voltage vector $\boldsymbol{V}_{intermediate}$ is converted back to the dense representation which is equivalent numerically to $\boldsymbol{V}$.

As stated previously in this section, the FFTs in (3.3-15) are being calculated unconventionally as one-dimensional. It is conventional to solve them as two-dimensional FFTs for a decrease in FFT compute time; however, this involves storing and computing at each iteration a set of four current arrays, while the 1D FFTs use two. Thus, it saves memory for a slight impact on performance. It is shown in Fig. 3.3-2 that 2D FFTs are slightly more than twice as fast as 1D FFTs within MATLAB. The 2D times also appear to be more stable vs. FFT length than the 1D times.

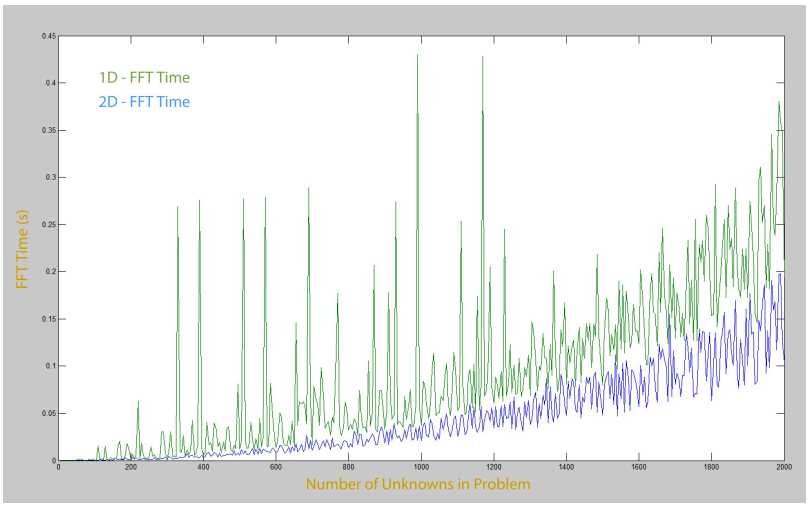

Fig. 3.3-2. 2D vs. 1D FFT Time, with 2D being more stable and approximately 2x faster.

3.3b Discussion of Error Terms in UFFT-GT

A key contribution of UFFT-GT is fast characterization of a device without loss of accuracy compared to the direct MoM solution. In UFFT-GT, there are three error components as compared to conventional Sonnet that all must demonstrate near-equivalency to numerical noise to make this claim. These components are:

1. Error from performing an iterative solve (GMRES)
2. Error from implicit storage of the matrix
3. Error from implicit matrix vector multiplies with FFT

An error budget for these terms is calculated. The first error component is demonstrated to reliably perform better than 10^{-10}, and commonly 10^{-12}, however it is seen to be dependent on the kernel of individual problems that are simulated. The second and third error components are shown in this work to reliably demonstrate error residual of 10^{-12} or better, nearly within numerical precision.

To make such a claim, it is important to quantify each type of error. For the first error type, iterative solve error, a simple example shown in Fig. 3.3-3 is considered.

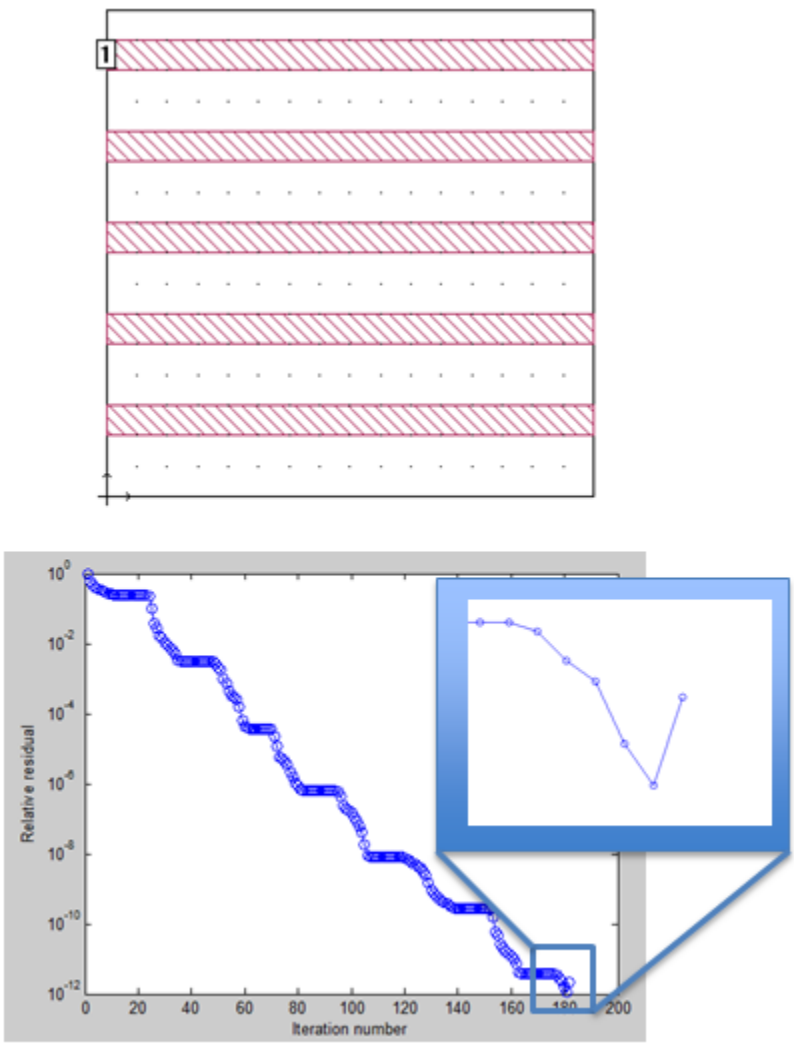

Fig. 3.3-3. A simple thru-line example and the associated convergence limit, which is demonstrated by the non-monotonic behavior.

By using such a simple example the condition of the matrix is kept good so as to keep preconditioning from affecting results. As GMRES converges monotonically—any iteration

worse than the preceding indicates that the algorithm can do no better—it is possible to set the desired residual to an impossibly small number, ensuring that GMRES will fail at the maximum possible precision. Results of this simulation are also shown in Fig. 3.3-3, with GMRES going non-monotonic and failing at a residual smaller than 10^{-11}. This proves that it is possible for GMRES to reach close to numerical precision (approximately 10^{-15} depending on data types). Indeed, some examples in Chapter 4 show even better convergence.

For the second error type—error that is due to the implicit storage of the matrix—the matrix-vector multiplies are carried out as a set of full matrix-vector multiplies (i.e., no usage of FFT) on a set of dense, fully reconstructed Toeplitz and Hankel matrices. As a lossless circuit is considered, all real values of matrix elements are zero and the error is restricted to the imaginary component. More specifically for the lossless condition with all real values being zero, there are three conditions.

1. There is no dielectric loss.
2. Conductivity is infinite.
3. The shielding enclosure fully encompasses the circuit and also has infinite conductivity.

Once a voltage array (the result of the matrix-vector multiply) has been calculated, it is then compared with a second full matrix-vector product, where the vector is the same but the matrix is now the original dense MoM matrix. The difference between the two is the error vs. conventional Sonnet due to the implicit storage of the matrix, and is shown on the

z-axis of Fig. 3.3-4. The *x*-axis shows iteration count and the *y*-axis shows the subsection index.

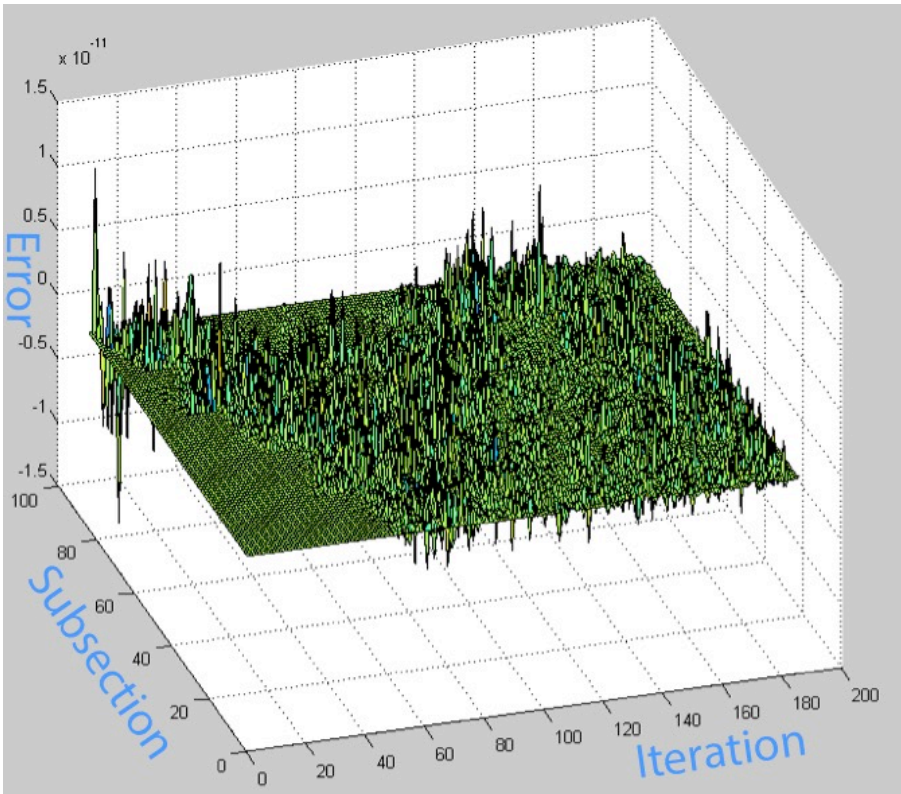

Fig. 3.3-4. Error from the implicit storage of the matrix.

This error term is calculated for each subsection (*y*-axis), and at each iteration of the convergence process (*x*-axis). The maximum error seen is $1.45*10^{-11}$, while average error is $4.6*10^{-13}$. More promising is the shape of the error—it appears to be noise, which further suggests that the root of this error is numerical precision. Note that there are flat spots on the graph at low iteration and low subsection noise; this is due to the way GMRES assigns current to subsections. Early in the process, many subsections are assigned no current and thus exhibit no error.

The third and final type of error is that due to the precision of the FFTs. This error term is calculated by comparing the result of a full $O(N^2)$ matrix-vector product with a set of dense matrices reconstructed from the implicitly stored Toeplitz and Hankel matrices, and the result of an $O(N_bM_b \log N_bM_b)$ matrix vector product calculated with Toeplitz and Hankel arrays via FFT. Results are shown in Fig. 3.3-5, again with iteration on *x*-axis, subsection on *y*-axis, and difference on *z*-axis.

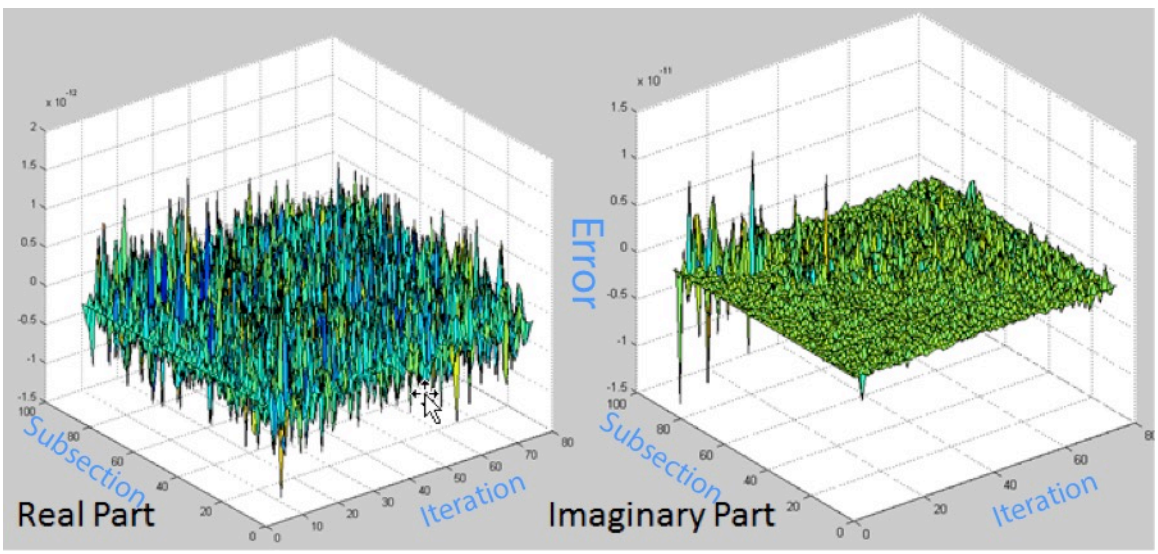

Fig. 3.3-5. Error from the usage of FFTs to solve matrix-vector products. Note that scales are different between the two.

Interestingly, despite a real part that *should* be precisely zero, error is seen almost equally on the real and imaginary parts of the result. This is because the numerical precision error is accumulated regardless of value as the multiplication takes place in the Fourier domain. The maximum error due to this component is shown to be $1.45*10^{-11}$ and the mean error is $6.33*10^{-13}$. Note that in Fig. 3.3-6 the scales are different between real and imaginary data

to better show the distribution. The average error between the two is not significantly different.

Finally, although not a source of error, it is interesting to examine the different path that GMRES takes due to the error terms previously mentioned. Shown in Fig. 3.3-6 is the difference in calculated voltage at every iteration due to error types (2) and (3).

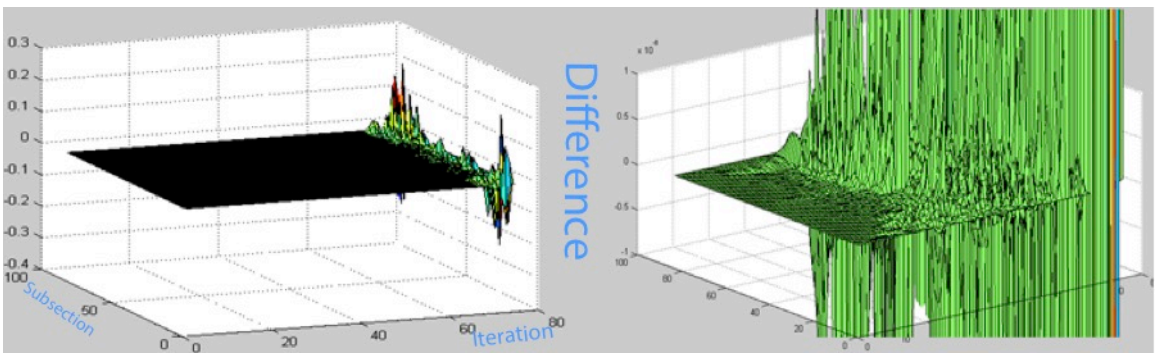

Fig. 3.3-6. Difference at each iteration with and without implicit FFT-based matrix vector products. Macroscopic view at left and expanded *z*-axis view at right.

At left is a macroscopic view, and at right is a view with the *z*-axis greatly expanded to show error in the earlier iterations. The final solution (specifically the last iteration) is identical within the limits previously shown, however each iteration is slightly different, getting worse as iteration count increases. This represents a difference in the path GMRES takes within the optimization space.

This means that the UFFT-GT solver is operating with precision limited, in effect, by the numerical precision of the computer. In this case, conventional 64-bit hardware is used, which is limited to double precision without poor performance. It is expected that with a

128-bit machine able to efficiently perform quad precision calculations, precision could improve up to double.

More generally, within the context of both UFFT solvers, the UFFT-P version has demonstrated considerably less control of error, often with residual norms on the order of 10^{-4} to 10^{-6}. This is because PFFT-based solvers by nature must include all error sources present in the UFFT-GT solver (with exception for the ability to simulate geometries which are off-grid), as well as additional sources of error from the projection and interpolation of the geometry. This error is only partially removed by pre-correction routines. In terms of error percentage, calculated S-parameters generally were within 1-3% of Sonnet results. In terms of current distribution, areas with low interaction (e.g., subsections which are far away from significant sources or circuit activity) might see error up to or above 15% in their calculated current.

3.3c Extension to Multilayer Geometries

In previous sections, both versions of UFFT operate on a single, planar metallization layer at the boundary of two stratified dielectrics spanning the cross-section of a waveguide. UFFT-GT has been further developed to be capable of simulating structures with metallization on multiple planes at the boundaries of stratified media. This is well suited for multi-layer circuit boards and integrated circuits as compared with volume meshing tools, since increasingly complicated dielectric stack-ups do not significantly increase the size of the problem space. Moreover, very thin layers of dielectric and equivalently layers of

metallization close together do not affect the discretization of the circuit geometry or of the grid.

However, this multi-level capability does come at the expense of a minor penalty to scaling. Given N/P unknowns per layer (where N is the number of unknowns and P is the number of metallization planes), interactions must be calculated and stored for all unknowns on each layer for every combination of layers, P^2. Computational complexity thus becomes $O\left(P^2 N_b M_b \log\left(N_b M_b\right)\right)$. Memory is $O(P^2 N_b M_b)$. Since $N_b M_b$ is commonly very large (e.g., $N_b M_b$ = 200 000) as compared with P (e.g., P = 10) this is typically not a major performance penalty, although it remains noteworthy for some thick-metal cases which may require up to P = 200 layers. Notably, structures that feature many metal layers usually confine the bulk of their respective subsections to a few layers. In theory, it should be possible to save time and memory by calculating the contribution of these layers to the matrix-vector product with direct and full multiplies for their respective components rather than explicit FFT-driven multiplies.

A multilayer example geometry, sourced from Greg Kinnetz of [2], is shown in Fig. 3.3-7 with layers of coplanar waveguide (CPW). A multilayer example simulation is demonstrated further in Section 4.5.

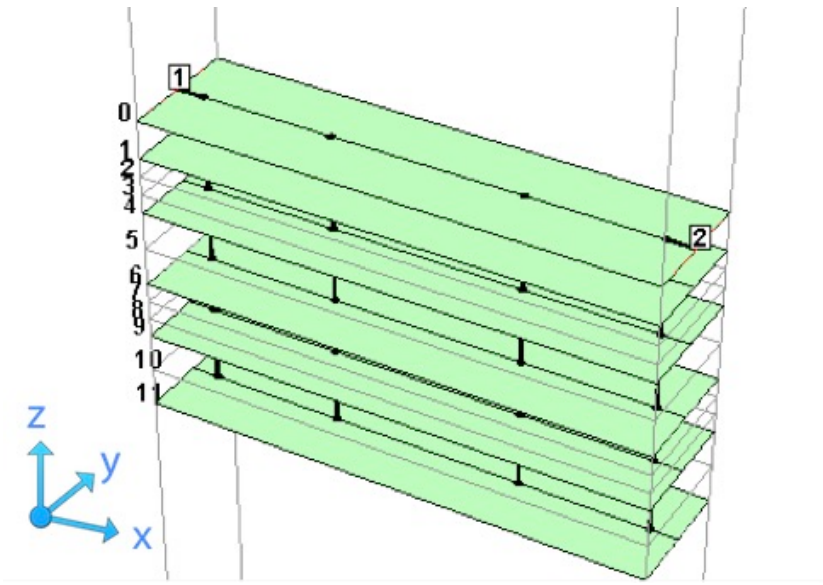

Fig. 3.3-7. A multi-layer example geometry of a co-planar waveguide circuit. Example sourced from Greg Kinnetz of [2].

3.3d Extension to 3D Geometries

Note that there are generally two types of 3D solvers. Arbitrary 3D solvers discretize volumes or surfaces in such a way that the problem geometry is relatively unrestricted in terms of angles, planes, etc. 3D planar solvers are an extension of multiplane solvers, which allow for current to be discretized in the z-direction between planes. For this work, the multiplane solver is extended to a 3D planar solver. Scaling still remains $O\left(P^2 N_b M_b \; log(N_b M_b)\right)$ and memory remains $O(P^2 N_b M_b)$, however, at this point it is no longer possible to ignore the five z-component terms of (3.3-3), and so all terms, between all P^2 layer interactions must be stored as,

$$\begin{aligned}\bar{\bar{G}} = \hat{x}\hat{x}G_{xx} + \hat{x}\hat{y}G_{xy} + \hat{x}\hat{z}G_{xz} \\ +\hat{y}\hat{x}G_{yx} + \hat{y}\hat{y}G_{yy} + \hat{y}\hat{z}G_{yz} \\ +\hat{z}\hat{x}G_{zx} + \hat{z}\hat{y}G_{zy} + \hat{z}\hat{z}G_{zz},\end{aligned} \tag{3.3-16}$$

And in a similar fashion to (3.3-4), current is discretized and stored for each of P layers as,

$$\begin{aligned}\overline{J}\left(x',y'\right)=&\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1}J_{m'n'}^{x}\overline{B}^{x}\left(x'-x_{m'},y'-y_{n'}\right)\\&+\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1}J_{m'n'}^{y}\overline{B}^{y}\left(x'-x_{m'},y'-y_{n'}\right)\\&+\sum_{m'=0}^{M-1}\sum_{n'=0}^{N-1}J_{m'n'}^{z}\overline{B}^{z}\left(x'-x_{m'},y'-y_{n'}\right)\end{aligned}\tag{3.3-17}$$

Note that (3.3-17) does not reflect a z-dependence for the basis functions. This is because z-oriented current, just like planar x- and y-oriented current, may not be at arbitrary locations along the z-axis. Rather, it extends from the plane, P, that it is located on upward to the next plane. If there are no further planes, it extends to the top of the waveguide enclosure. The matrix vector product is calculated by iterating through each set of P^2 layer interactions and using FFT to perform the matrix vector multiply of the impedance matrix data with the appropriate current array. Results of the $4 * 9 * P^2$ multiplies are cumulatively summed in the appropriate voltage array, where four represents the convolution correlation terms and nine represents each component of the DGF. This has implications on the performance of the matrix-vector product, as previously it was only $4 * 5 * P^2$ multiplies.

It also affects memory usage, although, the full P^2 arrays do not need to be stored. Interactions between basis functions on the same axis show reciprocity for level interaction. This means that from level 1 to level 2, the interactions for x-directed source and x-directed observation will be the same as from level 2 to level 1. This is also true for y-

to *y*-, *z*- to *z*-, and even *x*- to *y*- and *y*- to *x*-. It is not, however, true for *x*- to *z*-, *y*- to *z*-, *z*- to *x*-, and *z*- to *y*-, so these arrays must be stored additionally.

Fig. 3.3-8 shows an example of a geometry with *z*-directed currents, in the form of interconnecting vias to approximate a 3D bond wire interconnect using multiple planes with interconnecting vias. Section 4.6 shows the solution of an example geometry.

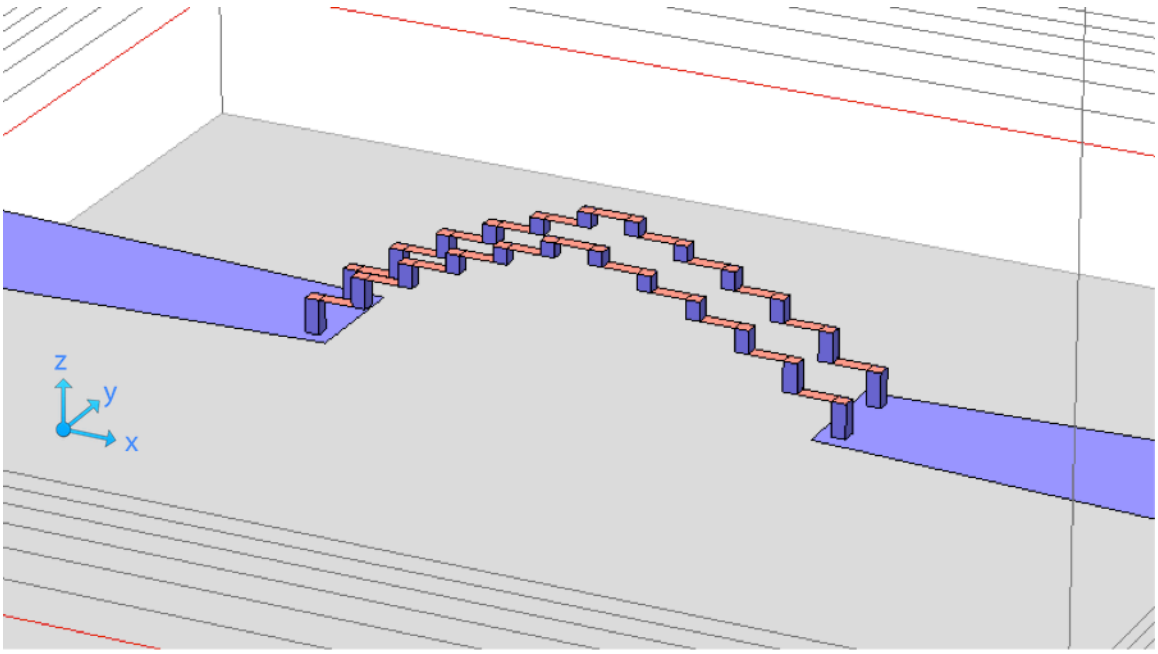

Fig. 3.3-8. An example circuit featuring a bond wire approximation constructed with 3D planar features, including multiple planes and interconnecting vias.

3.4 Geometry Projection and UFFT-P

In the context of MSO for UFFT-P, modification of the PFFT algorithm [3] is used for acceleration of the matrix-vector product $\hat{\mathbf{Z}}^{far} \cdot \mathbf{I}$. It is important to emphasize that the distinction between *near-zone* and *far-zone* interactions in the PFFT algorithm is not based

on electrical distance [56], [58], [78]. Thus, the method can still be applied without loss of effectiveness for structures ranging from electrically large to sub-wavelength [58].

First, a regular rectangular grid is introduced with discrete increments $\Delta x = a/(K_1 - 1)$ and $\Delta y = b/(K_2 - 1)$, which is referred to as the PFFT grid. Unlike UFFT-GT in the previous section, this is a new grid that is distinct from the regular grid used in MFO for off-grid elements. The grid is overlain on the rectangle defined by the intervals $x \in [0, 2a]$ and $y \in [0, 2b]$, corresponding to twice the waveguide cross-section over *x* and *y*. In the following, the indices $k_{1,2} = 0, \ldots, 2(K_{1,2} - 1)$ are used to identify the nodes of the grid. In Fig. 3.4-1, a portion of this grid for $k_{1,2} = 0, \ldots, K_{1,2} - 1$ is shown by circles for the case when the structure under study is a six-patch antenna array. Also shown in Fig. 3.4-1 is the MoM grid that is used for the discretization of the unknown current density on patches. It is noted that the MoM mesh can be non-uniform.

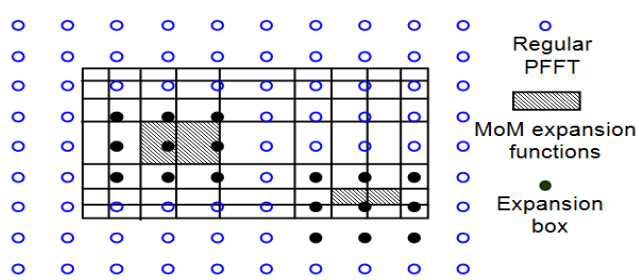

Fig. 3.4-1. Projection of the MoM roof-top basis functions on the regular PFFT grid.

It is stressed that the PFFT grid, like the CG-FFT grid, is introduced over an area four times that of the original cross-section, in order for the DGF used in (2.3-3) to yield all elements of the periodicity [79] for the **Z**-matrix elements. Once the PFFT grid is introduced, each of the basis and testing functions $b_i^\alpha(\vec{r})$ and $t_i^\alpha(\vec{r})$ are replaced by M^2 delta sources, where *M* sets the density of the projection as shown in Fig. 3.4-1. This defines the so-called expansion box (a.k.a. stencil) [3],

$$\hat{b}_i^\alpha(\vec{r}) = \sum_{k_1=0}^{2(K_1-1)} \sum_{k_2=0}^{2(K_2-1)} B_{k_1,k_2,i}^\alpha \delta(x - x_{k_1,i}^\alpha)\delta(y - y_{k_2,i}^\alpha) , \tag{3.4-1a}$$

$$\hat{t}_i^\alpha(\vec{r}) = \sum_{k_1=0}^{2(K_1-1)} \sum_{k_2=0}^{2(K_2-1)} T_{k_1,k_2,i}^\alpha \delta(x - x_{k_1,i}^\alpha)\delta(y - y_{k_2,i}^\alpha) . \tag{3.4-1b}$$

In the above, $\{x_{k_1,i}^\alpha, y_{k_2,i}^\alpha\}$ are the locations of the delta sources associated with the *i*th basis or testing function, and *B* and *T* are the original basis and testing functions. Even though the indices of summation in (3.4-1) run over the entire PFFT grid, only M^2 terms are non-zero for each basis/test function, which depicts the way the delta-source representation (3.4-1) is performed for a basis function in the case of $M = 3$, a commonly used value. Lower values involve fewer calculations at the risk of increased error. Given the order *M* of the expansion box, M^2 nodes of the PFFT grid are allocated for the assignment of the delta sources. The choice of the expansion coefficients, represented in compact form through the arrays **B** and **T**, is not unique. Various criteria may be used to specify their values. Investigation shows that even though the multipole reproduction criteria of [3] traditionally used in PFFT for open structures provides sufficient accuracy, more specific criteria based on the least squares approximation of the fields of the waveguide eigen-

modes can provide better accuracy. One may consider the expression for the β component of field produced by jth expansion function $b_j^{\beta}(\vec{r})$,

$$E_j^{\alpha\beta}(\vec{r}) = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} f_{mn}^{\alpha\beta}\varphi_{mn}^{\alpha}(\vec{r})\int_{S_j} b_j^{\beta}(\vec{r}')\varphi_{mn}^{\beta}(\vec{r}')\,ds', \tag{3.4-2}$$

where α and β represent x or y directions for observation and source basis functions, f represents the spectrum of the waveguide Green's function, and $\varphi_{mn}^{\alpha}(\vec{r})$ and $\varphi_{mn}^{\beta}(\vec{r}')$ are the part of the mn^{th} waveguide mode depending on observation and source coordinates, respectively. It appears that should $b_j^{\beta}(\vec{r})$ be replaced with an approximation, $\hat{b}_j^{\beta}(\vec{r})$, the error in the scattered field produced by the jth expansion function is given by the functional, Θ,

$$\Theta_{m,n,j} = \int_{S_j} (b_j^{\beta}(\vec{r}') - \hat{b}_j^{\beta}(\vec{r}'))\varphi_{mn}^{\beta}(\vec{r}')\,ds' \tag{3.4-3}$$

for $m = 0,\ldots,\infty$ and $n = 0,\ldots,\infty$. Hence, the problem of projecting the jth expansion function on the PFFT grid can be formulated as the problem of minimization of the functional (3.4-3) in the least squares sense [79] as follows:

$$\text{minimize over } B_{k_1,k_2,j}^{\beta} : \sum_{m=0}^{\mu}\sum_{n=0}^{\nu} \Theta_{m,n,j}^2 , \tag{3.4-4}$$

where μ and ν are truncation indices that must be chosen such that all eigen-modes that are excited appreciably by the specific expansion function are taken into account. One can

show [79] that minimization of (3.4-4) is equivalent to the solution of the following redundant set of $\mu \cdot \nu$ linear algebraic equations with M^2 unknowns $B^{\beta}_{m_1,m_2,j}$

$$\sum_{m_1=1}^{M}\sum_{m_2=1}^{M}\varphi^{\beta}_{mn}(x^{\beta}_{m_1,j},y^{\beta}_{m_2,j})B^{\beta}_{m_1,m_2,j}=\int_{S_j}\varphi^{\beta}_{mn}(\bar{r})b^{\beta}_{j}(\bar{r})\,ds\ , \tag{3.4-5}$$

where $m=0,\ldots,\mu,\quad n=0,\ldots,\nu$. The indices $m_{1,2}$ are used in (3.4-5) instead of $k_{1,2}$ to single out the M^2 non-zero elements $B^{\beta}_{m_1,m_2,j}$ from the elements $B^{\beta}_{k_1,k_2,j}$ of the entire array, where $k_{1,2}=0,\ldots,2(K_{1,2}-1)$. The procedure for defining the coefficients of the projection of a testing function on the PFFT grid is the same as the one for the expansion functions.

Once the coefficients of the arrays **B** and **T** have been determined, substitution of (3.4-1) into the integrals in (1.2-4) yields the following expressions for the matrix-vector product $\hat{\mathbf{Z}}\cdot\mathbf{I}$,

$$\begin{aligned}\hat{\mathbf{Z}}\cdot\mathbf{I}_{\alpha\beta}&=\sum_{j=1}^{N^{\beta}}\hat{Z}^{\alpha\beta}_{ij}I^{\beta}_{j}=\sum_{k_1=0}^{2(K_1-1)}\sum_{k_2=0}^{2(K_2-1)}T^{\alpha}_{k_1,k_2,i}\times\\&\sum_{k_1'=0}^{2(K_1-1)}\sum_{k_2'=0}^{2(K_2-1)}G_{\alpha\beta}(k_1\Delta x,k_2\Delta y;k_1'\Delta x,k_2'\Delta y)\sum_{j=1}^{N^{\beta}}I^{\beta}_{j}B^{\beta}_{k_1',k_2',j}\end{aligned} \tag{3.4-6}$$

where $i=1,2,\ldots,N^{\alpha}$, and α,β is x or y for 2D. Substitution of (2.3-3) into (3.4-6) with well-known MoM procedure yields the convolution-correlation representation for the matrix-vector product $\hat{\mathbf{Z}}\cdot\mathbf{I}$,

$$\hat{\mathbf{Z}}\cdot\mathbf{I}_{\alpha\beta} = \sum_{k_1=0}^{2(K_1-1)}\sum_{k_2=0}^{2(K_2-1)} T^{\alpha}_{k_1,k_2,i}\times$$

$$\begin{aligned}
&\Bigg\{ s_1^{\beta}\sum_{k_1'=0}^{2(K_1-1)}\sum_{k_2'=0}^{2(K_2-1)}\Gamma^{\alpha\beta}[(k_1-k_1')\Delta x,(k_2-k_2')\Delta y]\Lambda^{\beta}_{k_1',k_2'} \\
&+s_2^{\beta}\sum_{k_1'=0}^{2(K_1-1)}\sum_{k_2'=0}^{2(K_2-1)}\Gamma^{\alpha\beta}[(k_1-k_1')\Delta x,(k_2+k_2')\Delta y]\Lambda^{\beta}_{k_1',k_2'} \\
&+s_3^{\beta}\sum_{k_1'=0}^{2(K_1-1)}\sum_{k_2'=0}^{2(K_2-1)}\Gamma^{\alpha\beta}[(k_1+k_1')\Delta x,(k_2-k_2')\Delta y]\Lambda^{\beta}_{k_1',k_2'} \\
&+s_4^{\beta}\sum_{k_1'=0}^{2(K_1-1)}\sum_{k_2'=0}^{2(K_2-1)}\Gamma^{\alpha\beta}[(k_1+k_1')\Delta x,(k_2+k_2')\Delta y]\Lambda^{\beta}_{k_1',k_2'}\Bigg\}
\end{aligned} \tag{3.4-7}$$

where the matrix $\Lambda^{\beta}_{k_1',k_2'}$ is defined in terms of the product

$$\Lambda^{\beta}_{k_1',k_2'} = \sum_{j=1}^{N^{\beta}} B^{\beta}_{k_1',k_2',j} I^{\beta}_{j} \tag{3.4-8}$$

In matrix form,

$$\begin{aligned}
\hat{\mathbf{Z}}\cdot\mathbf{I} &= \begin{bmatrix} \mathbf{T}^x\cdot\mathbf{G}^{xx}\cdot(\mathbf{B}^x)^T\cdot\mathbf{I}^x + \mathbf{T}^x\cdot\mathbf{G}^{xy}\cdot(\mathbf{B}^y)^T\cdot\mathbf{I}^y \\ \mathbf{T}^y\cdot\mathbf{G}^{yx}\cdot(\mathbf{B}^x)^T\cdot\mathbf{I}^x + \mathbf{T}^y\cdot\mathbf{G}^{yy}\cdot(\mathbf{B}^y)^T\cdot\mathbf{I}^y \end{bmatrix} = \\
&= \begin{bmatrix} \mathbf{T}^x\cdot\left(\sum_{n=1}^{4} s_n^x\Gamma_n^{xx}\cdot(\Lambda^x)^T + \sum_{n=1}^{4} s_n^y\Gamma_n^{xy}\cdot(\Lambda^y)^T\right) \\ \mathbf{T}^y\cdot\left(\sum_{n=1}^{4} s_n^x\Gamma_n^{yx}\cdot(\Lambda^x)^T + \sum_{n=1}^{4} s_n^y\Gamma_n^{yy}\cdot(\Lambda^y)^T\right) \end{bmatrix}
\end{aligned} \tag{3.4-9}$$

where $\mathbf{\Gamma}_n^{\alpha\beta}$, $n=1,\ldots,4$, denote the Toeplitz and Hankel matrices formed by the four convolution-correlation terms entering in (3.4-7), and the superscript T stands for matrix

transpose. Each partial matrix-vector product $\Gamma_n^{\alpha\beta} \cdot (\Lambda^{\beta})^T$ can be computed using the FFT algorithm [79], thus providing the desired $O(N \log N)$ complexity for the MSO

$$\Gamma_n^{\alpha\beta} \cdot (\Lambda^{\beta})^T = FFT_n^{-1}\left\{FFT_n\left\{\Gamma_n^{\alpha\beta}\right\} \cdot FFT_n\left\{(\Lambda^{\beta})^T\right\}\right\},$$

where $FFT_n\{\mathrm{K}\}$ and $FFT_n^{-1}\{\mathrm{K}\}$ are operations of forward and inverse DFT, respectively. The index *n* has been assigned to the FFT operations because in the case when the matrix-vector product $\Gamma_n^{\alpha\beta} \cdot (\Lambda^{\beta})^T$ contains correlation dependence over a certain index, in addition to the forward FFT operation one has to perform a certain rearrangement of the elements in the spectral domain with respect to that index [79].

Since the FFTs of the DGF matrices $FFT_n\left\{\mathbf{\Gamma}_n^{\alpha\beta}\right\}$ are computed once and then stored, it is straightforward to show that, for an iterative matrix solution with number of iterations N_{iter}, the number of required operations scales as $N_{iter} N \log N$, where $N \log N$ is the time per FFT operation.

At every iteration, the above process calculates the product $\hat{\mathbf{Z}} \cdot \mathbf{I}$ (see (3.1-1)). While the calculation of the *far-zone* interactions using the PFFT process are reasonably accurate, the calculated *near-zone* interactions are not. Thus, the operation $\hat{\mathbf{Z}} \cdot \mathbf{I} - \mathbf{Z}_{near} \cdot \mathbf{I}$ is required to correct the calculation of the *near-zone* interactions by replacing the PFFT calculated ones with those obtained using the exact MoM representation of the expansion function interactions (see (2.2-2)). These operations are of $O(N)$ complexity. In addition to reduction

in the computational complexity of iterative solutions, PFFT implementation relaxes memory requirements.

Since only the *near-zone* matrix elements are stored, the $O(N^2)$ overhead associated with the storage of the MoM matrix is avoided. Instead, storing of the "near-zone" interactions results in memory requirements that scale as $O(N)$ with aggressive near part sizing. However, one can increase the size of the near part to provide better convergence behavior, essentially trading memory capacity for CPU time, and potentially, accuracy. Furthermore, due to the Toeplitz/Hankel-like characteristics [79] of the Green's function matrices $\mathbf{\Gamma}^{\alpha\beta}$ on the PFFT grid, those memory requirements scale similarly as $O(N_b M_b)$.

For the benefits of PFFT to be meaningful, the *far-zone* interactions need to be calculated with sufficient accuracy. This, in turn, is dependent on the accuracy of the equivalence from the projection defined in (3.4-1).

In summary, Chapter 3 covers the theory and implementation of the FFT-enhanced MSO for UFFT. It explains the differences between the two variants of UFFT, UFFT-GT and UFFT-P. Further discussion involves improvements that are made in the transition between the two solvers, with UFFT-P being able to handle off-grid circuit discretization and UFFT-GT being faster, more efficient, and extremely accurate. Given the high-accuracy nature of UFFT-GT, an error budget is established, explained, and verified experimentally. Moreover, an overview of the general iterative algorithm, GMRES, is given, including the effects of

preconditioning and matrix banding. Finally, the specifics of MSO as applied to UFFT-GT and then UFFT-P are detailed, including the geometry projection in UFFT-P and the extension of the technique to multilayer geometries as well as 3D geometries in UFFT-GT.

4. Results and Discussions

To demonstrate the performance of the proposed UFFT scheme for shielded structures, many examples are considered. For parity, all examples are analyzed on a microstrip substrate with $\varepsilon_r = 4$.

4.1 Clock Network Example

An example is developed that is easily scalable with regards to unknowns for demonstration of scaling effects on a common geometry. A digital clock network is chosen as it is a fractal geometry for which scaling of *N* is readily observed by extending the depth of the fractal as in Fig. 4.1-1. The example is tested with many different geometries, each with successively more fractal elements, scaling from *N* = 4 190 to *N* = 50 658 (Sonnet), to *N* > 125 000 (UFFT-P), and to *N* > 500 000 (UFFT-GT). All examples are analyzed on a 1 cm ^{2} (5/32 in ^{2}) square substrate at 100 GHz. Note that the number of unknowns in conventional Sonnet is limited by the 32-GB memory capacity of the test system. Timing and memory usage are benchmarked against conventional Sonnet for each example possible. Additionally, accuracy is compared in the form of Y-parameters and also calculated current distributions.

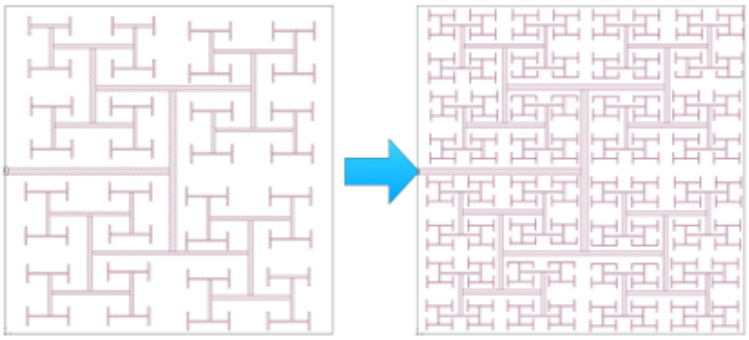

Fig. 4.1-1. The fractal nature of clock networks allows for scaling the number of unknowns in simulation of an applicable example.

All results for this example are computed on the inexpensive Intel Core i5-3570k 3.4-GHz Quad-Core CPU, and can be seen in Fig. 4.1-2.

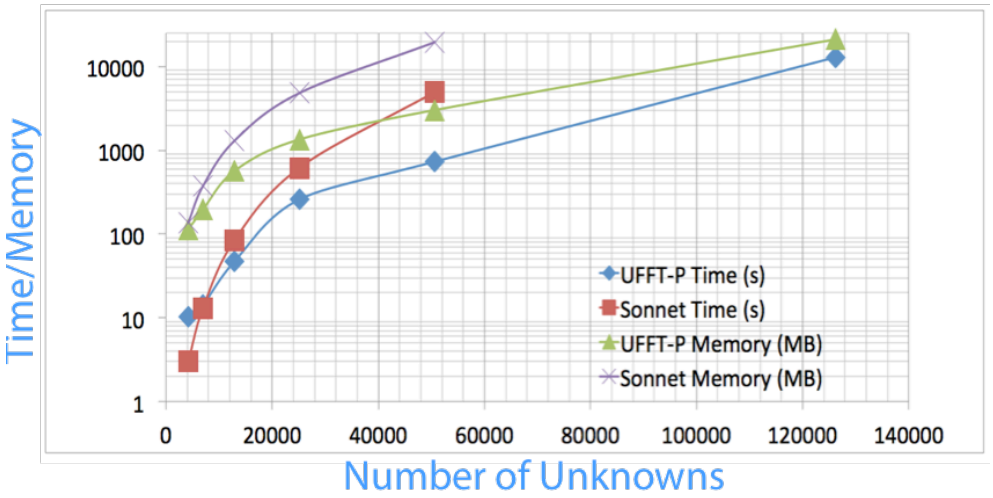

Fig. 4.1-2. Performance comparison between UFFT-P and Sonnet for complete solution of the problem.

UFFT demonstrates substantial performance benefits over conventional Sonnet both in terms of memory utilization and CPU time. Moreover, while Sonnet is able to take advantage of all CPU cores, multi-threading is left to future work for UFFT, meaning optimized code will see further performance increases. In a similar fashion to the first example, the overhead of the PFFT algorithm results in circuits featuring a small number of

unknowns being computed faster in traditional Sonnet. This takeover occurs for UFFT-P when unknowns surpass around 7 000.

Accuracy in this example is compared to conventional Sonnet by way of Y-parameters and current distributions. Generally, switching to UFFT-P from conventional Sonnet yields an additional 1 to 3% error component for dominant Y-parameters in multiport circuits.

UFFT-GT is an improvement to UFFT-P, as discussed in Section 3.1, which yields greater accuracy and efficiency. In this example the efficiency is demonstrated by comparing the scaling data from UFFT-GT against Sonnet and UFFT-P. It is expected to see similar scaling to UFFT-P albeit slightly faster. What was somewhat unexpected is that the curvature of the scaling is also different between the two, as is shown in Fig. 4.1-3. The relatively smooth lines—exhibiting linear behavior when plotted on a logarithmic scale—presented by UFFT-P are not demonstrated with UFFT-GT, and while performance is generally better, at low unknown counts they interchange. In hindsight, this is relatively easy to explain. As UFFT-GT is coded in MATLAB and UFFT-P in FORTRAN, with circuit geometry that is limited by the FFT time (e.g., a large box size with a relatively small number of unknowns), the interpreted code is intrinsically slower. For code with a large number of unknowns where the FFT time is not a limiting factor (and instead, projecting and interpolating the basis functions becomes a limiting factor), the algorithm efficiency significantly outweighs the penalty for interpreted code.

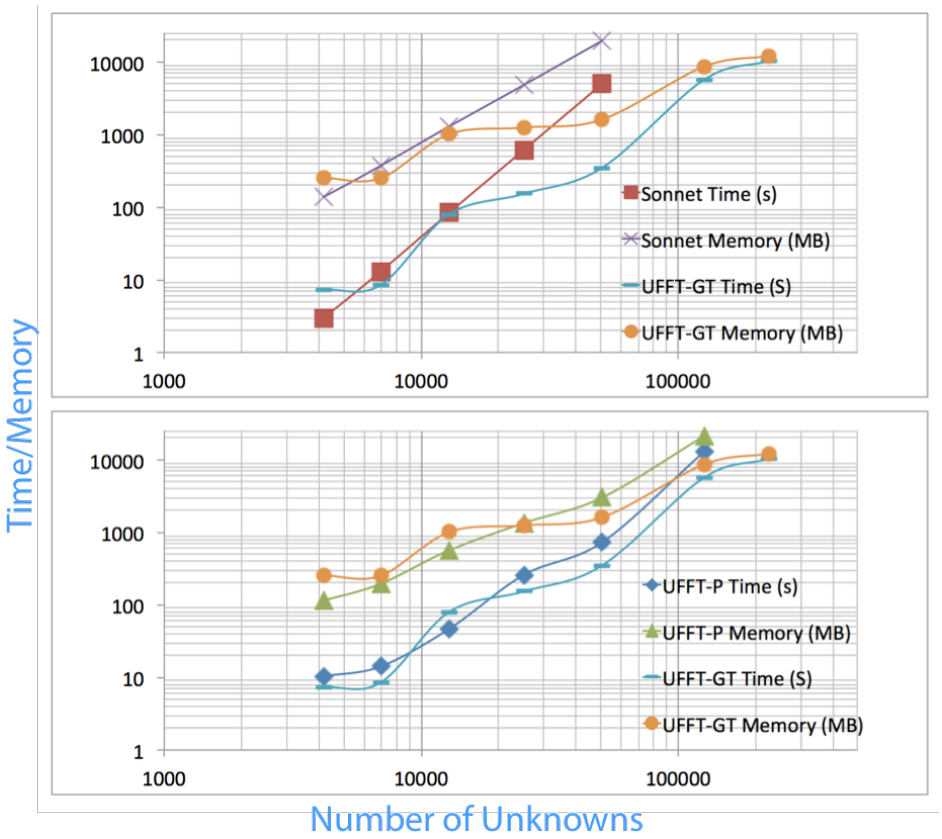

Fig. 4.1-3. Scaling between UFFT-GT and Sonnet (top) and UFFT-GT and UFFT-P (bottom).

Notably, the largest clock network example features more than 250 000 unknowns and was able to run on an Intel i5-3570k quad core CPU in under 3 hours.

4.2 Interdigital Capacitor Example

The capacitance of an interdigitated filter is extracted so as to benchmark the accuracy of the UFFT solver, as well as to compare performance vs. a state of the art MLFMA algorithm [80]. The example circuit is a microstrip configuration containing three terminals for two capacitors for device loading, following general principles established in [81], and is shown in Fig. 4.2-1.

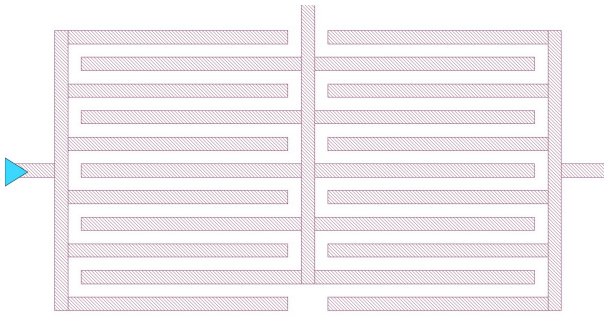

Fig. 4.2-1. The interdigital capacitor example circuit geometry. Port excitation is 1.0 V at the blue triangle.

The structure is on a 1 cm ^{2} (5/32 in ^{2}) square microstrip substrate of thickness 0.1 mm (0.004 in) and is simulated at 100 – 140 GHz. All fingers are 3.55 mm (0.140 in) long and 0.2 mm (0.008 in) wide. The structure is centered on the substrate with feed lines to the edges. All other lines and gaps are also 0.2 mm (0.008 in) wide with their length being deterministic based on other defined parameters. S-parameters are seen in Fig. 4.2-2, and times are shown in Fig. 4.2-3. UFFT-P converged in 16 iterations to 10^{-6}, while UFFT-GT converged in 34 iterations to 10^{-13}.

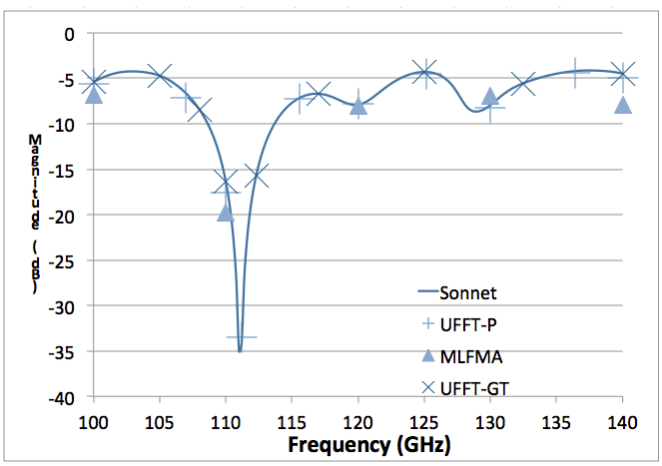

Fig. 4.2-2. S11 vs. Frequency, Sonnet (blue line) vs. UFFT-P (+ marker), vs. MLFMA (triangle marker), vs. UFFT-GT (X marker).

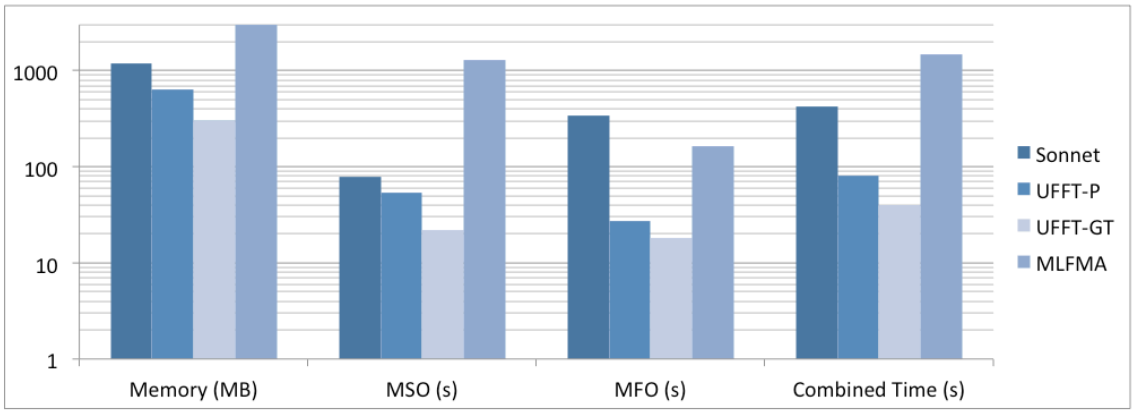

Fig. 4.2-3. Logarithmic timing data for MSO, MFO, and overall time, as well as memory usage for Sonnet, UFFT, and MLFMA simulations of the interdigital capacitor geometry.

Note that while the UFFT-based and Sonnet solvers share partial data from matrix fill operations (MFO), the methodology, code, results, required resources, and even language used for coding, are distinct between the two. MFO times in Fig. 4.2-3 are measured as longer than would be seen in practice due to prototype-mandated diagnostics. As they are uniformly increased as a percentage, times are valid for comparison. Note that the performance of the UFFT algorithms exceed MLFMA in a large part due to the 3D Planar nature of the UFFT implementations vs. the arbitrary 3D implementation of the MLFMA algorithm used [80]. Simply, UFFT and Sonnet have more efficient meshing of the structure. This is because only the conductor needs to be surface-meshed in UFFT while the entire substrate needs to be volume-meshed in MLFMA, as discussed in Section 1.2. This results in N = 12 491 for UFFT vs. N = 97 944 for MLFMA.

TABLE 4.2-1: DETAILED MAGNITUDE AND PHASE ERROR COMPARISON OF UFFT-GT AND MOM

DETAILED MAGNITUDE COMPARISON			DETAILED PHASE COMPARISON		
Freq. (GHz)	S11 Magnitude (dB)		Freq. (GHz)	S11 Phase (Degree)	
	UFFT-GT	Sonnet		UFFT-GT	Sonnet
100	-5.427606066924	-5.427606066925	100	-60.00573797465	-60.00573797460
105	-4.711806439012	-4.711806439013	105	-113.8468205447	-113.8468205447
108	-8.422235655276	-8.422235655271	108	-149.0690098914	-149.0690098914
110	-16.37773802790	-16.37773802790	110	-178.7939190381	-178.7939190381
112.3	-15.69878289660	-15.69878289659	112.3	-23.91528562615	-23.91528562616
117	-6.710118871767	-6.710118871773	117	-81.98287649010	-81.98287649013
125	-4.303694252007	-4.303694252007	125	-107.2371265149	-107.2371265149
132.5	-5.570591718543	-5.570591718542	132.5	-110.8248949553	-110.8248949553
140	-4.489905666691	-4.489905666690	140	-155.1378164179	-155.1378164179

As *S*-parameter data appears virtually identical, a detail of *S*-parameters written out from high-precision Touchstone files is included in Table 4.2-1. The results are identical to 12-13 significant figures, with the notable exception of rounding differences. This level of precision is notable even in slow direct $O(N^3)$ solvers, it is unheard of and has not previously been done in a speed-enhanced $O(N \log N)$ solver. For categories of circuits that require high accuracy design, such as superconducting filters and spiral inductors, UFFT-GT enables a paradigm shift in what can be simulated accurately. Note that the term precision, specifically, has been used as opposed to the term accuracy. MoM itself makes an approximation in terms basis functions and discretization. Accuracy as compared to a 'true' solution is difficult to discern; accuracy as compared to measurement is validated extensively in [10].

4.3 Digital Bus with Delay Line

A digital bus with a meander delay line example is used to demonstrate the accuracy of the UFFT solver for circuits featuring curved metallization. The example geometry, shown in Fig. 4.3-1, consists of N = 26 286 unknowns modeling an 8-bit bus printed in a microstrip configuration on a 10 cm by 10 cm by 1 mm (3.94 in by 3.94 in by 0.039 in) substrate. The bus features lines with width and separation of 1 mm (0.039 in), two of which have delays of separate amounts. The meander geometry is of interest as it demonstrates the capability of the solver to accurately model rounded surfaces despite being discretized solely by rectangular basis functions.

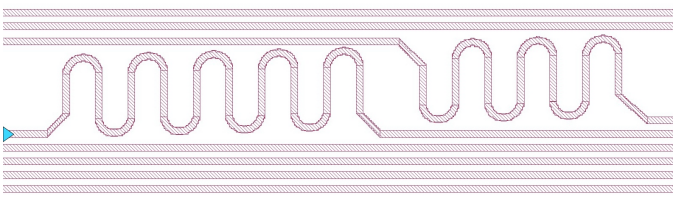

Fig. 4.3-1. The geometry for the digital delay line (meander) example. Port excitation is 1-V at the blue triangle.

Results of current distribution behavior, shown in Fig. 4.3-2 with excitation of the upper delay line, are highly similar. Plots show accurate physical behavior in terms of full-wave patterns, edge singularities, and other physical phenomena. The figure compares Sonnet (top), UFFT-P (middle), and UFFT-GT (bottom). While they are all very similar, Sonnet and UFFT-GT are visually indistinguishable. As such, it is necessary to compare the current distributions numerically. The current data is loaded into MATLAB and the difference

between each element is explicitly calculated. The maximum difference is found to be 7.456 * 10^{-13}, while the mean of all differences is calculated as 2.955 * 10^{-18}.

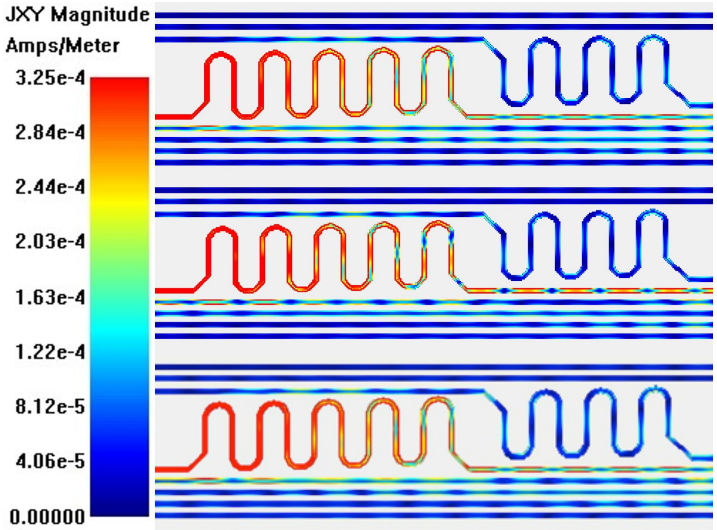

Fig. 4.3-2. Current distributions as calculated by Sonnet (top) and UFFT-P (middle), and UFFT-GT (bottom).

As seen in Fig. 4.3-3, simulation time and memory requirements again heavily favor both UFFT varieties vs. conventional Sonnet, despite the single-threaded implementation for UFFT-P vs. multi-threaded implementation for Sonnet, and the interpreted implementation of UFFT-GT vs. the compiled version of Sonnet. This suggests that when UFFT-GT is ported to a compiled version it will vastly outperform UFFT-P. UFFT-P converged in 7 iterations to 10^{-6}, while UFFT-GT converged in 20 iterations to 10^{-13}. This discrepancy is likely not related to the algorithm; converging to more precision takes additional iterations.

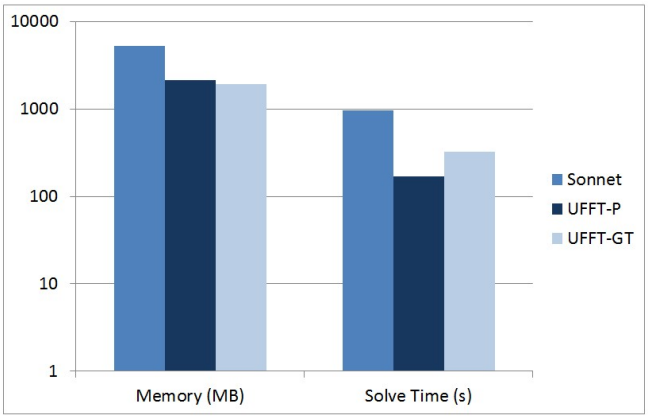

Fig 4.3-3. Memory requirements and MSO time for Sonnet and UFFT for the meander example.

4.4 Computer Motherboard

Between the capacity to handle a large number of unknowns and the meshing efficiency of the shielded planar EFIE, UFFT-GT can handle circuit board geometries of unprecedented complexity. As there are few existing examples capable of pushing the limits of the solver without resorting to unnecessarily dense meshes, a new methodology is developed to reverse-engineer existing circuit geometries.

Starting with the code-base developed in [82], a photograph of a computer motherboard is used to reverse-engineer the geometry of the memory to CPU bus—one of the highest performance parts of the board. The resulting geometry is shown in Fig. 4.4-1. Digital signal processing (DSP) is used in conjunction with SonnetLab to extract and draw polygons from the photograph.

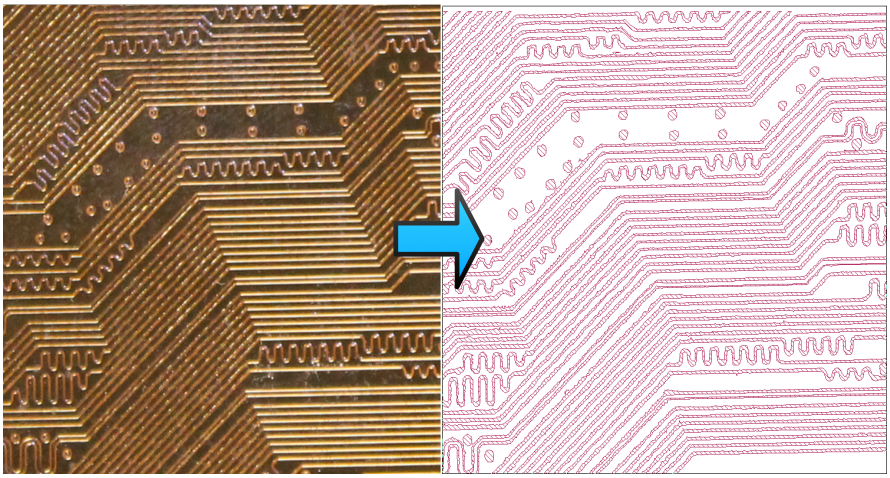

Fig. 4.4-1. The motherboard circuit geometry as reverse-engineered from a photograph.

The DSP process involves eight steps. First, the image is manually retouched in a photo-editing editing program to enhance details and remove artifacts produced by dirt and camera equipment. Then, the image is loaded into MATLAB for the remaining five DSP steps, as shown in the upper left of Fig. 4.4-2. Second, a Gaussian low-pass filter (a blur) is applied, as shown in the upper center. The blur helps to reduce small artifacts from creating polygons. Third, the background level of the image is extracted to produce a gradient, which is shown at the upper right. This gradient is subtracted from the image, shown at the lower left, to account for non-uniform lighting, as well as non-uniform editing during the manual editing process. Fourth, the image is converted to gray scale and the contrast is boosted, shown in the lower center. Finally, the image is "thresholded," whereby values darker than a certain quantity are made black and lighter than the same quantity are made white, making the storage of each pixel simply a "1" or a "0."

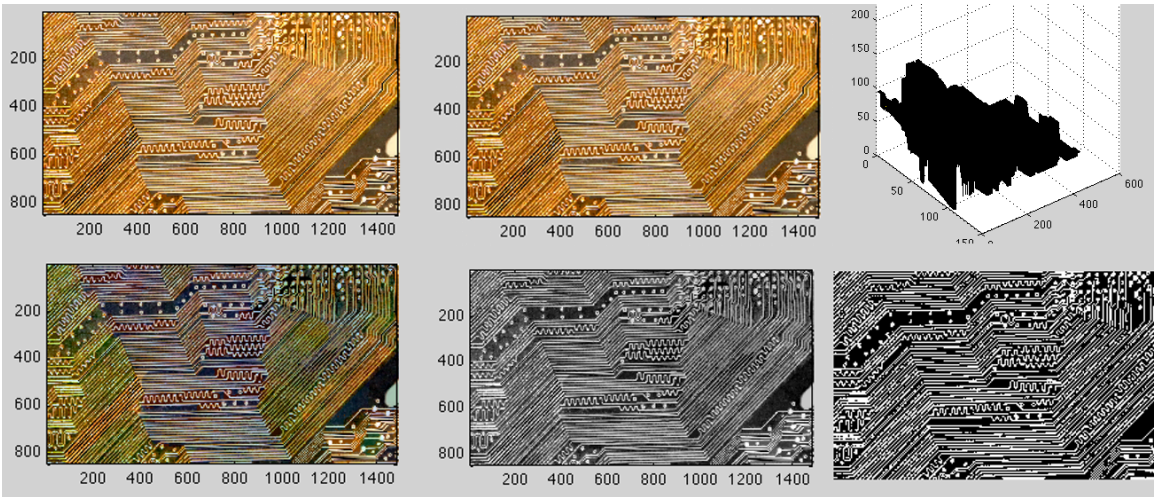

Fig. 4.4-2. The DSP workflow to extract the circuit geometry. The five steps that occur within MATLAB are shown.

The seventh step is to iterate through the "binary" image and produce polygons, shown in Fig. 4.4-3, where each polygon is a drawn in a distinct color to ease the process of finding errors. Finally, the polygons are drawn in a Sonnet project file using the SonnetLAB interface.

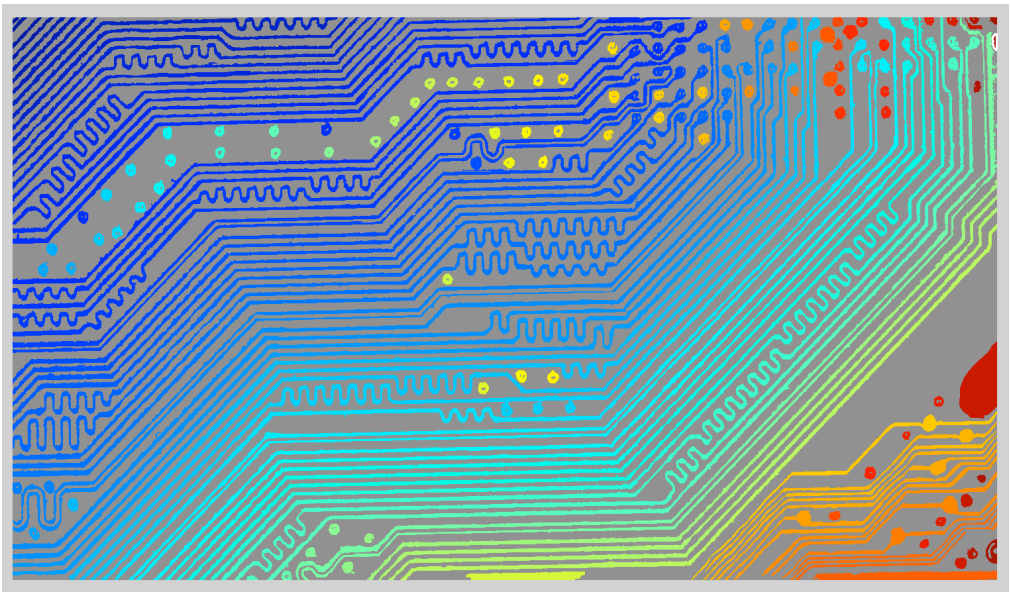

Fig. 4.4-3. The extracted polygons for the image. Each polygon is a different color, aiding the user to identify errors in the extraction process.

The circuit covers approximately 4 cm × 4 cm (1.57” × 1.57”), and is simulated at 100 GHz, which represents the top end of a broadband sweep which would be necessary for transient analysis. It features 300,385 unknowns, and MSO is completed in an average time of 11.4 minutes per port, while consuming 6.8 GB of memory without preconditioning. It converges to 10^{-13} in 40 iterations. The calculated current distribution is shown in Fig. 4.4-4. At left is with every other line that meets the left boundary excited with 1V, as is indicated by the blue arrows. At right is a single line excited with 1V near the center of the figure, as is indicated by the blue arrow.

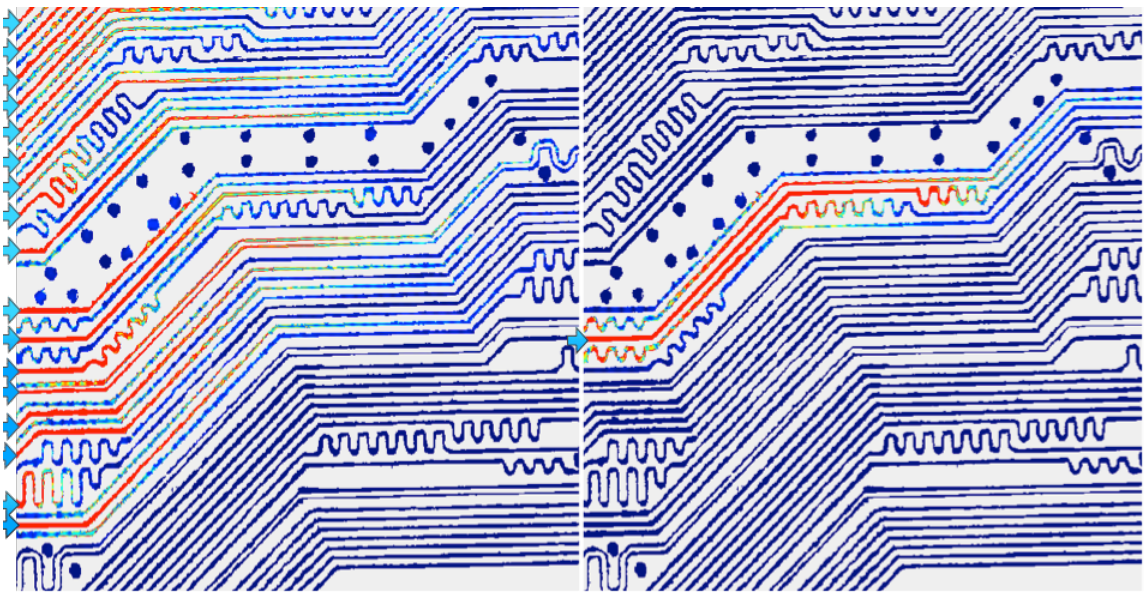

Fig. 4.4-4. Current distribution of the motherboard memory bus example. Left: several lines along the left border excited with a 1 V source. Right: A single line near the center of the figure excited with a 1 V source.

Notably absent from this discussion is a comparison with other solvers. While there are other solvers available which can simulate this number of unknowns, none of these high-level of complexity solvers also feature the ability to efficiently mesh the structure as a planar circuit in a shielded environment. The capability of UFFT-GT to solve this circuit on

a conventional desktop computer is unique, and thus there is no other currently available solver to compare the results to with the available computing resources.

4.5 Complex-Bus Example of Multiple Planes

In order to demonstrate the capability of UFFT-GT to handle structures featuring multiple planes, the example from Section 4.3 is extended to a larger bus featuring additional lines, two additional metallization layers, and reversal network—a type of structure not geometrically possible on a single plane. The example is shown in Fig. 4.5-1.

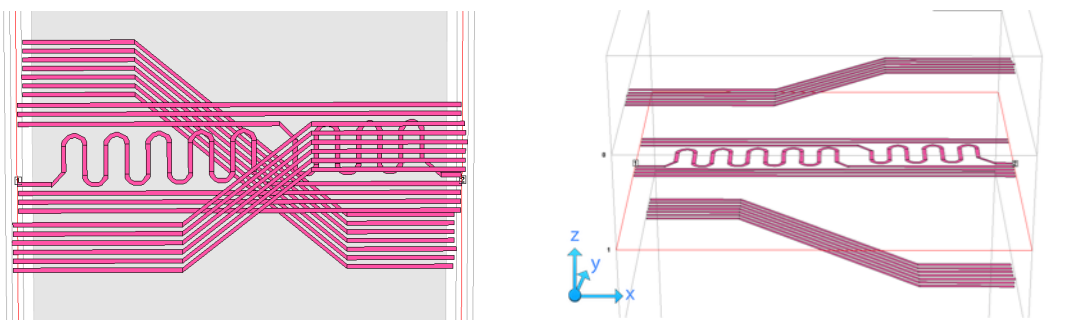

Fig. 4.5-1. The complex-bus example structure geometry, complete with meanders and a reversal network.

The circuit features 57,866 unknowns. It is simulated in 25.2 minutes and occupies 5.49 GB of RAM, vs. 2 hours 24 minutes and 25.5 GB for simulation in conventional Sonnet, as shown in Fig. 4.5-2. Convergence was achieved in 34 iterations for each port to better than 10^{-13}. This, compared with convergence of 20 iterations in the previous meander example (which featured considerably less unknowns), suggests that having multiple planes does not significantly impact the condition of the matrix.

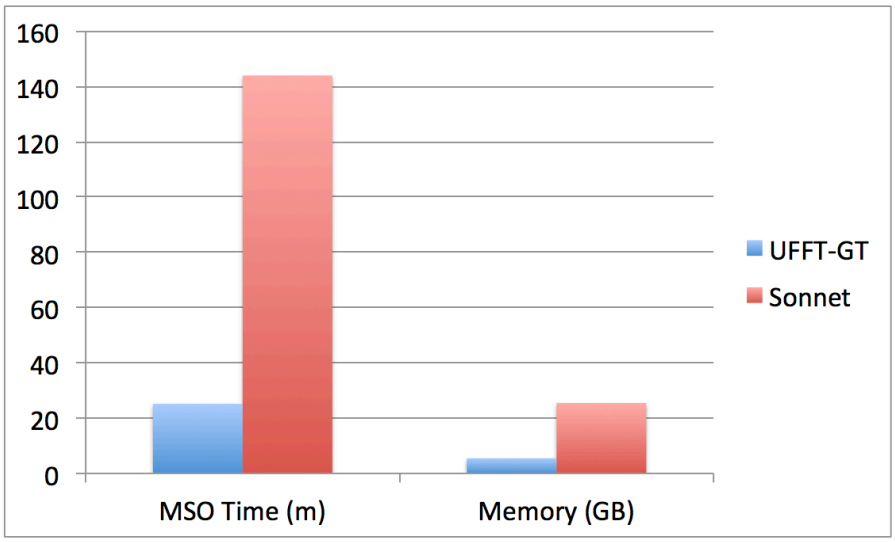

Fig. 4.5-2. Performance between Sonnet and UFFT-GT for the multiplane bus example.

Fig. 4.5-3 shows current distribution data for the multiplane bus circuit, with a 3D view at the upper left and individual planes in all other quadrants.

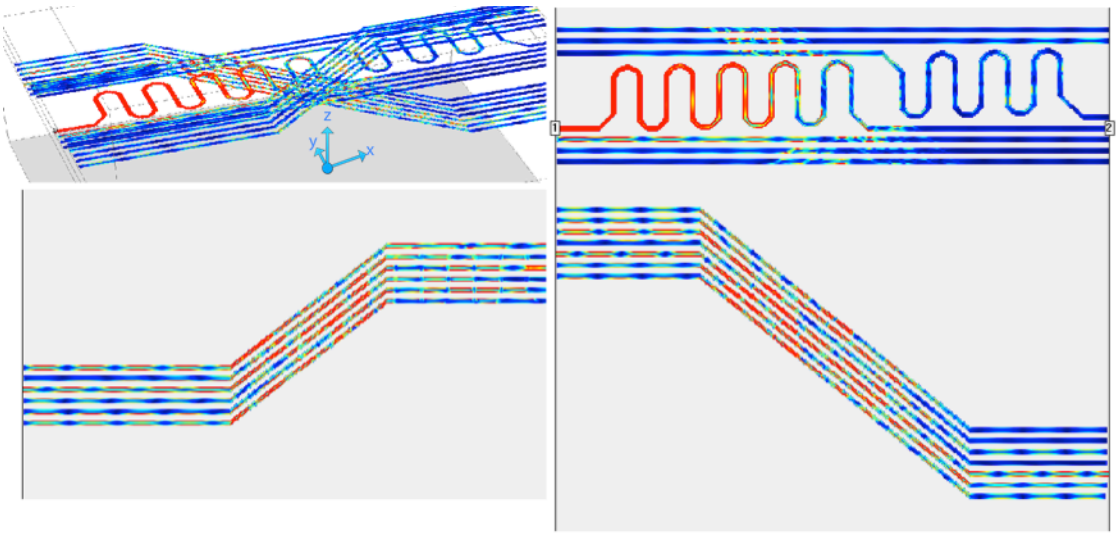

Fig. 4.5-3. Current data for the multiplane bus circuit. 3D at upper left quadrant, individual layers in others.

For comparison purposes, a similar plot of current is shown generated from conventional Sonnet data in Fig. 4.5-4. The two are visually identical.

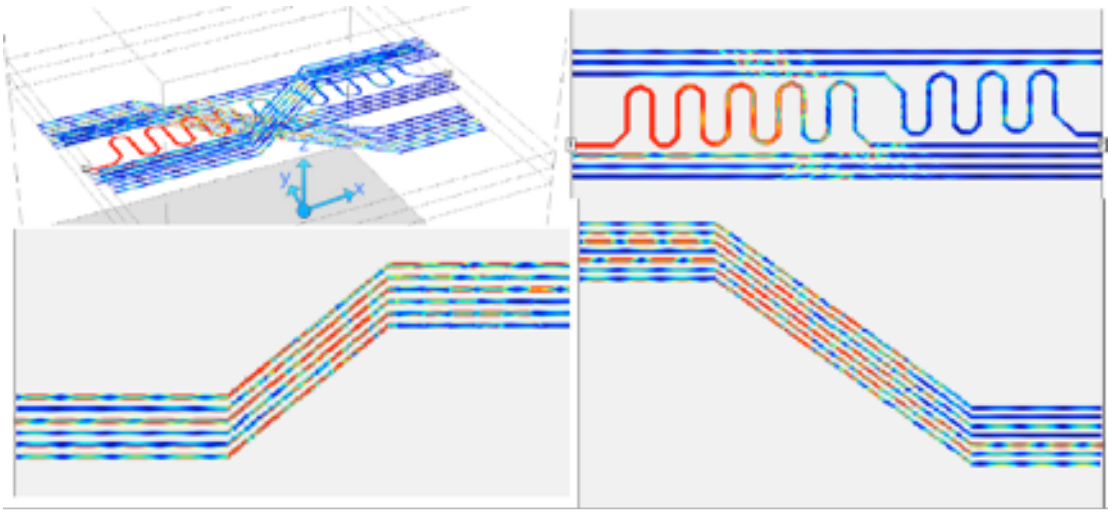

Fig. 4.5-4. Current data for the multiplane circuit simulated in conventional Sonnet.

4.6 3D Planar Multilayered Bus Example

For demonstration of the 3D planar capability of the solver, the meander example from Sections 4.3 and 4.5 is extended to include vias. All lines of the bus start and end on the same plane as they would likely lie in actual designs, however, they must cut through different levels of the PCB so as to allow crossing over. The example is shown in Fig. 4.6-1. The upper left quadrant shows an overhead view, the upper right shows an isometric view, the lower left shows another isometric view, and the bottom right shows an isometric view with the thru-lines removed so that the crossover network details can be more easily seen.

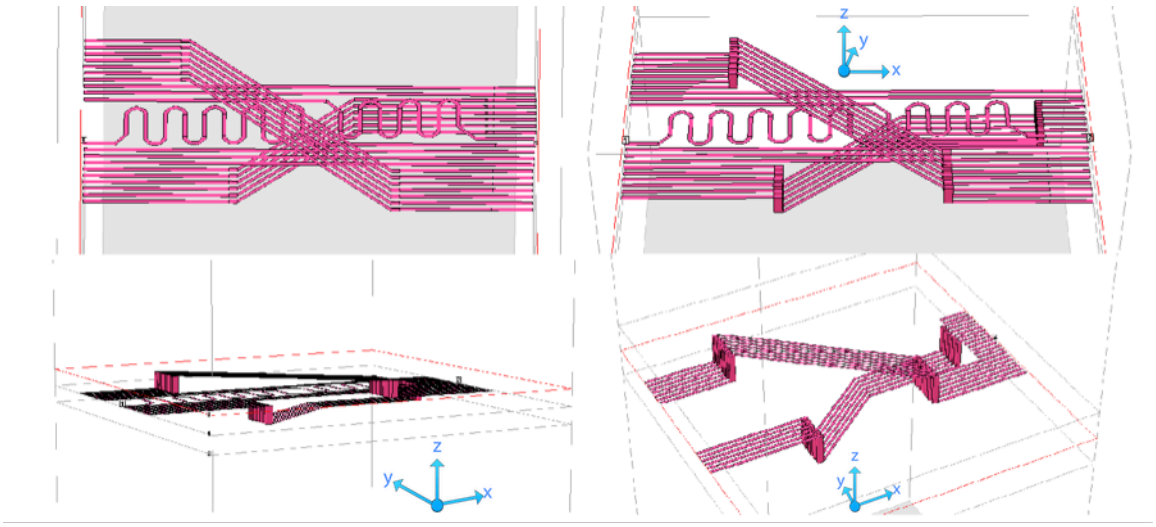

Fig. 4.6-1. The 3D planar example. Top left: overhead view, top right: isometric view, bottom left: additional isometric view. Bottom right: isometric view with thru-lines removed for clarity.

The example consists of a total of 65 159 subsections. In UFFT-GT, it takes 2 hours 4 minutes for MSO, although most of this time is spent with an overly dense and largely ineffective preconditioner, as well as the sheer number of iterations required (101 iterations, whereas 30-40 seen in previous examples is a more common and reasonable number). This means that vias significantly degrade the condition number of the matrix. Individual matrix-vector products take 35 seconds, meaning that with an effective and efficient preconditioner, it should be possible to simulate the circuit in approximately 20 minutes. Simulation consumes 8 GB of memory before solving the preconditioner, and a maximum of less than 20 GB when solving and applying the preconditioner, although this number can be drastically reduced through the use of a more efficient preconditioner. Comparatively, in conventional Sonnet, it takes 2 hours and 27 minutes and consumes 32.4 GB of memory to solve, as shown in Fig. 4.6-2.

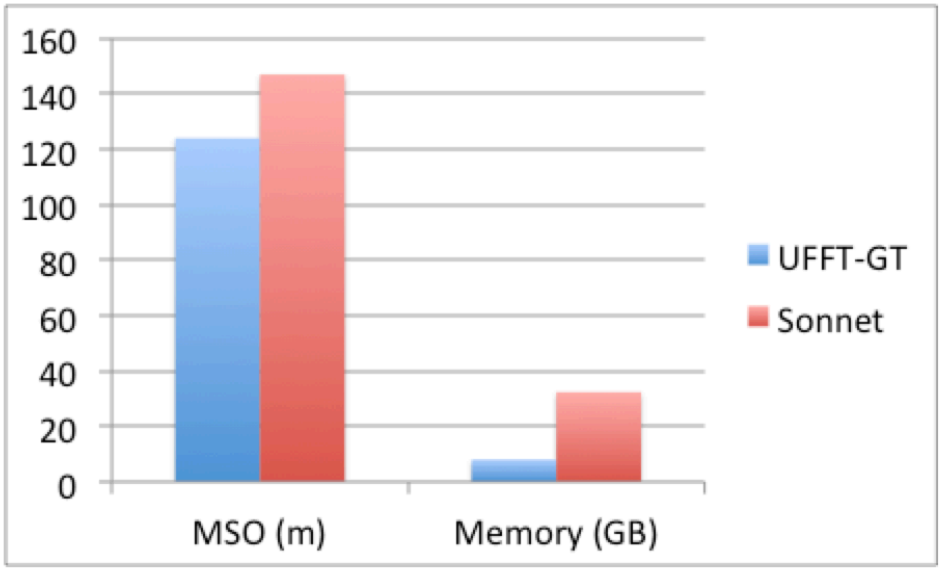

Fig. 4.6-2. Performance comparison for UFFT-GT and Sonnet for the 3D planar multilayer example.

In Fig. 4.6-3, the current distribution of the example is shown for both Sonnet and UFFT-GT. It is very close to visually identical, particularly on the planar subsections. There are slight variances for the current values within the vias at the convergence level that was achieved (approximately 10^{-5}). Better convergence will be achieved with further research into preconditioners as applied to vias, which is observed to have unique requirements for preconditioning as compared to planar subsections.

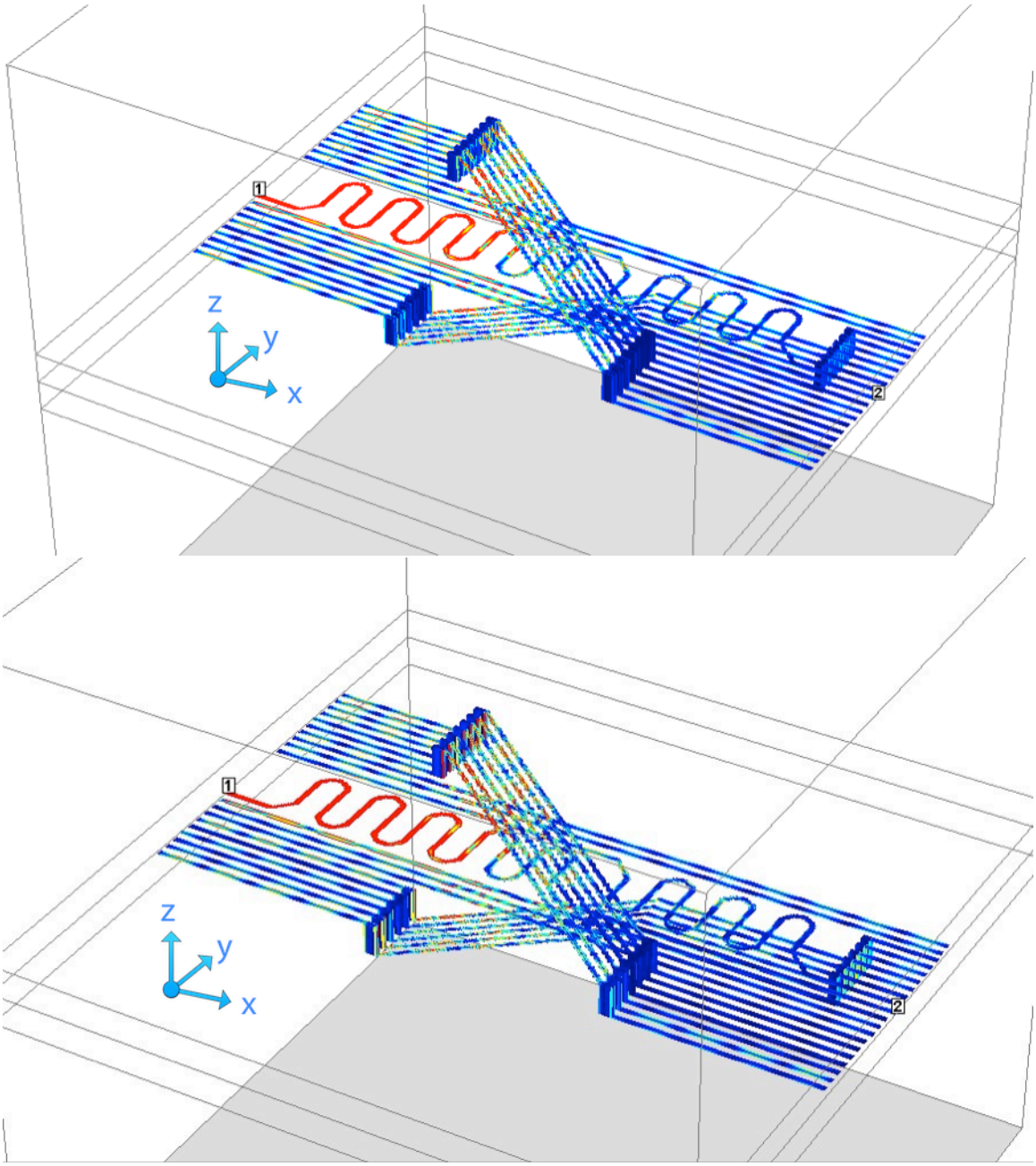

Fig. 4.6-3. The current distribution in Sonnet (upper) and UFFT-GT (lower) for the 3D planar example.

In summary, Chapter 4 has introduced six example circuits featuring a variety of topologies and configurations. These circuits are used to benchmark the accuracy, computational performance, and memory consumption of UFFT-GT and/or UFFT-P, and comparisons with

various other solvers are included. The six circuits start with a digital clock network, from which the fractal nature of the geometry is exploited to demonstrate the scaling of the algorithm. The relatively simple layout of an interdigital capacitor is then used as an easily portable geometry to various simulators. A digital bus with a delay line is used three times, once to demonstrate curved structures, again to demonstrate multilayer geometries, and to demonstrate 3D geometries. Finally, a large area of a computer motherboard is simulated to show the capability of the solver on large and intricate circuits for which full-wave electromagnetics are not generally possible to simulate by conventional means.

5. Possible Extensions to the Algorithm

Although considerable work has been completed and both variants of UFFT have been tested extensively, there are still many future improvements that can be made.

5.1 Loop-Tree Implementation

A high priority extension to this work is the addition of Loop-Tree and or Loop-Star basis functions [56], [83], [84]. Currently, GMRES performance is good for high frequencies, however, the noise and the condition number of the matrix rise as frequency goes down. This can be combatted with loop-tree basis functions. Essentially, the circuit is split into basis functions featuring current *loops* (curling) and current *trees* (curl-free). This allows the splitting of the contributions made from electrostatics and magnetostatics at low frequency. By splitting them, it is possible to improve the convergence of GMRES at low frequency for a more robust solution.

5.2 Calderon Preconditioning

While this work focuses on the algorithmic efficiency of GMRES and the matrix vector products, the overall performance of implementations of the presented algorithm is highly dependent on preconditioning. The near-interaction preconditioner used extensively in this work is highly effective for electrically large structures (i.e., those which have features on the order of a wavelength or more), however for electrically small structures an adequate preconditioner is not implemented. One such preconditioner, the Calderon

Multiplicative Preconditioner (CMP) [75], partially resolves this issue and is a natural extension for this work.

The CMP is, relatively speaking, straight forward to implement in that it still uses a standard GMRES with a standard (multiplicative) way to apply the preconditioner. There are other methods that aim to achieve similar results, which require sophisticated routines to apply the preconditioner [85]. The complexities of the CMP technique lie instead in replacing existing rooftop basis functions for similar but different Buffa-Christiansen [86] basis functions. Rather than treating the *edges* of the rooftops as the basis functions, it is the *nodes* between rooftops that make up the basis functions in this scheme. Current flows both in a divergent fashion from the node as well as in a curling fashion around the node. Fig 5.2-1 from [75] shows a comparison between a standard diagonal preconditioner (EFIE) and a Calderon preconditioner.

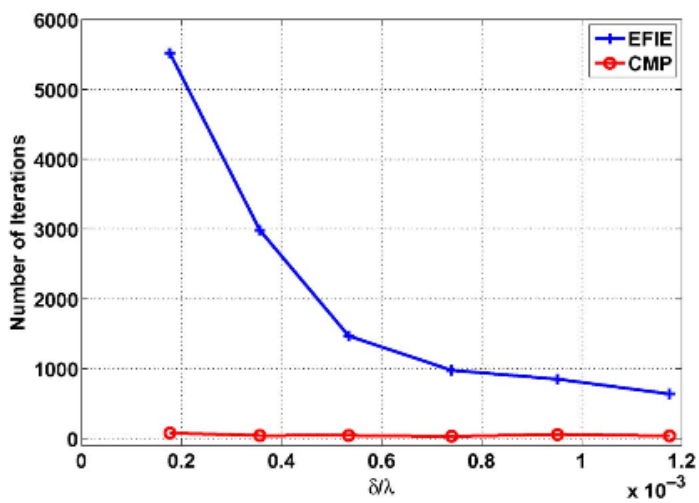

Fig. 5.2-1. Convergence for a Calderon preconditioner vs. a diagonal EFIE-based preconditioner [75].

Note that the blue curve in this figure is remarkably similar to the curve shown in Fig. 3.2-2. The *y*-axis shows iteration count and the *x*-axis shows electrical size of subsections. The low-frequency behavior of the diagonal preconditioner is quite poor, and the high-frequency behavior is merely acceptable. Conversely, CMP is effective at low-frequencies.

5.3 Subsection Combining

In this work, all subsections span only a single edge on the grid. This effectively performs a uniform resolution for current distributions at the expense of an increased unknown count. As current behavior can sometimes be predicted, e.g., higher currents at the edges of transmission lines, it is often a good approximation to increase resolution in some parts of the circuit while having lower resolution at others, as in Fig. 5.3-1. At the left of the figure, no subsections are combined, although the cells have a wide aspect ratio as the uniform grid is divided more densely on the *y*-axis than on the *x*-axis. This means that the number of *elemental subsections* equals the number of unknowns ($N_e = N$). In the center, some elemental subsections are combined into fewer regular subsections to reduce the number of unknowns ($N_e > N$), while subsections along the edge of the metal are not combined. At the right, subsections are aggressively combined in the center, and less aggressively at the edges ($N_e >> N$).

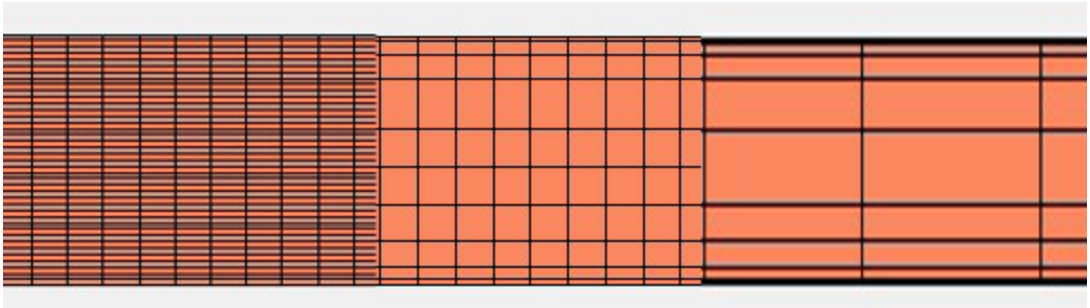

Fig. 5.3-1. Three different subsection-combining schemes for the same transmission line, from dense (left) to coarse (right).

Although combining subsections will not affect the timing or scaling for the matrix vector multiply, which scales with the size of the grid (N_b and M_b) and not the number of unknowns (N), subsections may still be combined with UFFT to reduce N. This will not only reduce the size of the optimization space within which GMRES must find a minimum, but also reduces the number of matrix elements in the preconditioner and thus reduces the preconditioner application time, often a limiting factor. As such, it is a high-priority extension of UFFT.

5.4 Extension to Optimizers

While the faster simulation time provided by this work is a very important contribution to any EDA/CAD workflow, the high quantity of numerical simulations required in optimization workflows, such as those employed in antenna design, makes speed increasingly important. To demonstrate potential efficacy, an example antenna optimization is carried out in conventional software, and a theoretical approximation is made for the speed enhancement of this workflow with UFFT-P.

A 2.4 GHz patch antenna array is designed based on [87] to present an example workflow. The structure, shown in Fig. 5.4-1, features a meshing with N = 2 858 unknowns requiring 87 MB of memory to simulate.

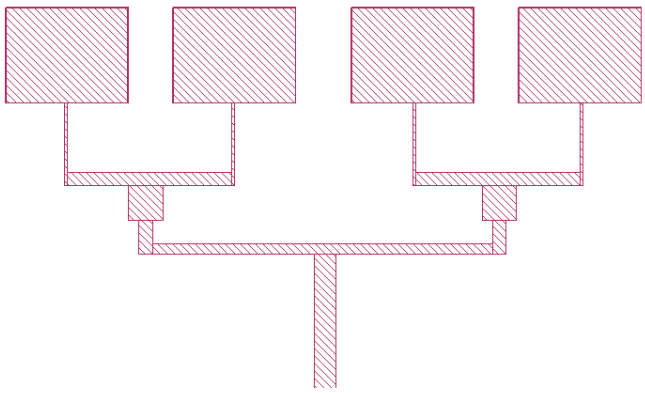

Fig. 5.4-1. The general layout of the microstrip antenna array.

5.4a Optimization of the Antenna Patch With SonnetLab

The first component of the design requiring optimization is the antenna patch itself. As this can generally be narrowed down to two parameters—length and width—it is both possible and convenient to sweep both parameters across a range of possible values. While this process could be rolled into the next section, it is both a convenient way to get an approximate design and a good demonstration of the parameterization process that is used.

Parameters (shown in Fig. 5.4-2) are created in Sonnet's Xgeom user interface to automatically control the length and width of each of the four patch antennas in unison. An initial nominal *guess* consists of a width of 36 mm and length of 35 mm.

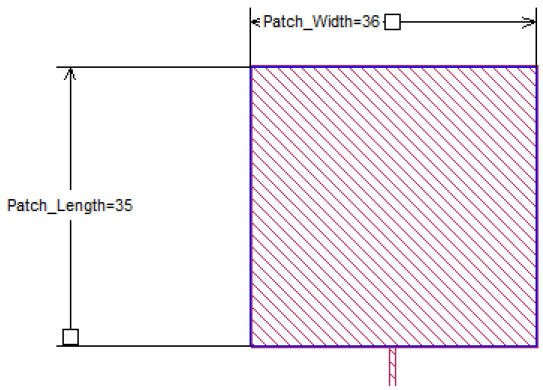

Fig. 5.4-2. Close-up view of a single patch antenna with its parameterization. Units are mm.

The Sonnet simulation is set up to automatically simulate all possible combinations of values between 25 mm and 35 mm for width (with a step of 5 mm) and between 30 mm and 40 mm (with a step of 2 mm) for length. This procedure results in a total of 18 simulations of the structure, each of which calculates approximately 300 frequency points with an adaptive band sweep. In total, all 18 simulations complete on a 2.3 GHz Intel Core i7 Quad Core CPU in around 10 minutes.

Once completed, the VSWR of all combinations of both parameter values can be plotted on the same graph as shown in Fig. 5.4-3.

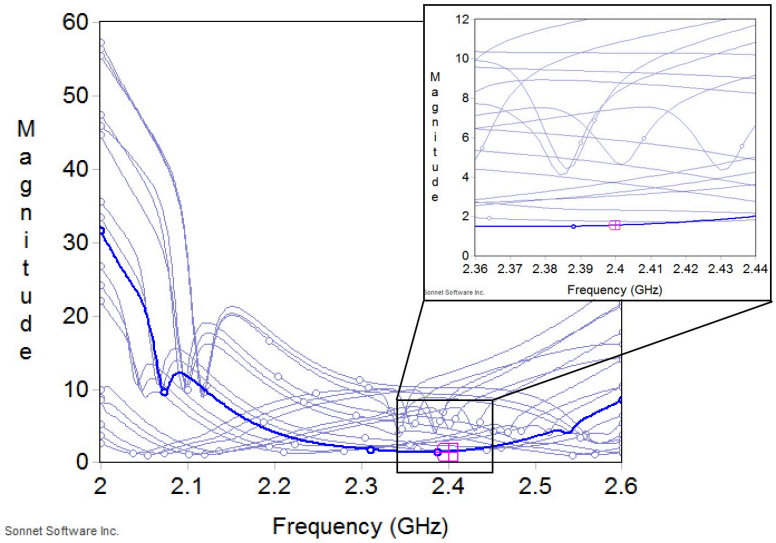

Fig. 5.4-3. S-parameters for the parameter sweep are shown. Each line graphed represents a different combination of length and width values for the patch.

As a preliminary sizing for the patch elements, it is convenient to simply take the values for the curve that features the lowest VSWR at our desired design frequency, 2.4 GHz. These values are a length of 34 mm and a width of 35 mm, and result in a VSWR of 1.61, which, while unacceptable for a final design, represents a good starting point for work in the next section. Note that while VSWR is far from the only parameter to consider in antenna design, it is convenient to choose it as a singular design goal so as not to confuse engineering tradeoffs and design goals with operation of the optimizer in this generalized example.

5.4b Optimization of the Feed Network with SonnetLab

With an approximate size for the patch antennas in place, lower VSWR and thus a better antenna design can yet be achieved with adjustment of additional parameters of the corporate feed network. For this work, a symmetric feed network is used, although the

design could also be implemented with an asymmetric network for additional control of phase and pattern of the array. The layout is split into 11 parameters (including the original 2) as shown in Fig. 5.4-4. While it would be possible to simulate and plot all combinations of these parameters as in Fig. 5.4-3, with just 3 values per parameter this would result in a prohibitively large set of 59 049 simulations. Thus, instead of using a brute force approach, the SonnetLab application-programming interface (API) is leveraged to interface the circuit and its associated parameter variables to a nonlinear optimizer [88] in MATLAB [64]. This optimizer controls each of the variables and attempts to find a minimum for the scalar VSWR result from the simulation.

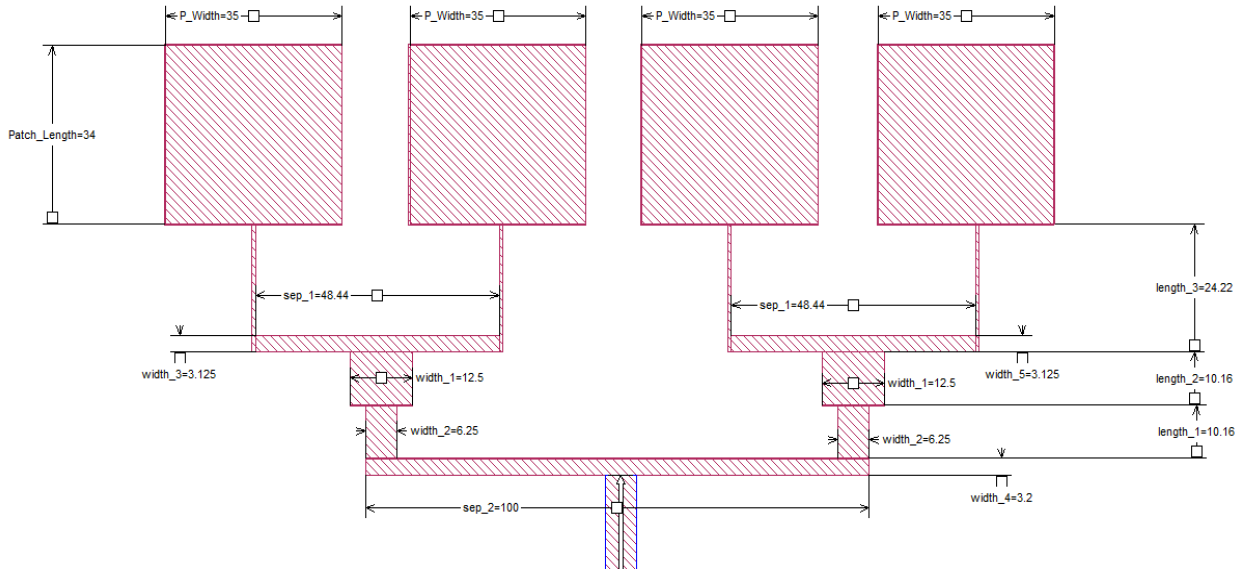

Fig. 5.4-4. The full parameterization of the patch array. Squares denote anchor points in the parameterization.

After 113 iterations which took a total of less than 54 minutes, the SonnetLab/MATLAB optimizer converges to a VSWR of 1.048, with variables as in Table 5.4-1.

TABLE 5.4-1: OPTIMIZED VALUES FOR THE PARAMETERIZED CIRCUIT

Parameter	Optimized Nominal	**Parameter**	Optimized Nominal
Patch Width	37.794	**Length 1**	9.7035
Patch Length	34.786	**Length 2**	10.272
Width 1	11.522	**Length 3**	25.141
Width 2	6.4368	**Separation 1**	64.301
Width 3	3.0244	**Separation 2**	155.81
Width 4	3.0837		

Additionally, as the SonnetLab API is used, it is possible to generate a full 3D plot of the antenna pattern, shown in Fig. 5.4-5. This functionality is not typically available in 3D planar simulators such as Sonnet. While it would also be possible to use this data to optimize phasing of the area to control the pattern, this is beyond the scope of this work.

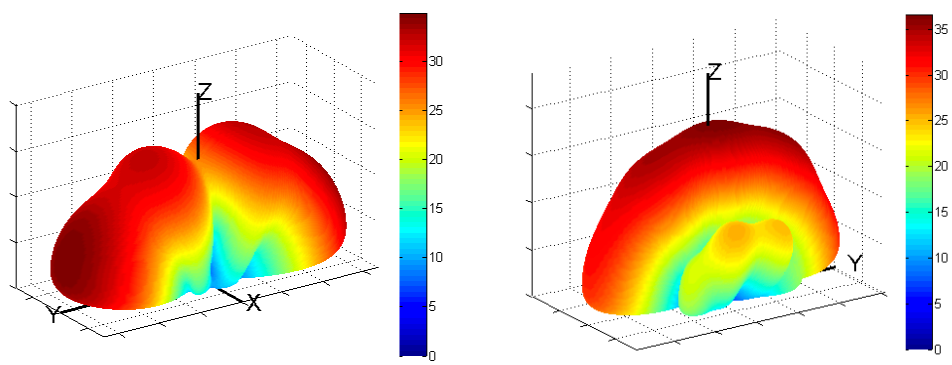

Fig. 5.4-5. The simulated 3D antenna pattern of the array before (left) and after (right) the optimization process.

5.4c Speed Enhancement Potential with the Unified FFT Algorithm

This represents a direct-electromagnetic-optimization based on MoM solution for which CPU time scales with problem size N as $O(N^3)$ [2]. While this is acceptable for simple structures such as the demonstrated example, (N = 2 858), as geometries get more complex, the full electromagnetic simulation required at every iteration becomes increasingly more time and memory prohibitive. There are numerous optimization methods that can be used to circumvent the need for a fully electromagnetic simulation every iteration, e.g. [89], however, recent advancements in full electromagnetic simulation may make this optimization workflow a viable alternative with the potential for considerably easier end-user implementations. In this example, UFFT-P is applied to a very tightly meshed, single-patch antenna example, and a projected optimization time is extracted. This new example features N = 27 915 unknowns, and based on a similar optimization of 133 iterations, would take a projected 38 hours to simulate, vs. less than 24 hours with the UFFT algorithm, as seen in Fig. 5.4-6. Moreover, as UFFT is not linearly faster but rather scales better, as N increases, this gap becomes larger. Thus, a UFFT-based electromagnetic solver used with this optimization workflow should be able to obtain similar VSWR in considerably less time. The positive results of this example demonstrate that future work applying optimizers to both variants of UFFT should be carried out.

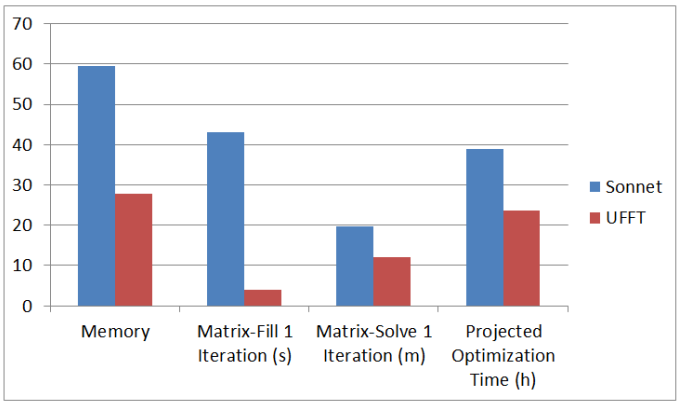

Fig. 5.4-6. Actual single iteration simulation times and projected full optimization time.

In summary, Chapter 5 explains a number of research directions that this topic may take in the future. Most of these are natural progressions to make the algorithm more robust and powerful, such as loop-tree basis functions and Calderon preconditioning, which can ensure that the iterative solver has good convergence for an even wider range of structures. Others include methods to further increase the efficiency of the solver. Indeed, some of these extensions may be considered a pre-requisite for the commercialization of the algorithm to take place. Finally, a detailed examination of the potential benefits of UFFT as applied to optimizer workflows is discussed.

6. Conclusions

This work has developed two new methodologies for speed-enhanced full-wave electromagnetic numerical simulation. It combines two existing forms of speed-enhanced MSO with a speed-enhanced MFO to form the UFFT-GT and UFFT-P solvers. Both demonstrate high accuracy simulation with great speed improvements vs. existing code bases, with UFFT-GT notably demonstrating accuracy to numerical precision and UFFT-P notably allowing for off-grid basis functions.

This work makes three primary contributions. First, FFT-enhancement is unified for both MFO and MSO in a single solver. Previously, existing solvers have employed FFT for either MFO or MSO exclusively. Solvers with only FFT-enhanced MSO commonly require slow, inaccurate, or pre-stored data for the Green's function. Solvers with only FFT-enhanced MFO are generally slow to simulate. By combining FFT-enhanced operations for both MFO and MSO, UFFT makes it possible to have a simulator that is fast, accurate, reliable, and that does not require any Green's function or matrix data to be pre-computed and stored before starting the simulator. Fig. 4.2-3 below is repeated to demonstrate the speed of the UFFT-based solvers developed in this work.

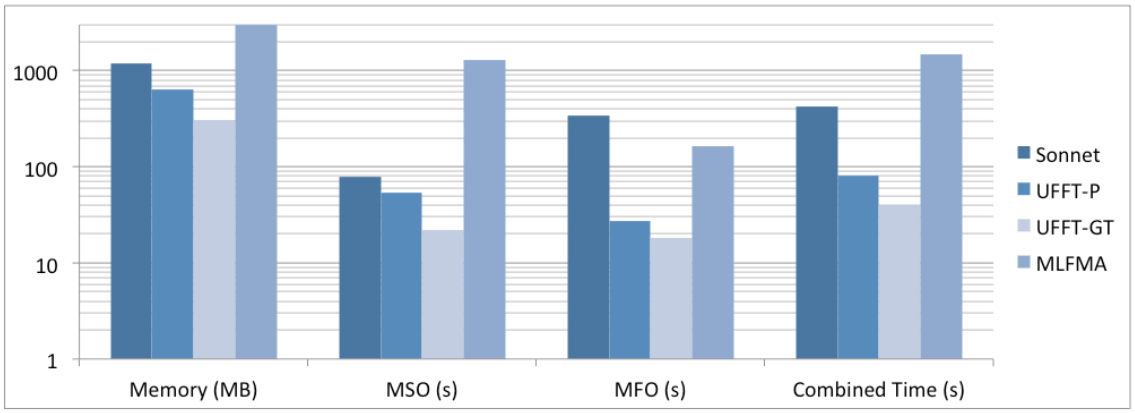

Fig. 4.2-3 (Redux). Logarithmic timing data for MSO, MFO, and overall time, as well as memory usage for Sonnet, UFFT, and MLFMA simulations of the interdigital capacitor geometry.

Second, a fast electromagnetic solver is developed which does not lose precision compared to full, direct MoM. Conventional wisdom suggests that doing something faster often means doing it with lower quality. Many existing speed-enhanced solvers that lose accuracy compared to full, direct MoM reinforce this notion. In contrast, UFFT-GT is able to perform high-speed analysis with no loss in precision compared to MoM to the numerical precision of the computer, commonly 12 significant figures. Table 4.2-1, which compares results from a direct solver to those from UFFT-GT to a full 13 digits for an example circuit, are repeated below to show this level of precision.

TABLE 4.2-1 (REDUX): DETAILED MAGNITUDE AND PHASE ERROR COMPARISON OF UFFT-GT

DETAILED MAGNITUDE COMPARISON			DETAILED PHASE COMPARISON		
Freq. (GHz)	S11 Magnitude (dB)		Freq. (GHz)	S11 Phase (Degree)	
	UFFT-GT	Sonnet		UFFT-GT	Sonnet
100	-5.427606066924	-5.427606066925	100	-60.00573797465	-60.00573797460
105	-4.711806439012	-4.711806439013	105	-113.8468205447	-113.8468205447
108	-8.422235655276	-8.422235655271	108	-149.0690098914	-149.0690098914
110	-16.37773802790	-16.37773802790	110	-178.7939190381	-178.7939190381
112.3	-15.69878289660	-15.69878289659	112.3	-23.91528562615	-23.91528562616
117	-6.710118871767	-6.710118871773	117	-81.98287649010	-81.98287649013
125	-4.303694252007	-4.303694252007	125	-107.2371265149	-107.2371265149
132.5	-5.570591718543	-5.570591718542	132.5	-110.8248949553	-110.8248949553
140	-4.489905666691	-4.489905666690	140	-155.1378164179	-155.1378164179

Third, a fast, shielded 3D planar extension is made to the solver. Conventional planar research solvers typically omit this step as it adds additional data and complexity to the solver. However, rarely are real circuits limited to a single plane. By adding support for multilayer geometries to the UFFT-GT solver, it makes it possible to simulate structures with multiple layered planes of metallization, such as those commonly found on a PCBs and ICs. The additional contribution of *z*-directed basis and testing functions allows for the simulator to handle vias between these planes of metallization. This is all proven to work with the same precision as the most accurate conventional solvers and with similar or better performance as compared to speed-enhanced solvers.

The new UFFT solvers have been demonstrated in many practical examples in both the microwave and high-speed digital engineering domains, including a digital clock network,

an interdigital capacitor, three variants of a digital bus with a meander line, and a partial computer motherboard memory bus. This rigorously defines robustness, performance, and capability. The initial results show extreme potential, surpassing a level that warrants commercial development of the algorithms for potential widespread use within both industry and academic research. As such, significant effort has been put into defining and outlining future directions that research on the topic may take. Research effort is intended to continue well beyond this work.

Appendices

Appendix A: UFFT Terminology

N – The number of unknowns as seen by GMRES (e.g., after subsectioning)

N_e – The number of elemental unknowns

N_b – The number of cells across the box (X-axis), columns.

M_b – The number of cells down the box (Y-axis), rows.

P – The number of planes occupied by metal

i – The number of GMRES iterations

N_d – The number of diagonals in the banded pre-conditioner matrix

$N_{pre-corr}$–The number of "near-interaction" unknowns involved in the PFFT pre-correction.

K – The matrix that speeds MFO through the use of FFT.

Appendix B: Evaluation of DGF via DCIM Fitting and Ewald's Transform

For evaluation of the near interactions between elements non-conformal to the uniform grid, the approach based on subtraction and analytic integration of the 1/r Green's function singularity followed by quadrature based integration [90] of the non-singular part of the DGF is commonly used [78],[56], and [58].

First, the components $G_{\alpha\beta}$ are expressed in terms of the scalar potentials [91] as in

$$G_{xx} = \frac{\partial^2 F_x^{TM}}{\partial x^2} - k^2 \frac{\partial^2 F_x^{TE}}{\partial y^2} \tag{B1}$$

where k is the wave number of the media at the observation point. The scalar potentials $F_\beta^{TM,TE}$ are the contributions to the total field. Their spatial dependence has the form

$$\begin{aligned} &F_\beta^{TE,TM}(x,y;x',y') \\ &= s_1^\beta \Phi^{TE,TM}(x-x',y-y') + s_2^\beta \Phi^{TE,TM}(x-x',y+y') \\ &+ s_3^\beta \Phi^{TE,TM}(x+x',y-y') + s_4^\beta \Phi^{TE,TM}(x+x',y+y') \end{aligned} \tag{B2}$$

where sign factors are as in (2.2-2) and potential $\Phi^{TE,TM}$ is,

$$\Phi^{TE,TM}(\xi,\eta) = \frac{-\eta_0 i}{8ab}\frac{1}{k}\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\frac{\varphi_{mn}^{TE,TM}}{k_{mn}} e^{ik_x\xi + ik_y\eta}, \tag{B3}$$

where η_0 is the intrinsic impedance of free space, k_{mn} is the wavenumber for the mn^{th} element of the series, and k_x and k_y are the x and y components of the wavenumber. In (B3) $\varphi_{mn}^{TE,TM}$ is dimensionless and defined by the layered media and waveguide, as in equations. Consider the case where the media consist of a single layer of thickness l and of relative permittivity ε. Let 1 denote free space and 2 dielectric.

$$\varphi_{mn}^{TE} = \frac{1}{i\gamma_{mn}^{(1)} k_{mn}} \left[1 + R_{mn}^{TE} \right], \tag{B4a}$$

$$\varphi_{mn}^{TM} = \frac{i\gamma_{mn}^{(1)}}{k_{mn}} \left[1 + R_{mn}^{TM} \right], \tag{B4b}$$

where

$$\gamma_{mn}^{(j)} = \begin{cases} \sqrt{k_0^2 \varepsilon_j - k_{mn}^2}, \text{ if } k_0^2 \varepsilon_j \geq k_{mn}^2 \\ -i\sqrt{k_{mn}^2 - k_0^2 \varepsilon_j}, \text{ if } k_0^2 \varepsilon_j < k_{mn}^2 \end{cases}$$

is the propagation constant of the mn^{th} mode inside the jth domain, $j = 1,2$. In (B4) it is assumed that the source and observation points have the same location, $z = z' = 0$, corresponding to the plane located above the air-dielectric interface. The reflection coefficients in (B4a) and (B4b) for this single layer configuration are for a waveguide terminated with a grounded substrate, and are analogous to the reflection coefficient of plane waves incident on a grounded single layer substrate, which can be found in [92].

$$R_{mn}^{TE} = \frac{[(\gamma_{mn}^{(1)})^2 - (\gamma_{mn}^{(2)})^2] i \tan(l\gamma_{mn}^{(2)})}{2\gamma_{mn}^{(1)}\gamma_{mn}^{(2)} + [(\gamma_{mn}^{(1)})^2 + (\gamma_{mn}^{(2)})^2] i \tan(l\gamma_{mn}^{(2)})}, \tag{B5a}$$

$$R_{mn}^{TM} = \frac{[(\gamma_{mn}^{(2)})^2 - \varepsilon_2^2 (\gamma_{mn}^{(1)})^2] i \tan(l\gamma_{mn}^{(2)})}{2\varepsilon_2 \gamma_{mn}^{(1)}\gamma_{mn}^{(2)} + [(\gamma_{mn}^{(2)})^2 + \varepsilon_2^2 (\gamma_{mn}^{(1)})^2] i \tan(l\gamma_{mn}^{(2)})}. \tag{B5b}$$

The key to the acceleration of the series in (B3) is the fitting of the spectra $\varphi_{mn}^{TE,TM}$ as a function of k_{mn} by exponentials, making use of the generalized pencil-of-function technique (GPOF) [93]. Herewith, the fitting is performed for such $k_{mn} > K$ that the spectra $\varphi_{mn}^{TE,TM}$ are purely real. Hence, it is,

$$\varphi_{mn}^{TE,TM} \simeq \hat{\varphi}_{mn}^{TE,TM} = \sum_{p=1}^{P} u_p^{TE,TM} \exp(-\zeta_p^{TE,TM} k_{mn}),\ k_{mn} \in (K,\infty) \tag{B6}$$

where all constants, $\zeta_p^{TE,TM}$, $p = 1,\ldots,P$, in (B6) are positive real, and $u_p^{TE,TM}$ is the residue. Then the series in (B3) with the approximate spectra $\hat{\varphi}_{mn}^{TE,TM}$ can be taken as the reference series for the application of Kummer's transformation [94]. Thus, (B3) is computed as

$$\Phi^{TE,TM}(\xi,\eta) = \frac{-\eta_0 i}{8ab}\frac{1}{k}\left[\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}(\varphi_{mn}^{TE,TM} - \hat{\varphi}_{mn}^{TE,TM})\cdot \frac{e^{ik_x\xi + ik_y\eta}}{k_{mn}} + \sum_{p=1}^{P} u_p^{TE,TM}\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\frac{e^{-\zeta_p^{TE,TM}k_{mn}}}{k_{mn}} e^{ik_x\xi + ik_y\eta}\right] \tag{B7}$$

In the above expression the series containing the difference $(\varphi_{mn}^{TE,TM} - \hat{\varphi}_{mn}^{TE,TM})$ converges very fast as soon as the indices m and n assume values such that $k_{mn} > K$. The second term is first reduced to the free space periodic DGF using Poisson summation formula [90], [95]

$$\frac{1}{4ab}\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\frac{e^{-\zeta_p^{TE,TM}k_{mn}}}{k_{mn}} e^{ik_x\xi + ik_y\eta} = \frac{1}{2\pi}\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\frac{1}{\sqrt{(\xi+2ma)^2 + (\eta+2nb)^2 + (\zeta_p^{TE,TM})^2}} \tag{B8}$$

The series in the right hand side of (B8) can be evaluated using Ewald's technique yielding exponential convergence [96].

Appendix C: Acronym Definitions

ACA - adaptive-cross-approximation

BEM - boundary element method

CAD - computer aided design

CG - conjugate gradient

CG-FFT - conjugate-gradient FFT

CMP - Calderon multiplicative preconditioner

DFT - discrete Fourier transform

DG - discontinuous Galerkin

DGF - dyadic Green's function

EDA - electronic design automation

EFIE - electric field integral equation

EM - electromagnetic

FDTD - finite difference time domain

FEM - finite element method

FFT - fast Fourier transform

FIPWA - fast inhomogeneous plane wave algorithm

FIT - finite integration technique

FVTD - finite volume time domain

GMRES - generalized minimum residual

GMT - generalized multipole technique

GPOF - generalized pencil-of-function technique

GPU - graphics processing unit

IC - integrated circuit

IE-FFT - integral equation FFT

MFO - matrix fill operations

MLFMA - multi-level fast multipole algorithm

MoL - method of lines

MoM - method of moments

MSO - matrix solve operations

PCB - printed circuit board

PEC - perfect electric conductor

PFFT - pre-corrected FFT

PMC - perfect magnetic conductor

RCS - RADAR cross-section

SLAE - system of linear algebraic equations

SM/CG - sparse-matrix/canonical grid

TLM - transmission line matrix

UFFT - unified-FFT

UFFT-GT - UFFT-grid totalizing

UFFT-P - UFFT-Precorrected

VSWR - voltage standing wave ratio

References

[1] R. F. Harrington, *Field Computation by Moment Methods,* New York, NY: Macmillan, 1968.

[2] --, The Sonnet Suite User's Manual, Sonnet Software, Inc., 2013.

[3] 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, Sep.-Oct. 1996.

[4] M. F. Catedra, R. F. Torres, J. Basterrechea, and E. Gago, *The CG-FFT Method—Application of Signal Processing Techniques to Electromagnetics*. Norwood, MA: Artech House, 1995.

[5] E. M. Dede, "Multiphysics optimization, synthesis, and application of jet impingement target surfaces," in *12th IEEE Intersoc. Conf. on Thermal and Thermomechanical Phen. in Elec. Sys.,* Las Vegas, NV, 2-5 June 2010, pp. 1-7.

[6] G. Falciasecca, "Marconi's early experiments in wireless telegraphy, 1895," *IEEE Antennas Propagat.* Mag., vol. 52, no. 6, pp. 220-221, Dec. 2010.

[7] R. Garg, *Analytical and Computational Methods in Electromagnetics*, Norwood, MA: Artech House, Inc., 2008.

[8] J. K. Lee and J. A. Kong, "Active microwave remote sensing of an anisotropic random-medium layer," *IEEE Trans. Geoscience and Remote Sens.,* Vol. GE-23, No. 6, pp. 910-923, 1985.

[9] D. Hill, *Electromagnetic Fields in Cavities*, Hoboken, NJ: John Wiley & Sons, Inc., 2009.

[10] J. C. Rautio, "An ultra-high precision benchmark for validation of planar electromagnetic analyses," *IEEE Trans. on Microw. Theory Tech.,* Vol. 42, No. 11, pp. 2046-2050, Nov. 1994.

[11] K. Rashid, E.M. Freeman, and J.A. Ramirez, "Optimisation of electromagnetic devices using computational intelligence techinques," *IEEE Trans. on Magnetics.,* Vol. 35, No. 5, pp. 3727-3729, 1999.

[12] E. N. Vasil'ev, "Excitation of smooth ideally conducting body of revolution," *Radiophysika,* vol. 2, No. 4, pp. 588-601, 1959.

[13] D. B. Davidson, *Computational Electromagnetics for RF and Microwave Engineering,* Cambridge, UK: Cambridge University Press, 2011.

[14] J. A. Kong, *Electromagnetic Wave Theory,* Cambridge, MA: EMW Publishing, 2005.

[15] S. M. Rao, D. R. Wilton, and A. W. Glisson, "Electromagnetic scattering by surfaces of arbitrary shape," *IEEE Trans. Antennas Propag.,* Vol. AP-30, No. 3, pp. 409-418, May 1982.

[16] B. A. Finlayson and L. E. Scriven, "The method of weighted residuals—a review," *Applied Mechanics Reviews*, vol 19., no. 9., pp. 735-748.

[17] K. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Trans. Antennas Propag.,* vol. 14, no. 3, p. 302-307, May 1966.

[18] T. Weiland, "On the unique numerical solution of Maxwellian eigenvalue problems in three dimensions," *Particle Aceclators*, vol. 17, pp. 227-242, 1985.

[19] P. Thoma and T. Weiland, "A subgridding method in combination with the finite integration technique," in *25th European Microw. Conf.,* Bolgna, IT, Sept. 4 1995, pp. 770-774.

[20] N. K. Madsen and R. W. Ziolkowski, "A three-dimensional modified finite volume technique for Maxwell's equations," *Electromagnetics*, vol. 10, no. 1-2, pp. 147-161, 1990.

[21] W. H. Reed and T. R. Hill, *Triangular Mesh Methods for the Neutron Transport Equation*, Tech. Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.

[22] T. Lau, E. Gjonaj, and T. Weiland, "Investigation of high order symplectic time integration methods for discontinuous Galerkin method with a centered flux," in *Workshop on Computational Electromagnetics in Time-Domain,* Perugia, Oct. 15-17 2007, pp. 1-4.

[23] P. L. Arlett, A. K. Bahrani, and O. C. Zienkiewicz, "Application of finite elements to the solution of Helmholtz's equation," *Proc. of the IEE*, Vol. 115, No. 12, pp. 1762-1766, Dec. 1968.

[24] C.F. Smith, A.F. Peterson, and R. Mittra, "The biconjugate gradient method for electromagnetic scattering," *IEEE Trans. Antennas Propag*., vol. 38, no. 6, pp. 938–940, June 1990.

[25] R. Coifman, V. Rokhlin, and S. Wandzura, "The fast multipole method for the wave equation: A pedestrian prescription," *IEEE Antennas Propagat. Mag*., vol. 35, no. 3, pp. 7–12, Jun. 1993.

[26] W. C. Chew, J. - M. Jin, E. Michielssen, and J. Song, (ed.) *Fast and Efficient Algorithms in Computational Electromagnetics*, Norwood, MA: Artech House, 2001.

[27] N. N. Bojarski, "K-Space formulation of the electromagnetic scattering problems," Air Force Avionic Laboratory, Wright-Patterson Air Force Base, OH, Tech. Rep. AFAL-TR-71-75, Mar. 1971.

[28] J. D. Morsey, V.I. Okhmatovski, and A.C. Cangellaris, "Finite-thickness conductor models for full-wave analysis of interconnects with a fast integral equation method," *IEEE Adv. Packag*., vol. 27, no. 1, pp. 24–33, Feb. 2004.

[29] J. D. Morsey, *Integral Equation Methodologies for the Signal Integrity Analysis of PCB and Interconnect Structures in Layered Media from DC to Multi-GHz Frequencies*, Ph.D. dissertation, Clemson Univ., Clemson, SC., 2003.

[30] C.-F. Wang, F. Ling, and J.-M. Jin, "A fast full-wave analysis of scattering and radiation from large finite arrays of microstrip antennas," *IEEE Trans. Ant. and Propag.*, vol 46, pp. 1467–1474, Oct. 1998.

[31] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz, "A fast integral- equation solver for electromagnetic scattering problems," in *IEEE AP-S Int. Antennas Propag. Symp. Dig.*, Seattle, WA, vol. 1, Jun. 20-24, 1994, pp. 416–419.

[32] A. Zhu and S. D. Gedney, "A quadrature-sampled precorrected FFT method for the electromagnetic scattering from inhomogeneous objects," *IEEE Antennas Wireless Propag. Lett.*, vol. 2, no. 1, pp. 50–53, 2003.

[33] A. E. Yilmaz, J.-M. Jin, and E. Michielssen, "Time domain adaptive integral method for surface integral equations," *IEEE Trans. Antennas Propag*., vol. 52, no. 10, pp. 2692–2708, Oct. 2004.

[34] F. Ling, C.-F. Wang, and J.-M. Jin, "An efficient algorithm for analyzing large-scale microstrip structures using adaptive integral method combined with discrete complex image method," *IEEE Trans. Microw. Theory Tech.*, vol. 48, pp. 832–839, May, 2000.

[35] N. Yuan, T. S. Yeo, X.-C. Nie, Y.-B. Gan, and L.-W. Li, "Analysis of probe-fed conformal microstrip antennas on finite grounded substrate, " *IEEE Trans. Antennas and Propag.*, vol. 54, no. 2, pp. 554–563, Feb. 2006.

[36] S. M. Seo, C.-F. Wang, and J.-F. Lee, "Analyzing PEC scattering structures using an IE-FFT algorithm," *Appl. Comput. Electromagn. Soc. J.*, vol. 24, no. 2, pp. 116–128, 2009.

[37] S.-Q. Li, Y.-X. Yu, K. F. Chan, C. H. Chan, and L. Tsang, "A sparse-matrix/canonical grid method for analyzing densely packed interconnects," *IEEE Trans. Microw. Theory Tech.*, vol. 49, no. 7, pp. 1221–1228, July 2001.

[38] F. Sheng and D. Jiao, "A deterministic-solution based fast eigenvalue solver with guaranteed convergence for finite-element based 3-D electromagnetic analysis," *IEEE Trans. Antennas Propag.,* vol. 61, no. 7, pp. 3701-3711, July 2013.

[39] S. Kapur and D. E. Long, "IES/sup 3/: A fast integral equation solver for efficient 3-dimensional extraction," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Washington, DC, Nov. 9-13, 1997, pp. 448-455.

[40] D. Gope and V. Jandhyala, “Pilot: A fast algorithm for enhanced 3D parasitic capacitance extraction efficiency,” in *Electrical Perf. of Elec. Packag. Dig.*, Princeton, NJ, Oct. 27-29, 2003, pp. 337–340.

[41] D. Gope and V. Jandhyala, “Efficient solution of EFIE via low-compression of multilevel predetermined interactions,” *IEEE Trans. Antennas Propag.*, vol. 53, no. 10, pp. 3324–3333, Oct. 2005.

[42] W. Hackbusch, “A sparse matrix arithmetic based on H-matrices. Part I: Introduction to H-matrices,” *Computing*, vol. 62, no. 2, pp. 89–108, 1999.

[43] S. Borm, *Efficient Numerical Methods for Non-local Operators: H2-Matrix Compression, Algorithms, and Analysis*, Zürich, CH: European Mathematical Society, 2010.

[44] W. Chai and D. Jiao, “A linear-complexity direct integral equation solver accelerated by a new rank-minimized H2-representation for large-scale 3-D interconnect extraction,” in *Proc. MTT-s Intl. Microw. Symp.*, Montreal, CA, June 2012, pp. 1–3.

[45] J. Ostrowski, Z. Andjelic, M. Bebendorf, B. Cranganu-Cretu, and J. Smajic, “Fast BEM-solution of Laplace problems with H-matrices and ACA,” *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 627–630, April 2006.

[46] K. Zhao, M. N. Vouvakis, and J.-F. Lee, "The adaptive cross approximation algorithm for accelerated method of moments computations of EMC problems," *IEEE Trans. Electromagn. Compat.*, vol. 47, no. 4, pp. 763–773, Nov. 2005.

[47] M. Bebendorf, "Approximation of boundary element matrices," *Numerische Mathematik,* vol. 86., pp. 565-589, June 2000.

[48] W. C. Chew, J.-M. Jin, C.-C. Lu, E. Michielssen, and J. M. Song, "Fast solution methods in electromagnetics," *IEEE Trans. Antennas Propag.*, vol. 45, no. 3, pp. 533–543, Mar. 1997.

[49] W. Chai and D. Jiao, "A complexity-reduced H-Matrix based direct integral equation solver with prescribed accuracy for large-scale electrodynamic analysis," in *IEEE AP-S Int. Antennas Propag. Symp. Dig.*, Toronto, CA, July, 2010, pp. 1-4.

[50] L. Jiang, *Studies on Low Frequency Fast Multipole Algorithms*, Ph.D. dissertation, Dept. Electr. Comput. Eng., Univ. Illinois at Urbana-Champaign, Champaign, IL, 2004.

[51] B. Hu and W. C. Chew, "Fast inhomogeneous plane wave algorithm for 3-D buried object problems," in *IEEE AP-S Int. Antennas Propag. Symp. Dig.*, San Antonio, TX, Jun. 2002, vol. 3, pp. 560–563.

[52] V. Jandhyala, E. Michielssen, and R. Mittra, "A memory-efficient, adaptive algorithm for multipole-accelerated capacitance computation in a stratified dielectric medium," *Int. J. Microw. Millimeter-Wave Comput.-Aided Eng.*, vol. 6, no. 6, pp. 381–390, 1996.

[53] Y. C. Pan and W. C. Chew, "A hierarchical fast-multipole method for stratified media," *Microw. Opt. Technol. Lett.*, vol. 27, no. 1, pp. 13–17, 2000.

[54] V. Rokhlin and S. Wandzura, "The fast multipole method for periodic structures," in *IEEE AP-S Int. Antennas Propag. Symp. Dig.*, Seattle, WA, vol. 1, Jun. 20-24, 1994, pp. 424-426.

[55] T. J. Cui and W. C. Chew, "Fast algorithm for electromagnetic scattering by buried 3-D dielectric objects of large size," *IEEE Trans. Geosci. Remote Sens.*, vol. 37, no. 5, pp. 2597–2608, Sep. 1999.

[56] V. Okhmatovski, J.D. Morsey, and A.C. Cangellaris, "Loop-tree implementation of the adaptive integral method for numerically stable broadband fast electromagnetic modeling," *IEEE Trans. Antennas Propag.*, vol. 52, no. 8, pp. 2130–2140, Aug. 2004.

[57] K. Yang and A.E. Yilmaz, "A three-dimensional adaptive integral method for scattering from structures embedded in layered media," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 4, pp. 1130–1139, Apr. 2012.

[58] V.I. Okhmatovski and A.C. Cangellaris, "Fast electromagnetic analysis of dense shielded integrated circuits using the adaptive integral method (AIM)," in *Proc. IEEE MTT-s Int Symp. Dig.*, Phoenix, AZ, May 2001, vol. 3, pp. 1929–1932.

[59] V. Okhmatovski, M. Yuan, I. Jeffrey, and R. Phelps, "A three-dimensional precorrected FFT algorithm for fast method of moments solutions of the mixed-potential integral equation in layered media," *IEEE Trans. Microw. Theory Tech.*, vol. 57, no. 12, pp. 3505–3517, Dec. 2009.

[60] J. M. Taboada, M. G. Araujo, F. O. Basterio, J. L. Rodriguez, and L. Landesa, "MLFMA-FFT parallel algorithm for the solution of extremely large problems in electromagnetics," *Proc. of the IEEE,* Vol. 101, No. 2, pp. 350-363, Jan. 2013.

[61] V. Dang, Q. M. Nguyen, and O. Kilic, "GPU cluster implementation of FMM-FFT for large-scale electromagnetic problems," *IEEE Antennas Wireless Propag. Lett.,* vol. 13, pp. 1259-1262, June 2014.

[62] K. Yang and A. E. Yilmaz, "An FFT-accelerated integral-equation solver for analyzing scattering in rectangular cavities," *IEEE Trans. Microw. Theory Tech.,* vol. 62, no. 9, pp. 1930-1942, Sep. 2014.

[63] *Intel Visual FORTRAN for Windows,* Intel, Santa Clara, CA, 2013.

[64] *MATLAB R2013a*, The Mathworks, Inc., Natick, MA, 2013.

[65] --, "SonnetLab Toolbox for MATLAB®, http://www.sonnetsoftware.com/support/sonnet-suites/sonnetlab.html, 2013.

[66] D. D. Kalamkar, J. D. Trzasko, S. Sridharan, M. Smelyanskiy, D. Kim, A. Manduca, Y. Shu, M. A. Bernstein, B. Kaul, and P. Dubey, "High performance non-uniform FFT on modern x86-based multi-core systems," in *Proc. IEEE 26th Int. Parallel and Distrib. Proc. Symp.,* Shanghai, CN, May 21-25, 2012, pp. 449-460.

[67] G. V. Eleftheriades, J. R. Mosig, and M. Guglielmi, "A fast integral equation technique for shielded planar circuits defined on nonuniform meshes," *IEEE Trans. Microw. Theory Tech.*, vol. 44, no. 12, pp. 2293–2296, Dec. 1996.

[68] J. C. Rautio, *A Time–Harmonic Electromagnetic Analysis of Shielded Microstrip Circuits*, Ph.D. dissertation, Elect. Comput. Eng. Dept., Syracuse Univ., Syracuse, NY, 1986.

[69] M. Abramovitz and I. A. Stegun, *Handbook of Mathematical Functions*, New York, NY: Dover, 1965.

[70] Y. L. Chow, J.J. Yang, D.G. Fang, and G.E. Howard, "A closed-form spatial Green's function for the thick microstrip substrate," *IEEE Trans. Microw. Theory Tech.*, vol. 39, no. 3, pp. 588–592, Mar. 1991.

[71] A. Alparslan, M.I. Aksun, and K.A. Michalski, "Closed-form Green's functions in planar layered media for all ranges and materials," *IEEE Trans. Microw. Theory Tech.*, vol. 58, no. 3, pp. 602–613, Mar. 2010.

[72] D.R. Wilton, S.M. Rao, A.W. Glisson, D.H. Schaubert, O. Al-Bundak, and C.M. Butler, "Potential integrals for uniform and linear source distributions on polygonal and polyhedral domains," *IEEE Trans. Antennas Propag.*, vol. 32, no. 3., pp. 276–281, Mar. 1984.

[73] A. E. Yilmaz, "A two-scale AIM for fast solution of volume integral equations," in *Proc. Appl. Comput. Electromagn*. Soc. Conf., Monterey, CA, Mar. 2009, pp. 511–516.

[74] Y. Saad, *Iterative Methods for Sparse Linear Systems,* Philadelphia, PA: Society for Industrial and Applied Mathematics, 2003.

[75] F. P. Andriulli, K. Cools, H. Bagci, F. Olyslager, A. Buffa, S. Christiansen, and E. Michielssen, "A multiplicative Calderon preconditioner for the electric field integral equation," *IEEE Trans. Antennas Propag.*, vol. 56, no. 8, pp. 2398-2412, Aug. 2008.

[76] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Baltimore, MD: Johns Hopkins University Press, 1996.

[77] E. Cuthill and J. McKee, "Reducing the bandwidth of sparse symmetric matrices," in *Proc. Of the 24th National ACM Conf.*, New York, NY, 1969, pp. 157-172.

[78] F. Ling and J.-M. Jin, "Full-wave analysis of multilayer microstrip problems," in *Fast and Efficient Algorithms in Computational Electromagnetics*, W. C. Chew, J.-M. Jin, E. Michielssen, J. Song, Eds. Norwood, MA: Artech House, pp. 729–772, 2001.

[79] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in Fortran 77*, Cambridge, UK: Cambridge University Press, 1999.

[80] Wave3D CAD Manual, CEMWorks, 2013.

[81] N. Miyazawa, "Device having interdigital capacitor," U.S. Patent 6 949 811, September 27, 2005.

[82] B. J. Rautio, Q. Long, A. Agrawal, and M. A. El Sabbagh, "Simulation geometry rasterization for applications toward graphene interconnect characterization," in *Proc. IEEE Intl. Symp. on Electromag. Compat,* Pittsburgh, PA, Aug. 6-10, 2012, pp. 406-410.

[83] M. Burton and S. Kashyap, "A study of a recent, moment-method algorithm that is accurate to very low frequencies," *Appl. Computational Electromagn. Soc. J.*, vol. 10, no. 3, pp. 58-69, Nov. 1995.

[84] W. Wu, A. W. Glisson, and D. Kajfez, "A comparison of two low-frequency formulations for the electric field integral equation," in *Proc. 10th Annu. Review of Progress in Applied Comp. Electromagnetics*, vol. 2, Montery, CA, Mar. 1994, pp. 484-491.

[85] J. Ostrowski, M. Bebendorf, R. Hiptmair, and F. Kramer, "*H*-matrix-based operator preconditioning for full Maxwell at low frequencies," *IEEE Trans.* Magn., vol. 46, no. 8, pp. 3193-3196 , August 2010.

[86] A. Buffa and S. Christiansen, "A dual finite element complex on the barycentric refinement," *Math. Comput.*, vol. 76, no. 260, pp. 1743–1769, Oct. 2007.

[87] P. Subbulakshmi and R. Rajkumar, "Design and characterization of corporate feed rectangular microstrip patch array antenna," in *2013 IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology,* Chennai, India, Mar. 25-26, 2013, pp. 547-552.

[88] --, "Find minimum of unconstrained multivariable function using derivative-free method," http://www.mathworks.com/help/matlab/ref/fminsearch.html, 2013.

[89] "Port Tuning with EM," http://www.sonnetsoftware.com/resources/IMS-port-tuning.html, 2013.

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

[91] J.R. Wait, *Electromagnetic Wave Theory*, New York, NY: Harper & Row Publishers, 1985.

[92] B. M. Notaros, *Electromagnetics*, Upper Saddle River, NJ: Pearson Education, Inc., 2011.

[93] T.K. Sarkar and O. Pereira, "Using the matrix pencil method to estimate the parameters of a sum of complex exponentials," *IEEE Antennas Propagat. Mag*., vol. 37, no. 1, pp. 48–55, Feb. 1995.

[94] M.-J. Park and S. Nam, "Rapid calculation of the Green's function in the shielded planar structures," *IEEE Microw. Guid. Wave Lett.*, vol. 7, no. 10, pp. 326–328, Oct. 1997.

[95] K. Knopp, *Theory and Application of Infinite Series*, New York, NY: Hafner Publishing Company, Inc., 1971.

[96] K.E. Jordan, G.R. Richter and P. Sheng, "An efficient numerical evaluation of the Green's function for the Helmholtz operator on periodic structures," *J. Comput. Phys.*, vol. 63, pp. 222–235, 1986.

Biography

Brian J. Rautio completed the B.S.E.E. degree from Rensselaer Polytechnic Institute in Troy, NY in 2009, and the M.S.E.E from Syracuse University in 2011. He completed internships with Sonnet Software, North Syracuse, NY in 2005, 2007 and (part time) 2008, and with Advanced Micro Devices, Austin, TX, in 2006. He has been a student member of IEEE since 2006.

In 2007, he was made a member of the Eta Kappa Nu Electrical Engineering Honor Society. He received honorable mention in the 2009 Lemelson Rensselaer $30 000 Student Prize Competition, the only undergraduate to receive acknowledgement. He is also the winner of the 2012 L.C. Smith Prize for Engineering and Technology Writing.

He is currently employed by Sonnet Software, North Syracuse, NY, where he is commercializing significant components of the research investigated within this dissertation.

Presented Conference Papers

- B. J. Rautio, V. Okhmatovski, and J. K. Lee, "Novel high precision UFFT methodology for fast analysis of 3D planar circuits embedded in shielded layered media," in *IEEE International Symposium on Antennas & Propagation and USNC/URSI Radio Science Meeting,* Memphis, TN, July, 2014.

- B. J. Rautio, V. Okhmatovski, and J. K. Lee, "The UFFT method proposed for optimization of a corporate feed network patch antenna array," in *30th Int. Review of Prog. In Appl. Comput. Electromagnetics*, Jacksonville, FL, Mar. 23-27, 2014, pp. 1-3.
- B. J. Rautio, V. Okhmatovski, and J. K. Lee, "A structured workflow for developing interoperable numerical analysis tools," in *IEEE AP-S Int. Antennas Propag. Symp. Dig.,* Orlando, FL, Jul. 7-13, 2013, pp. 922-923.
- B.J. Rautio, V. I. Okhmatovski, and J. K. Lee, "Fast 3D planar electromagnetic analysis via Unified-FFT method," in *Proc. IEEE MTT-S Int Symp. Dig.*, Seattle, WA, June 2-7, 2013, pp. 1–3.
- B. J. Rautio, Q. Long, A. Agrawal, M. A. El Sabbagh, "Simulation geometry rasterization for applications toward graphene interconnect characterization," in *2012 International Symposium on Electromagnetic Compatibility*, Pittsburgh, PA, Aug. 5-10, 2012.
- B. J. Rautio, V. Okhmatovski, J. K. Lee, and A. Cangellaris, "Unified FFT based acceleration of near and far interactions in moment method for microstrip circuits embedded in shielded multilayered media," in *IEEE International Symposium on Antennas and Propagation and USNC/URSI National Radio Science Meeting*, Chicago, IL, Jul. 8-14, 2012.
- B. J. Rautio, M. El Sabbagh, and J.C. Rautio, "Detailed error analysis and automation of the RA resonator technique for measurement of uniaxial anisotropic permittivity," in 78th *ARFTG Microwave Measurement Conference*, Phoenix, AZ, Dec. 2, 2011.
- B. J. Rautio, M. El Sabbagh, and J. C. Rautio, "Broadband analysis and characterization of anisotropic dielectric temperature dependence," in *IEEE Conference on Ultra Wideband*, Bologna, Italy, Sept. 14-16, 2011, pp. 541-544.

- J. C. Rautio and B. J. Rautio, "High accuracy broadband measurement of anisotropic dielectric constant using a shielded planar dual mode resonator," in *74th ARFTG Microwave Measurement Conference*, Boulder/Broomfield, CO, Dec. 1-4, 2009.

Journal Articles

- B. J. Rautio, V. I. Okhmatovski, and J. K. Lee, "The unified-FFT grid totalizing algorithm for fast $O(N \log N)$ method of moments analysis with accuracy to machine precision," *IEEE Trans. Microw. Theory Tech*, (In Review).
- B. J. Rautio, V. I. Okhmatovski, A. C. Cangellaris, J. C. Rautio, and J. K. Lee, "The unified-FFT algorithm for fast electromagnetic analysis of planar integrated circuits printed on layered media inside a rectangular enclosure," *IEEE Trans. Microw. Theory Tech*, Vol. 62, No. 5, pp. 1112-1121, May 2014.
- J. C. Rautio, R. L. Carlson, B. J. Rautio and S. Arvas, "Shielded dual mode microstrip resonator measurement of uniaxial anisotropy," *IEEE Transactions on Microwave Theory and Techniques*, vol 59, no. 3, pp. 748-754, Mar. 2011.
- J. C. Rautio, M. R. LeRoy and B. J. Rautio, "Synthesis of perfectly causal parameterized compact models for planar transmission lines," *IEEE Transactions on Microwave Theory and Techniques*, vol. 57, no. 12, pp. 2938-2947, Dec. 2009.

Magazine Articles and Workshop Presentations

- John Coonrod and Brian Rautio, "Comparing Microstrip and CPW Performance," *Microwave Journal*, July 2012.
- Brian Rautio, "State of the Art and Future Directions in Electromagnetic Simulation," *IMS 2012 MicroApps*, June 2012.

- Brian Rautio, “Boffins, Tech, and RADAR, Oh My!,” 6th Annual *Stone Canoe*, L.C. Smith Prize for Engineering and Technology, 2012.
- Yun Chase and Brian Rautio, "Cluster Computers Speed EM Simulation," *Microwaves & RF*, November 2005.