Assessing the Benefits of DCT Compressive Sensing for Computational Electromagnetics

by

Kristie D'Ambrosio

[S.B. Computer Engineering, M.I.T., 2010]

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

April 2011

Author____

Department of Electrical Engineering and Computer Science
April 28, 2011

Certified by____

Patrick H. Winston
Ford Professor of Artificial Intelligence and Computer Science
M.I.T. Thesis Supervisor

Accepted by____

Dr. Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Assessing the Benefits of DCT Compressive Sensing for

Computational Electromagnetics

by
Kristie D'Ambrosio

Submitted to the
Department of Electrical Engineering and Computer Science

April, 2011

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Computational electromagnetic problems are becoming exceedingly complex and traditional computation methods are simply no longer good enough for our technologically advancing world. Compressive sensing theory states that signals, such as those used in computational electromagnetic problems have a property known as sparseness. It has been proven that through under sampling, computation runtimes can be substantially decreased while maintaining sufficient accuracy. Lawrence Carin and his team of researchers at Duke University developed an in situ compressive sensing algorithm specifically tailored for computational electromagnetic applications. This algorithm is known as the discrete cosine transform (DCT) method. Using the DCT algorithm, I have developed a compressive sensing software implementation. Through the course of my research I have tested both the accuracy and runtime efficiency of this software implementation proving its potential for use within the electromagnetic modeling industry. This implementation, once integrated with a commercial electromagnetics solver, reduced the computation cost of a single simulation from seven days to eight hours. Compressive sensing is highly applicable to practical applications and my research highlights both its benefits and areas for improvement and future research.

Thesis Supervisor: Patrick Winston

Title: Ford Professor of Artificial Intelligence and Computer Science

Acknowledgements

I would first like to thank my thesis advisor, Professor Patrick Winston, for all his help and support that enabled me to pursue my research unabated. Professor Winston chose to accept me as an advisee under the knowledge that I would not be pursuing a research topic within his expertise. I am exceedingly grateful for his decision to believe in me and my work which ultimately made this thesis possible.

I would also like to thank my mentor, Dr. Ronald Pirich, for his guidance and endless expert advice. For four years Ron has continually encouraged my creativity and pushed me to achieve more than I ever believed possible. He has broadened my horizons and ultimately made me a better engineer. I am very excited to begin my career with Ron’s team upon my graduation.

I would like to give special thanks to my family, without whom my life would hold no meaning. My mother, Randi, and father, Michael, have sacrificed so much to ensure that I get the best education possible. They are the greatest role models a daughter could ever ask for. My sisters, Danielle and Marissa, have been my greatest allies during some of the most trying times in my life. I am eternally grateful to have both of you in my life. My grandmother, Awa, always encouraged my creative spirit and gave me the greatest gift of all, passion. I am blessed to have been given such an incredible group of people to call my family.

Finally, I would like to thank my fiancé, Eric Correll. Without him I would not have survived my final year of MIT. His unwavering support and love has made everything seem possible. I am inexpressibly excited to begin the rest of our lives together.

Contents

1 Introduction 8

2 Compressive Sensing 10

2.1 Compressive Sensing Theory 10

2.2 Algorithmic Background 12

2.3 Applications 14

3 Electromagnetic Modeling 14

3.1 Computational Electromagnetic Methods 15

3.2 Complexities of Electromagnetic Modeling 16

3.3 Electromagnetic Modeling Software: FEKO 17

4 DCT Algorithm 19

4.1 DCT Compressive Sensing Theory 19

4.2 Mathematical Formulation 21

4.3 Application in Electromagnetic Modeling 22

5 Fortran-Based DCT Implementation 23

5.1 Accuracy Testing 23

5.1.1 Theoretical Accuracy 24

5.1.2 Practical Accuracy 25

5.2 Efficiency Testing 27

5.2.1 Phase1: Cylinder 27

5.2.2 Phase 2: VFY218 29

5.3 Analysis and Conclusions 30

6 FEKO-DCT Integration 31

6.1 How FEKO Works 32

6.2 High-Level Implementation Description 34

6.3 In-Depth Implementation Description....35
6.3.1 Front-End....35
6.3.2 Back-End....38
7 Runtime Efficiency of FEKO-DCT Integration....39
7.1 Phase 1: Cylinder....40
7.2 Phase 2: VFY218....41
7.3 Analysis and Conclusions....43
8 Future Advancements in Compressive Sensing....45
9 Contributions....48

Appendix A....50
A.1 Runtime Data for Fortran-Based DCT Implementation....50
A.2 Runtime Data for FEKO-DCT Implementation....51

Bibliography....52

List of Figures

1 VFY218 Mesh Model 16

2 Haar Function..25

3 Discrete Cosine Function...25

4 VFY218 RCS at 200 MHz...26

5 Cylinder Mesh Model 28

6 Fortran-Based DCT – Cylinder Runtime..28

7 Fortran-Based DCT – VFY218 Runtime...30

8 High-Level Implementation Schematic 34

9 Front-End Implementation Schematic 36

10 Back-End Implementation Schematic...38

11 FEKO-DCT Integration – Cylinder Runtime 40

12 FEKO-DCT Integration – VFY218 Runtime 42

List of Acronyms

BCS – Bayesian Compressive Sensing

CEM – Computational Electromagnetics

CS – Compressive Sensing/Compressive Sampling

DCS – Distributive and multi-sensor Compressive Sensing

DCT – Discrete Cosine Transform

EM - Electromagnetic

FEKO – Field Computation for Objects of Arbitrary Shapes

FEM – Finite Element Method

FMM – Fast Multipole Method

MBCS – Model Based Compressive Sensing

MLFMM – Multi Level Fast Multipole Method

MoM – Method of Moments

RCS – Radar Cross Section

RVM – Relevance Vector Machine

Chapter 1

Introduction

The design of present and future electromagnetic systems, in a complex and increasingly electronic world, is going to depend upon the speed and efficiency of our computational systems. One way to increase the efficiency of our computational systems is by decreasing the runtime of signal processing computations. It has long been understood that signals possess sparseness properties. This allowed for the development of a technique, known as compressive sensing or sampling (CS), to emerge as an optimization method for signal processing. Many different CS algorithms have been developed over the last two decades that highlight the benefits of exploiting such sparseness properties.

In this thesis I document the implementation and testing of one particular compressive sensing algorithm designed specifically for electromagnetic simulation applications. This algorithm, known as the discrete cosine transform (DCT) method was developed by a team of researchers at Duke University. Led by Professor Lawrence Carin, the team theoretically argued that the DCT algorithm would provide substantial, time-reducing benefits to the field of electromagnetic modeling. I tested that theoretical argument by implementing the DCT algorithm and showed my implementation dramatically reduce the runtime of electromagnetic scattering problems such as radar cross section (RCS) analyses.

In order to assess that the DCT compressive sensing algorithm is beneficial to a broad range of electromagnetic, signal processing applications it must be tested in a comparative study. A stand-alone implementation of the DCT compressive sensing method was tested in a comparative study alongside a commercial method for computing electromagnetic simulations. The commercial method that will be used in this study is an industry-accepted, commercial software product known as FEKO (Field Computation for Objects of Arbitrary Shape). The FEKO software platform has a well-optimized electromagnetic solver and currently has the capability to run some of the fastest electromagnetic modeling simulations.

I tested an implementation of the DCT method against the industry standard to establish that the DCT method is an accurate technique for computing scattering problems and large-scale simulations. More importantly, I compared the runtimes of the two computation methods to determine the efficiency benefits of the DCT method in a practical application.

Throughout the course of this study I discovered that a stand-alone DCT implementation with a home-grown electromagnetics solver could not reduce runtime enough to successfully compete with the optimized, commercial software. Therefore, I integrated the DCT method with the FEKO electromagnetics solver. This integration brought together the benefits of compressive sensing theory as well as the widely-accepted optimization benefits of the FEKO electromagnetics software. The results of this integration implementation were drastic. My compressive sensing software was able to reduce computation runtime from 164 hours, almost 7 days, to approximately 8 hours.

Using compressive sensing in electromagnetic modeling has yet to become mainstreamed in industry. I showed the benefits of such theoretical integration with this thesis and prove that it is both accurate and efficient.

Chapter 2

Compressive Sensing

Signal processing theory states that the rate at which signals must be sampled in order to capture all of the information of that signal is equal to twice the Fourier bandwidth of the signal (Nyquist rate). This sampling method produces a large amount of data with a large amount of redundant information. However, researchers have found that many important signals have a property called sparseness, thus allowing the number of samples required to capture all of the signal's information to be reduced. This approach, called compressive sensing or sampling (CS) can be used to exploit signal sparseness and allow signals to be under sampled without the loss of information. [3]

2.1 Compressive Sensing Theory

Compressive sensing has the ability to effectively use randomization to minimize the redundancy inherently found in electromagnetic computation. In traditional signal sampling, the finite bandwidth of a signal is exploited to allow all of the information of the signal to be captured when it is sampled at even intervals. However, the goal of compressive sensing is to reconstruct a signal accurately and efficiently from a set of few non-adaptive linear measurements. Using linear algebra it can be shown that it is not possible to reconstruct an arbitrary signal from an incomplete set of linear measurements. Thus, the problem must restrict the domain in which the signal belongs. By doing so, the problem now only considers sparse signals, those with few non-zero coordinates.

Not all signals are inherently sparse but many can be transformed into some basis in which most of the transform coefficients are negligible, thus making them sparse within this basis domain. For example, a signal's spectrum, obtained by applying a Fourier transform, may be sparse, in which only a few of the Fourier coefficients would be non-zero. It is now known that many signals such as real-world images or audio signals are sparse either in this sense, or with respect to a different basis. Sparse signals lie in a lower dimensional space, which means that it can be represented by few linear measurements. Compressive sensing reduces the number of sampling points which directly corresponds to the amount of data collected. In addition, CS operates under the assumption that it is possible to directly acquire just the important information about the signals. This means that in effect, the part of the data that would get thrown away is never acquired. This enables the creation of net-centric and stand-alone applications with many fewer resources.

A surprising aspect of CS is that it does not attempt to measure the dominant coefficients of a transformed signal. Rather, weighted combinations of all wavelet coefficients are measured. Another surprising aspect of CS is that one may expect that the aforementioned weights should adapt to the signal under test. However, CS theory indicates that a fixed set of weights may be used for all signals within a given class, and that the quality of the CS estimate of that signal, based on these measurements, will not be significantly worse than the best adaptive measurements. This property makes an implementation much simpler since weights can be predetermined for each signal class without the need for adaptive calculations. [6]

2.2 Algorithmic Background

Every compressive sensing algorithm stems from the basic theory described above. In general terms, the algorithmic mathematical process can be described as follows: Suppose x is an unknown vector in R. Given the context of the problem, R could be a signal or image. We want to acquire the data contained in vector x though sampling and then reconstruct it to obtain the complete vector. Traditionally, this should require m samples, where m adheres to the Nyquist rate. However, suppose that it is known beforehand that x is compressible by transform coding with a known transform. This means we are allowed to acquire data about x by measuring n general linear functions rather than m. If the collection of linear functions is well-chosen, and we allow for a degree of reconstruction error, the size of n can be dramatically smaller than the size m usually considered necessary. Thus, certain natural classes of signals or images with m data points need only $n = O(m \cdot \log_{5/2}(m))$ non-adaptive samples for faithful recovery, as opposed to the usual m samples.

Over the past two decades there have been many different adaptations of compressive sensing theory. Each is different but all are perfectly acceptable methods. Described below are three of the major variations of CS algorithms found in today's literature. However, even within each category, every developer has a unique version of the technique.

Distributed and Multi-Sensor Compressive Sensing (DCS) is a branch of compressive sensing that enables new distributed coding algorithms for multi-signal ensembles. This method exploits correlation structures within and between the signals. DCS theory rests on a new concept called the joint sparsity of a signal ensemble. There are many algorithms for joint recovery of multiple signals from incoherent projections which and characterize the number of measurements per sensor required for accurate reconstruction. DCS is a framework for distributed compression of sources with memory, which has remained a challenging problem for some time. [2]

Bayesian Compressive Sensing (BCS) is a Bayesian framework for solving the inverse problem of compressive sensing which remains extremely challenged. The basic BCS algorithm adopts the relevance vector machine (RVM) and can marginalize the noise variance with improved robustness. Besides providing a Bayesian solution, the Bayesian analysis of CS, more importantly, provides a new framework that allows a user to address a variety of issues that have not been addressed in other compressive sensing algorithms. These issues include: a stopping criterion for determining when a sufficient number of CS measurements have been performed, as well as adaptive design of the projection matrix and simultaneous inverse of multiple related CS measurements. [8]

Model Based Compressive Sensing (MBCS) theory parallels the conventional theory and provides concrete guidelines on how to create model-based recovery algorithms with provable performance guarantees. Performance guarantees are extremely beneficial for the application of compressive sensing in industry-quality software. A highlight of this technique is the introduction of a new class of compressible signals and a new sufficient condition for robust model compressible signal recovery. The reduction of the degrees of freedom of a compressible signal by permitting only certain configurations of the large and small coefficients allows signal models to provide two immediate benefits to CS. The first benefit is that MBCS enables a user to reduce, in some cases significantly, the number of measurements M required to recover a signal. The second benefit is that, during signal recovery, MBCS enables a user to better differentiate true signal information from recovery artifacts, which leads to a more robust recovery. [1]

2.3 Applications

The benefit of compressive sensing is that, by processing fewer data points, the computation time of electromagnetic simulations can be drastically reduced. Compressive sensing or sampling has many applications with computationally intense problems. The problem of interest for this research is electromagnetic modeling, specifically radar cross section (RCS) analyses. Signal processing systems use first principle electromagnetic codes, such as Method of Moments (MoM) and Fast Multipole Method (FMM), to determine solutions to computational electromagnetic problems. The next chapter will discuss the use of first principle electromagnetic codes and the challenges faced today for complex modeling and simulation problems.

Chapter 3

Electromagnetic Modeling

Computational electromagnetics (CEM) or electromagnetic modeling is the process of modeling the interaction of electromagnetic fields with physical objects and the environment. It typically involves using computationally efficient approximations to Maxwell's equations and is used to calculate antenna performance, electromagnetic compatibility, radar cross section and electromagnetic wave propagation when not in free space. [13]

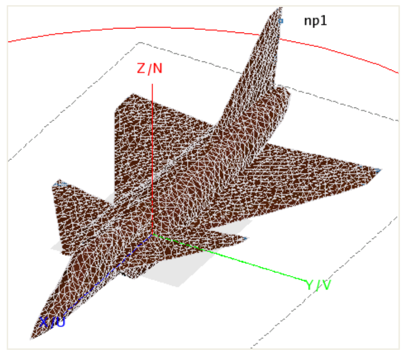
3.1 Computational Electromagnetic Methods

The method of moments (MoM) is a numerical computation method of solving linear partial differential equations which have been formulated as integral equations. MoM is characterized as an integral solver method because of its integral form. Conceptually, it works by constructing a mesh over the modeled surface and calculates boundary values. By limiting the calculation to boundary values, rather than values throughout the space, it is significantly more efficient in terms of computational resources for problems with a small surface to volume ratio.

Compression techniques (*e.g.* multipole expansions or adaptive cross approximation/hierarchical matrices) can be used to ameliorate efficiency problems, though at the cost of added complexity and with a success-rate that depends heavily on the nature and geometry of the problem. Fast multipole method (FMM) is an alternative to MoM. It is an accurate simulation technique and requires less memory and processor power than MoM. The FMM was first introduced by Greengard and Rokhlin and is based on the multipole expansion technique. FMM can also be used to accelerate MoM. [13]

Throughout the experiments in this thesis I focused on one problem-of-choice: radar cross section (RCS) analyses. Radar cross sections are a commonly calculated problem for engineers in the electromagnetic modeling industry and involve some of the most complex and time-consuming computations. For example, Figure 1 is a model of a generic aircraft, known as the VFY218. This model has hundreds of thousands of mesh elements for problems of relatively low frequencies. Determining solutions to RCS problems for aircrafts, such as the VFY218 are common practice in the aerospace industry. Because of these reasons I felt that RCS calculations would best highlight the benefits of compressive sensing as well as prove that practical applications are possible. [12]

Figure 1: VFY218 Mesh Model



3. 2 Complexities of Electromagnetic Modeling

Engineers in the computational electromagnetic industry face a series of complex set-backs that make finding a solution to CEM problems extremely difficult, if not impossible. The use of a mesh model as described above can create complex boundary conditions if not formulated correctly. Many CEM simulations can take days to compute and if the mesh is incomplete or has a boundary inconsistency, the problem may not be forseen until a solution is tried and fails. A user could spend days waiting for the software platform to determine that a solution cannot be found given the defined mesh model. Therefore, finding the RCS of complex objects can be some of the trickiest and most time consuming simulation and modeling problems in the field today. While more time efficient computation techniques will not improve the user's ability to create a complete and usable mesh model, it will allow users to determine if a problem exists sooner.

RCS analyses at low frequencies (such as 100 MHz) do not provide as much information about the shape of the object as an RCS computed at a high frequency (3 GHz). In order to complete the analysis at a higher frequency, the mesh model being used for the computation must be of a finer grain. Computers today have the capability to have terabytes of memory and many processing cores working simultaneously. However, despite these advancements, large and complex mesh models could have millions of elements that need to be processed. Even with the advancements in technology, the user's needs are growing faster than our computers can handle. Some problems can still take a week or two to complete at such high frequencies.

CEM is an extremely complex field with many shortcomings. I have described two of the major, time consuming problems faced in today's industry that can be remedied by the use of time-reducing analysis methods such as compressive sensing.

3.3 Electromagnetic Modeling Software: FEKO

FEKO is a commercial computational electromagnetics software suite that is used in a wide range of industrial application such as antenna placement, scattering analyses, and radome modeling. FEKO will act as the control in our accuracy and efficiency tests. I chose FEKO over other software on the market for three main reasons. First, FEKO is an industry standard in the CEM modeling field. It also provides the user with substantial information about its own accuracy. This allows us to be certain that any DCT-based solution with marginal error from a FEKO solution will be industry acceptable.

Second, FEKO has some of the most advanced Method of Moment based CEM solvers on the market today. This is because the software platform is most tailored to integral solver methods rather than FEM solvers. For scattering problems, such as radar cross section analyses, this methodology is ideal. The speed of the FEKO solver will prove useful later on in our DCT experimental analysis when we want to test efficiency.

FEKO offers users a wide spectrum of numerical methods and hybridizations, each suitable for a specific range of applications. The two methods described below are integral solver methods and are useful for solving RCS problems. Method of Moments (MoM) is the most simplistic of all the integral solver methods. However, it is the ideal solution method for radiation and coupling analyses and can out-perform more advanced methods with certain applications. Multi-level Fast Multipole Method (MLFMM) is a more complex and advanced integral solver method that is built off of the more simplistic FMM framework. It is the ideal method for electrically large, full-wave analyses. FEKO provides several other hybrid solver methods which I will not discuss in this this since they are not useful for our application. [11]

FEKO was the first software platform to use the MLFMM solver method which was hailed for its time-reduction capabilities. The traditional MoM treats each of the N basis functions in isolation, thus resulting in an N^2 scaling of memory requirements and N^3 in CPU. Therefore, it is clear that processing requirements for MoM solutions scale rapidly with increasing problem size. The MLFMM formulation's more efficient treatment of the same problem results in Nlog(N) scaling in memory and Nlog(N)log(N) in CPU time. In real applications this reduction in solution requirements can range to orders of magnitude. [10]

The FEKO software makes it simple for users to switch between computation methods given the needs of the specific problem in question. This flexibility makes FEKO one of the most time efficient software platforms for computational electromagnetics. In addition, users can make many manual optimizations to the FEKO solver in order to further increase runtime efficiency. The iterative stopping criterion, or convergence level, determines the accuracy of the result. The default value for this condition is extremely generous and users can manually alter the value in order to decrease runtime while still ensuring a known and acceptable accuracy for their simulation. The FEKO software and other CEM software platforms like FEKO are complex and advanced systems that have been proven effective and relatively efficient in industry for years. One of the challenges of this research is to find a time saving CEM solution that can outperform the FEKO software.

Chapter 4

DCT Algorithm

Researchers at Duke University, under the direction and supervision of Professor Lawrence Carin, have developed a new, in situ compressive sensing theory that uses a discrete cosine transform basis function. This theory, as described below, formed the basis of research for this thesis. All implementations of the DCT method, as described in chapters 5 and 6, are exact mathematical implementations of the theory presented by Carin and his team. In this way, I ensure that the accuracy of each of my tests is supported by the theoretical accuracy of Carin's research as presented in his published papers.

4.1 DCT Compressive Sensing Theory

When a target is situated in the presence of a complicated background medium, the frequency-dependent multi-static fields scattered from the target can be measured in a CS manner. The CS measurements are performed by exploiting the natural wave propagation of the incident and scattered fields. This method may be employed to reduce the number of computations required in numerical linear scattering analyses. [5]

A given source in the presence of the background medium yields a weighted set of plane waves traveling at different angles Φ_i in the presence of the target, and the induced scattered fields propagate away from the target at multiple angles Φ_s , and energy from the multiple of these scattering angles arrive at a given receiver, with the different received, Φ_s components weighted by the associated characteristics of the corresponding propagation medium.

This is called in situ compressive sensing because the background medium and the associated Green's function are used to constitute the CS projections on the angle-dependent target scattering response. This technique uses a discrete cosine transform rather than the wavelet transform because it was determined that the DCT yielded better sparseness that a wavelet basis; this is attributed to the fact that often the angle-dependent scattering fields constitute a very smooth function, and therefore the localizing properties of wavelets are less necessary.

The interesting aspect is that, by exploiting a highly multi-path environment, the resolution with which the fields are focused at the source may be much higher than that expected of the same physical antenna aperture situated in vacuum. This can be referred to as super resolution. Both time reversal and in situ compressive sensing exploit the complex propagation medium to realized related effects. [4]

4.2 Mathematical Formulation

The section above describes the in situ compressive sensing theory whereas this section will explain the mathematical foundation behind the method. All implementations of the DCT compressive sensing method in this paper follow the prescription described below.

I begin with a common task of computing the scatter fields from the N plane-wave incidences. In this case we will take N to equal 512. Each of the N waves has amplitude of 1 and ranges from 0 to 180 degrees at the following angles:

$$\Phi_0 = 0, \quad \Phi_1 = 180/N, \quad \Phi_2 = 180*(2/N), \quad \ldots \quad , \Phi_{N-1} = 180 - 180/N$$

Here we model the observation of scattered waved at the receiver which is at a fixed angle from the target. We will let u_n denotes that scatter field due to the n^{th} plane wave incident. Once plane waves reach the receiver, I perform the discrete cosine transform for the scattering fields u_n. In this case, $n \in [0, N-1]$ and θ_k are the DCT coefficients according to the following formula:

$$\theta_k = \sum_{n=0}^{N-1} u_n \cos\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right)$$

Once the DCT coefficients θ_k are calculated we find that there are M low-frequency components of those coefficients which are dominant or non-zero. It is important to realize that M is much smaller than the number of initial scattered angles N. Given these M DCT components, the scatter field u_n can be reconstructed very accurately later on.

Next we use a first principle electromagnetic code, such as the fast multipole method (FMM) to compute the M coefficients. In this calculation only M realizations are used with the multiple plane-wave incidences in each realization. Because this computation only requires M realizations, the computation time can be drastically reduced.

The final step of the DCT compressive sensing method is to compute the inverse DCT to get the scatter field due to each of the 512 plane-wave incidences with amplitude 1. [4]

4.3 Application in Electromagnetic Modeling

Linear scattering analyses, such as radar cross section (RCS) analyses result in a matrix equation of the form Zi = v where i represents a column vector of basis-function coefficients to be solved for, the column vector v represents the known source or excitation. Method of moments (MoM) is an example of a numerical technique that yields a matrix equation of the form above. These computations can become very large and unwieldy. Therefore, the principle bottleneck is established in the form of solving the matrix equation. This is the motivation behind using compressive sensing methods for scattering analysis problems.

In order to directly solve the ZxZ matrix with a numerical method such as MoM it would take $O(Z^2)$ time. Other technique, such as MLFMM as discussed in chapter 3, can reduce computation time to O(NlogN) or O(N). Unfortunately these can also be very time consuming for very large values of N. Compressive sensing calculations using the DCT method were performed with N = 16, 32, 64 random scattering computations. By using a fixed N value, Carin's research proved that it is possible to provide a sufficiently accurate solution with a non-scaling runtime. The DCT algorithm for in situ compressive sensing is highly specific for a computational electromagnetics application. This makes it perfectly tailored for reduced runtimes of our problem of interest: RCS. [9]

Chapter 5

Fortran-Based DCT Implementation Testing

This chapter describes my initial implementation of the DCT method written in Fortran. Fortran was chosen as the implementation language because it matched the language in which the electromagnetics solver was written. I obtained a non-commercial FMM solver for the purposes of this study in order to have access to the source code and directly incorporate the compressive sensing theory. Throughout this chapter the results of both accuracy and efficiency testing of this implementation are outlined and analyzed.

5.1 Accuracy Testing

The focus of my research was not the accuracy of compressive sensing methods of computation, but rather the runtime efficiency. However, this section gives a brief proof of the accuracy which is well supported by the published literature and other implementations of similar compressive sensing methods.

5.1.1 Theoretical Accuracy

In Situ Compressive Sensing, published by Carin's team, outlines the detailed theoretical analysis of the accuracy of the DCT method. Below is a summary of that analysis.

Two basis functions were chosen for testing, a discrete cosine function and the more traditional Haar function. Calculations were run in free space and within the waveguide. Because the data of the signal is significantly sparser in the DCT basis than in the Haar basis, DCT only required 6 nonzero coefficients while the Haar transform required 30. In the in situ compressive sensing method, n receiver positions and associated frequencies are randomly selected. Therefore, the fractional error of the calculation is a function of n. Figure 1 below, from the cited paper above, shows the error for both free space and waveguide calculations of the Haar basis function. Figure 2 shows the same error calculations but for a discrete cosine function method. As the figures show, the marginal error of the discrete cosine function is much less and more consistent than the Haar function. This data proves that the DCT method is an accurate compressive sensing method in theory and out-performs the more traditional Haar basis function. [4]

Figure 2: Haar Function

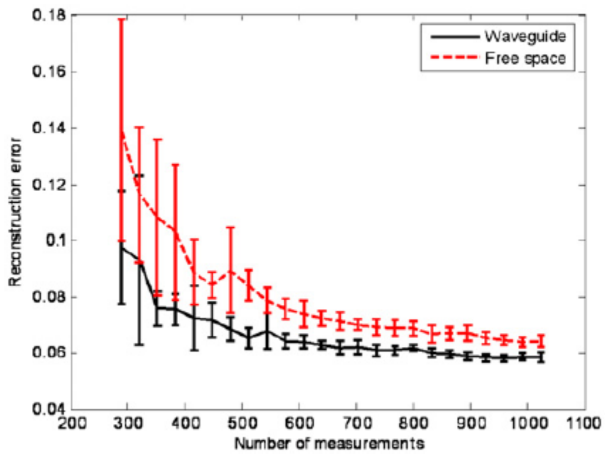
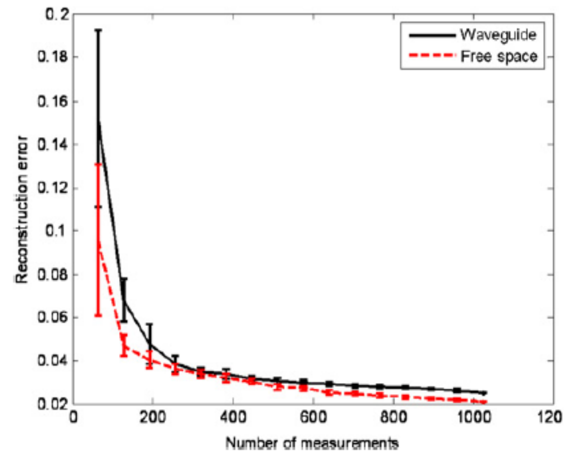


Figure 3: Discrete Cosine Function



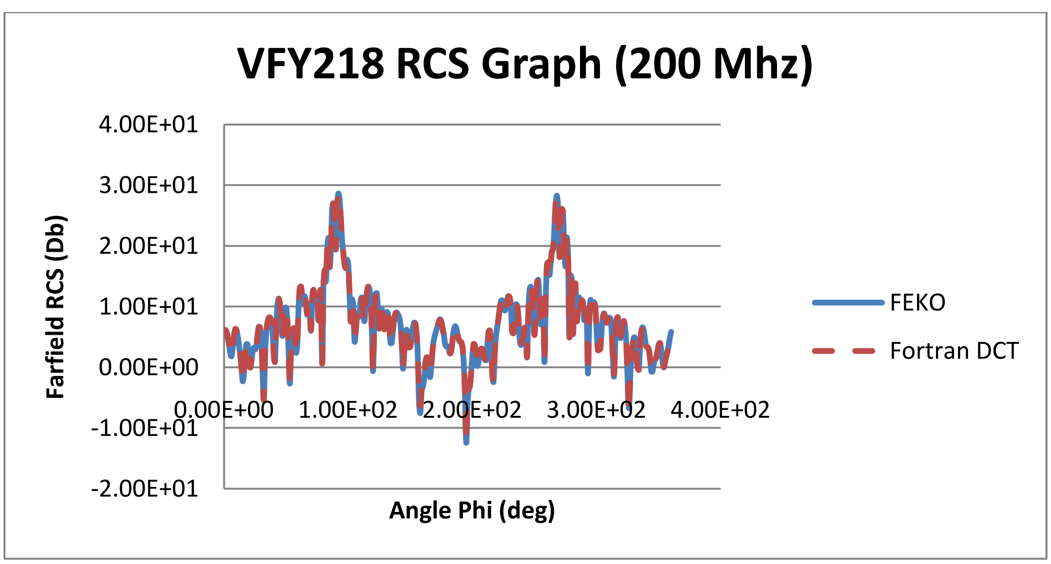
The proof above demonstrated that the DCT algorithm is an accurate CS method, but I wanted to be sure that it was also an accurate computational electromagnetics method. To test the accuracy of this new algorithm in a practical setting, I created an experiment comparing a stand-alone, DCT aided, computational electromagnetics solver and the standard commercial software known as FEKO. In order to be a valuable asset to industry, the DCT method must not only prove accurate in theory, it must remain acceptably accurate in practice.

5.1.2 Practical Accuracy

In this section I complement the theoretical proof of accuracy described above with an initial implementation of the DCT method and subsequent testing in order to prove practical accuracy. This initial implementation was coded in Fortran and worked in parallel with a basic Fast Multipole Method (FMM) solver. The Fortran software was run with the Absoft compiler. These software choices were made in order to best duplicate the initial implementation completed by Carin's team in the hopes of replicating their accuracy.

Once the code was completed as outlined by Carin's research and discussed in detail in chapter 4, I conducted tests using a practical and standard problem similar to those faced in industry. Figure 1, in section 3.1, shows the image of the mesh representation of an airplane model known as the VFY218. This model airplane is used as a stand-in for most complex EM modeling problems that involve aerial vehicle simulations. This is because the model contains a fair amount of complexity while remaining generic enough to represent many different military planes. In addition, this model contains no confidential information and is a publically-releasable file. This made the act of acquiring the model much simpler than an exact replica of a specific plane.

I ran multiple RCS simulations for the VFY218, all with frequencies ranging from 50 to 400 MHz. Identical simulation setups were run for the Fortran DCT implementation and the commercial FEKO software described in chapter 3. Figure 4 below shows the RCS graph for the VFY218 at 200 MHz. As can be observed from this figure, the Fortran implementation of the DCT method produces an RCS graph that is almost identical to the results produced by the FEKO software. Similar results were found for each calculation at each frequency.



The true interest of this research does not lie in the accuracy of the DCT computation. We have proven both theoretically and in practice that compressive sensing does yield acceptably accurate results. The topic of interest for this study is the reduction in runtime. Carin's team reveal potential for drastic time reducing capabilities with the use of compressive sensing theory. The remainder of this thesis will focus on my efforts to make that potential for faster computation a reality.

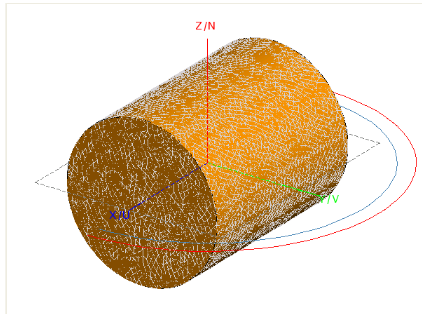
5.2 Efficiency Testing

It was already believed, and has now been confirmed, that the DCT method is an accurate approximation for EM modeling and simulations. Therefore, I focused my research on the increase of runtime efficiency that can be obtained by using compressive sensing.

5.2.1 Phase 1: Cylinder

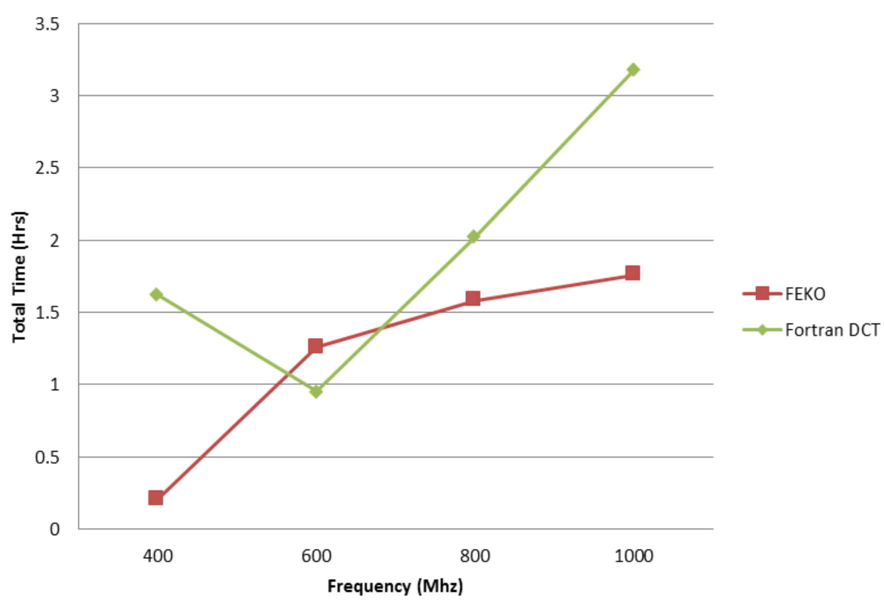
I designed two series of tests to accurately depict the efficiency of the DCT method. The first stage was based around a simple cylinder model which can be seen in Figure 5. The RCS graph was generated with both the Fortran implementation of the DCT method using an FMM electromagnetics solver. The control RCS was generated using the FEKO software solver based on the MLFMM method known for its efficiency. The simulation was run with frequencies ranging from moderate to high frequencies between 400 MHz and 1 GHz. This frequency range was chosen because it accurately represents the frequency range most often used in practical modeling and simulation problems.

Figure 5: Cylinder Mesh Model



The graph in Figure 6 below shows the trend line for the runtime of the Fortran-based DCT method and the traditional FEKO simulation. As you can see, the results are not consistent. The DCT method only outperforms the FEKO software at one frequency. At all other frequencies it is less efficient. In order to see if these results were consistent with another test case, I ran a similar RCS simulation on the VFY218 plane model.

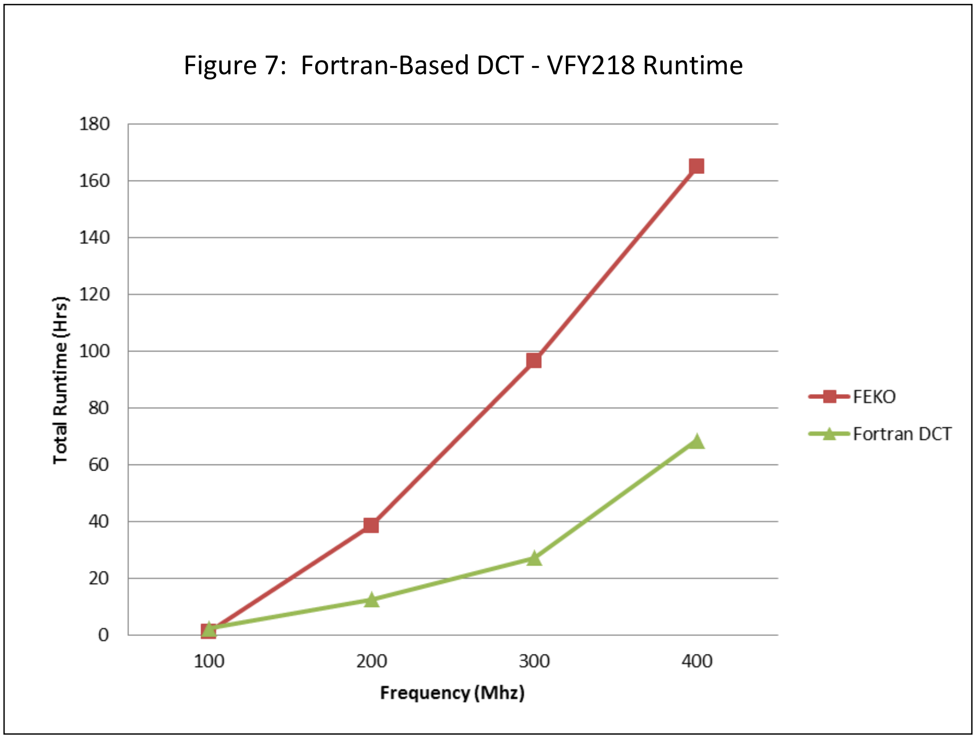
Figure 6: Fortan-Based DCT - Cylinder Runtime



5.2.2 Phase 2: VFY218

Because compressive sensing is used to simplify very complex problems, I ran an efficiency test on a model that is more complex than a simple cylinder. The VFY218 has much more complexity and should therefore benefit more from the DCT optimization. Again, RCS simulations were run on both the Fortran DCT implementation and the FEKO software. This VFY218 model has a course mesh and therefore was not compatible with the frequency range that was previously used on the cylinder model. Therefore, the frequency range for this test was reduced to values within the range of 100 to 400 MHz.

The graph in Figure 7 below shows the trend line for the runtime for the same comparison between the DCT implementation and the FEKO software, but this time with the VFY218 model. The results in this case are much more reassuring and consistent. The DCT implementation easily reduces the computation time of the RCS simulation by fifty percent for most frequencies. However, the data, which can be found in Appendix A.1, shows that at the lowest frequency of 100 MHz, the DCT implementation takes twice as long as the FEKO software.



5.3 Analysis and Conclusions

The results of the two tests described above have interesting implications. I observed that at low frequencies and with simple cases, the FEKO software runs faster than the DCT implementation. However, once the simulations became more complex with high frequencies, the DCT method was able to reduce the runtime of the traditional simulation by about fifty percent. While this is a significant reduction in time, it is not the great potential that is believed can be achieved through the use of compressive sensing.

I believe that there are several reasons as to why the simulations did not demonstrate the full capabilities of the DCT method and the compressive sensing theory. Firstly, the Fortran DCT implementation used an FMM electromagnetics solver whereas the FEKO software uses an MLFMM solver. As briefly discussed in chapter 3, MLFMM improves upon the more simplistic

FMM and is able to speed up runtimes. Secondly, the FEKO software is commercial quality and highly optimized. The FMM solver that I obtained is not as efficient and not up to par with a commercial-grade solver. FMM solvers of any kind are extremely tricky and difficult to implement. The benefit of using a commercial solver of any brand, like FEKO, is that the company specializes in implementing and optimizing such numerical analysis methods. FEKO is one of the leading creators of integral solvers and therefore has top-of-the-line efficiency quality. I believe that the lack of substantial time reduction was due to the use of FMM rather than MLFMM and using a home-grown rather than a commercial solver. [12]

These unforeseen disadvantages with my initial implementation prompted my investigation into an integration method. It would be beneficial to industry if users could harness the powers of both the DCT method and the optimized FEKO solver. Therefore, in the rest of this thesis I discuss a new DCT implementation. This implementation was created as an overlay to the FEKO solver that could work symbiotically with the optimized FEKO MLFMM method.

Chapter 6

FEKO-DCT Integration

In order to overcome the runtime challenges which were discovered in the previous chapter, I have proposed a new solution for implementing the DCT method. This method involves integrating compressive sensing with the commercial FEKO solver. This will allow the user to benefit from the runtime efficiency of both methods.

Since FEKO is a commercial solver, I am unable to gain access to the FEKO solver's source code. Therefore, in order for a successful integration, the FEKO solver will be treated as a black box and the DCT software will act as a wrapper. One of the benefits of using the FEKO platform, as opposed to any other electromagnetic software, is the way in which the FEKO suite communicates internally. The FEKO software platform is a suite of four components. The components communicate with one another using text files which can be easily manipulated and manually fed to one another as necessary. This method of communication is essential to the success of this implementation.

6.1 How FEKO Works

The following section outlines the internal communication protocol between the four independent components of the FEKO software suite. This protocol is important to understand for the implementation of the integration software. The FEKO suite's four components are as follows: editFEKO, cadFEKO, FEKO solver, and postFEKO. Each of these components has its own role in the modeling and simulation process.

There are three stages to any traditional modeling and simulation problem. First the user defines the problem. Second, a solver runs a numerical analysis method to gain a solution. Finally, the user can view those solutions as graphs and data. These three stages each correspond to a different FEKO suite component. During the first stage the user has two options for defining the problem. Either he can use the cadFEKO graphical interface or he can use the editFEKO text-based interface. The cadFEKO component allows a user to create geometric and mesh models in a CAD-like program. In addition, any other problem-defining settings can be created within this interface. Once the user is finished defining the problem, the cadFEKO program creates the necessary files to continue the simulation process. These files include a geometry file that describes the model, a mesh file that describes the mesh, and a FEKO-unique .pre file. The .pre file is essentially a text file that outlines the parameters of the problem for the second stage solver to interpret. The second component that can be used to complete the first stage of the modeling and simulation process is editFEKO. If the user is well-versed in the internal syntax of the FEKO suite, he can write a .pre file from scratch. Then, all the user needs to do is create a compatible geometry and mesh file in any other CAD software program of his choice.

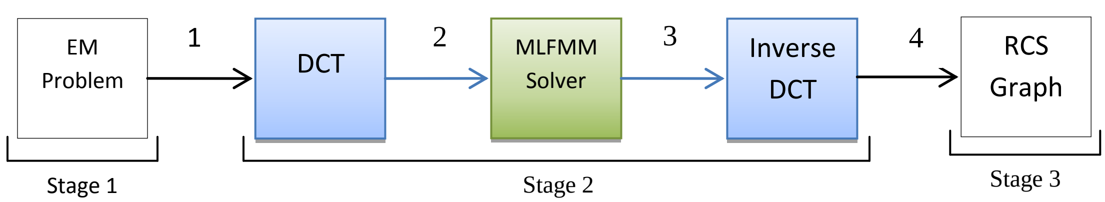
The second stage of the modeling and simulation process is the solution solving stage. This is where FEKO offers the user its greatest advantage. The FEKO solver is the component that is most desirable for runtime efficiency and it is the reason I am creating an integrated, compressive sensing software platform. The FEKO solver is the only component that has no graphical user interface. It can be triggered using the cadFEKO or editFEKO interfaces, or alternatively through the Windows or Linux command prompts. The FEKO solver interprets the information contained in the .pre file and generates a solution using any one of its CEM analysis methods which were described in chapter 3. All the solution information is written to an output text file that, similar to the .pre file, has its own unique syntactical formatting.

The final stage of the modeling and simulation process is the post processing interpretation of the solution found in the second stage. PostFEKO allows the user the view the output data stored in the output file and creates desirable graphics displaying information such as the RCS of a particular model. In order to incorporate compressive sensing into the FEKO suite, the DCT software will act as a wrapper around the FEKO solver component and manually read and write text files.

6.2 High-Level Implementation Description

This section explains how the integration software works on a high level. In Figure 8 below, the different colors of the boxes represent the different software platforms that the simulation data is being manipulated with during that stage of the process. The white boxes represent flexible interfaces. This means the user can use any software platform he prefers, including the FEKO suite components. The blue boxes represent the use of my own Java-based DCT implementation which will be explained in greater detail in the next section. The green box represents the use of the FEKO solver. The stages of a modeling and simulation problem which were described in the previous section are labeled in Figure 8. As the schematic shows, the DCT manipulation software wraps around phase 2.

Figure 8: High-Level Implementation Schematic



The transitions between boxes represent the passage of information. Black arrows represent manual transitions whereas blue arrows represent automatic transitions that are unknown to the user. Transitioning between the colored boxes is entirely automated to make the compressive sensing software as simple to use as the original FEKO solver. The information passed between each stage is contained in text files as described in the previous section. Transition 1 passes .pre files to the DCT wrapper software. Transition 2 passes those same .pre files to the MLFMM solver after the DCT manipulation has been applied to the plane wave.

Transition 3 passes output files to the back end of the wrapper software. Finally, transition 4 passes those same output files to a graphical interface after the inverse DCT manipulation has been applied to the solution data.

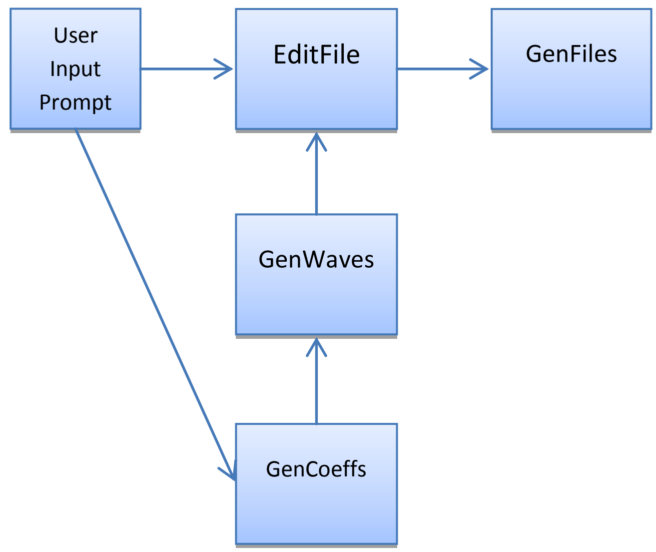
6.3 In-Depth Implementation Description

This section will delve deeper into the two stage DCT software wrapper that is portrayed in blue in Figure 8. As suggested by the schematic in Figure 8, the DCT wrapper software has two components: a front end and a back end. The front end completes all the pre-processing manipulations to the .pre text files prior to running the FEKO solver. The back end completes all the post-processing manipulations to the output text files after the solver has computed a solution.

6.3.1 Front-End

Figure 9 below displays a schematic representation for the front-end DCT software implementation. The entire DCT wrapper software was written in Java and compiled within the Eclipse environment. In the schematic below each box corresponds to a different file and the arrows represent the passage of information.

Figure 9: Front-End DCT Implementation Schematic



The purpose of the front-end files is to edit the text-based .pre file created by the user and replace the conventional plane waves with those that adhere to the DCT method as described in chapter 4. The first stage of the front-end implantation is the user input prompt. This file generates a simple, text-based input prompt that asks the user for some initial information about the electromagnetics problem. It is at this stage that the user specifies the name of the .pre file that was generate prior to running the DCT code as well as the directory in which the .pre file, mesh file, and geometry file are held. In addition, the FEKO solver needs to know how many processors are available for computation use since it has parallel processing capabilities. Finally, the DCT manipulation software needs the user to specify the number of realizations (k) and the number of incident angles (N) for the problem. These variables correspond to the variables defined in the mathematical formulas for DCT in chapter 4.

After the user has input the necessary information, the DCT software can begin processing and manipulating the .pre file. This involves the generation of new plane waves according to the DCT mathematical specifications. EditFile is the second stage of this process. This file reads and writes to the .pre file. I order to gather the new information that will be written to the new .pre file, editFile must communicate with some of the other files in the program. In Figure 9 arrows represent the passage of information from genCoeffs and genWaves to editFile.

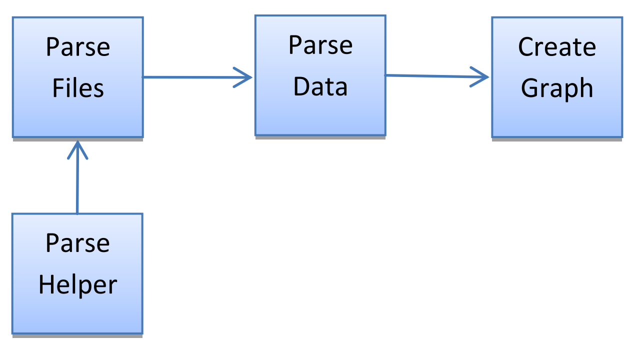
The genCoeffs file is responsible for all the mathematical computations necessary for the front-end half of the DCT algorithm. GenCoeffs generates new DCT plane wave coefficients which are calculated according to the mathematical formulas described in chapter 4. These coefficients are used by the genWaves files to generate a set of plane waves formatted to comply with the .pre text file syntax. Once these waves have been created, they are passed onto editFile to be written into the .pre file specified by the user.

The final stage of the front-end DCT software is completed in genFiles. This file creates all the necessary files needed to run the FEKO solver in the next stage of computation. Two kinds of files are created in genFiles. First, a certain number of .pre files are generated, each containing the correct DCT coefficient plane waves. The number of .pre files created is the same as the number of realizations (k) specified by the user in the input prompt. This particular program was run on a windows workstation and therefore, the second file created by genFiles is a batch file. The batch file is triggered at the end of the front-end routine and begins running the FEKO solver automatically for each .pre file that was generated.

6.3.2 Back-End

Figure 10 below displays the complementary schematic representation for the back-end DCT software implementation. Just as in the front-end schematic, each box represents a different file of the program. Again, all programs were written in Java to remain consistent with the front-end implementation.

Figure 10: Back-End DCT Implementation Schematic



The back-end software is more complex than the front-end software. These files interpret the solution data provided in the output files, reconstruct the solution through inverse discrete cosine transform, and output the RCS graph and data for the user. The first phase of the back-end implementation is to parse each of the k output files generated after the k .pre files were run through the FEKO solver. The output information that is relevant for our calculation is held within the last hundred lines of the output file. Because each output file is hundreds of thousands of lines long, it is time efficient to parse the file from the bottom up. The parseHelper file enables this capability. ParseHelper aids parseFiles by parsing the output file line by line starting at the end of the file and working towards the beginning. After parseHelper sends parseFiles each line of the output file, parseFiles extracts the solution data pertaining to the RCS calculation for the problem as well as important computation information such as peak memory usage and total runtime.

The important solution data held in the output file is the e-theta value. The e-theta values represent the scattered electric field strength in the far field at angle interval values surrounding the model. ParseData takes each line of the output file that contains the e-theta value and parses out only the numerical characters of interest. After this e-theta numerical data has been extracted, the real and imaginary parts are computed and placed in a kxN matrix where k refers to the number of realizations and N refers to the number of angles. These values were specified by the user in the front-end. This process is repeated for each output file and the inverse DCT function is applied.

The last phase of the back-end implementation is the computation of the RCS graph given the parsed and reconstructed output data. The createGraph file creates an RCS graph and displays both the graph and the RCS data for the user's use in a post-processing interface of choice later. In addition to the RCS data, the outputted graph displays the total runtime of the FEKO solver and the peak memory usage during runtime.

Chapter 7

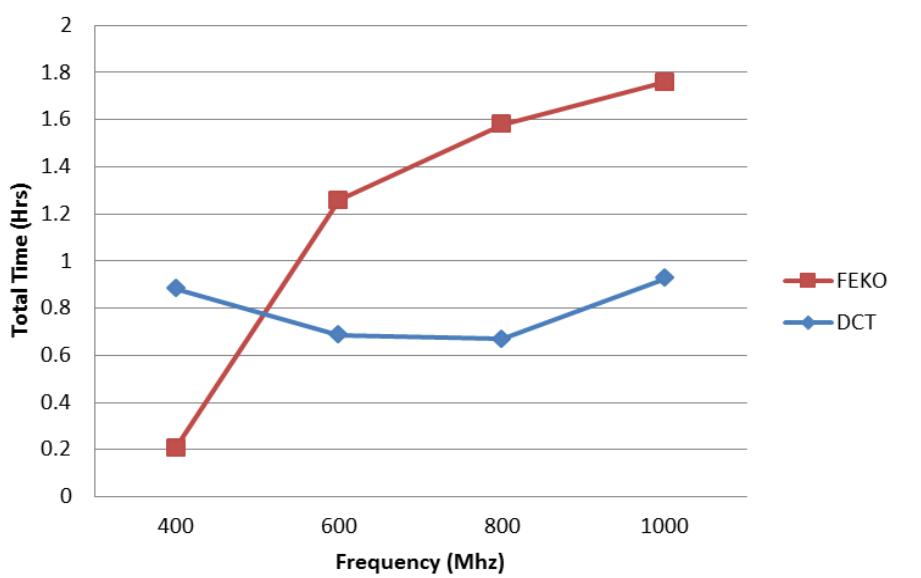
Runtime Efficiency of FEKO-DCT Integration

The previous chapter described a new implementation of the DCT algorithm. Compressive sensing was integrated into the FEKO solver in order to harness the benefits of the commercially optimized MLFMM method. In order to test that my new implementation scheme was runtime-efficient I performed two series of test simulations. The tests performed on the new implementation are identical to those performed for the runtime testing of my initial Fortran implementation which can be reviewed in section 5.2.

7.1 Phase 1: Cylinder

For the first phase of the DCT integration testing, a simple simulation case was conducted. As previously described in section 5.2.1, a simple cylinder was modeled and the RCS was calculated using both the conventional FEKO software platform and the new implementation of DCT using the FEKO solver component. Figure 11 shows the trend line for simulations run at frequencies within the range of 400 MHz and 1 GHz at intervals of 200 MHz. These frequencies represent a common range of problems often encountered in practical modeling and simulation scenarios.

Figure 11: FEKO-DCT Integration - Cylinder Runtime



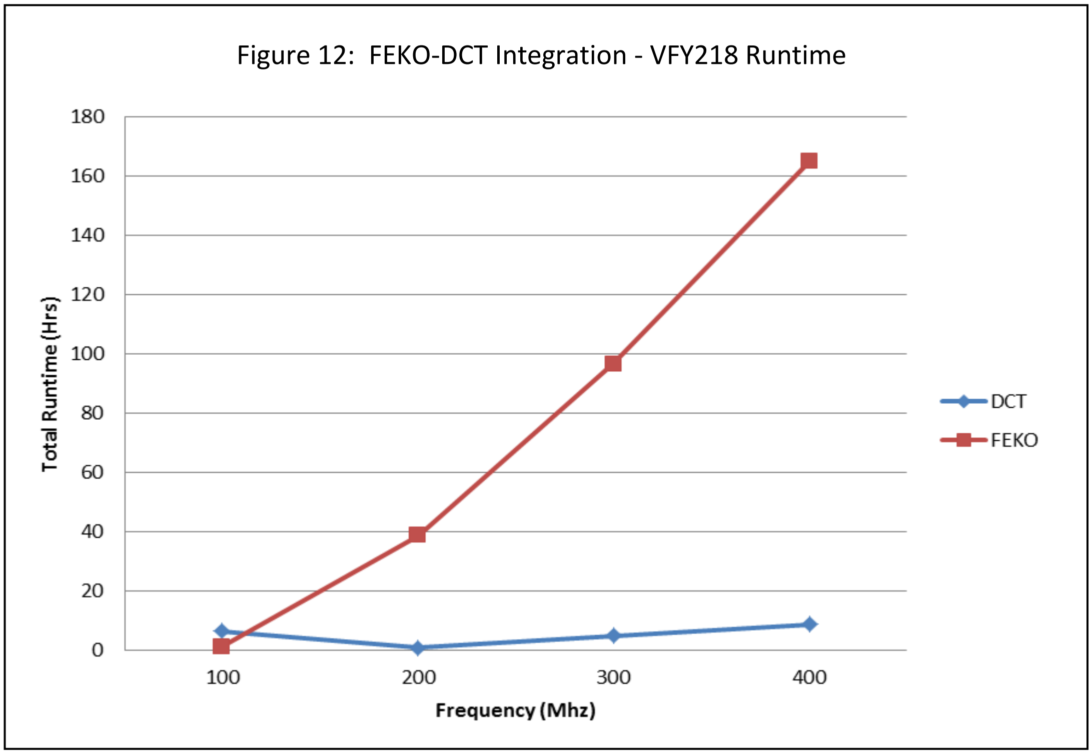
As the figure above shows, this new implementation of the DCT method had much more consistent results than the initial Fortran implementation. At the highest frequency of 1 GHz, the DCT implementation was able to reduce the runtime of the MLFMM solver from one hundred and five minutes to fifty-five minutes. At the three higher frequencies we encounter consistent results showing that the DCT method can reduce the runtime of the MLFMM solver by a factor of two. A complete summary of all the data from this table can be found in Appendix A.2. This data shows great improvement over the first implementation, but unfortunately a factor of two in speed-up time is a much smaller reduction than suggested by the compressive sensing literature.

While the DCT method was able to decrease the runtime of the simulation at the three highest frequencies, we find inconsistencies with the method's runtime when it comes to the lowest frequency. At 400 MHz the DCT method actually increased the runtime from twelve and a half minutes to almost fifty-three minutes. Despite the increased consistency of our data for the new implementation, it appears that the DCT method does not increase the efficiency of the MLFMM solver at low frequencies. Section 7.3 will discuss the possible reasons for inconsistencies to exist.

7.2 Phase 2: VFY218

Because the first phase of testing results suggested that the second implementation of the DCT method was more successful than the first, I believed that the second phase would result in even greater success. The efficiency testing of the first implementation proved that the DCT method was most effective at reducing computation time for simulations of complex models at high frequencies. Therefore, the second phase of the DCT integration was conducted with the same VFY218 model used in the previous implementation's second phase efficiency testing.

Figure 12 below displays the data results for the runtime of RCS computations of the VFY218 model. These tests were run with frequencies in the range of 100 MHz to 400 MHz at intervals of 100 MHz. Again, this test was identical to the one conducted for the efficiency tests of the first DCT implementation as outlined in section 5.2.2.



The data results above show staggering results from the new implementation of the DCT method. The graph shows that the runtime of the simulation increases linearly with frequency using only the FEKO software suite. However, when the DCT method is incorporated, the runtime remains fairly constant for all frequencies within the range of this test. While the runtime does increase slightly between 200 and 400 MHz, the increase is minimal compared to the runtime increase seen with the FEKO suite alone. As described in section 4.3, having a constant value for N can keep the runtime of EM problems relatively constant as complexity increases. I am reducing part of the EM problem from a linear runtime to a constant runtime. The other components of the EM simulation process, such as pre-processing, are the causes of the small increases in runtime as frequency increases. A complete summary of all the data found in the chart above can be found in Appendix A.2.

As seen in Figure 11 above, the FEKO suite takes almost seven days to complete a VFY218 simulation at 400 MHz. Using my DCT integration implementation, the runtime for the exact same simulation was reduced to a mere eight hours. Similarly, at 300 MHz, the FEKO suite spent over four days computing a solution whereas my DCT integration implementation took less than five hours. These test cases show a consistent runtime reduction of almost two orders of magnitude for the three highest frequencies in this test. These results are even greater than suggested in the compressive sensing literature. These tests prove that the DCT compressive sensing method is a successful and accurate way of drastically reducing the computation runtime of electromagnetic simulations.

However, despite the staggering runtime reductions for the three highest frequencies in the graph above, these tests again show an increase in runtime for the lowest frequency when using the DCT method. The next section discusses the possible reasons why such results occur.

7.3 Analysis and Conclusions

The test results shown above prove that my new compressive sensing implementation, which integrates the DCT method with the FEKO MLFMM solver, can drastically reduce computation runtime for electromagnetic problems. These results are very promising for the future of compressive sensing and its applications in computational electromagnetics. Reducing computation time by two orders of magnitude is greater than could have been imagined from simple theory. However, despite these grand results, it is evident from the data that compressive sensing has its drawbacks for certain classes of problems.

The tests described in the previous two sections show an inconsistency in the runtime efficiency at extremely low frequencies. Traditionally, CEM problems have a longer runtime if they are run at higher frequencies because the problem becomes more complex. The MLFMM solver, which is used in the FEKO suite, is highly optimized for low frequency cases since they are so simplistic. After viewing these results, it was clear that my integration implementation had a drawback that resulted in greater runtime for these simpler cases.

The FEKO solver takes a certain amount of time at the beginning of any simulation to set-up the problem. This involves processing the mesh model, solving a pre-conditioner, and computing various other values which can be stored for later use. After the initial set-up is complete the iterative solver begins the process of converging upon a solution for each angle of the farfield calculation. The number of angles can be very large and contain redundant information. The large amount of data that needs to be processed for such calculations is the reason why compressive sensing theory was developed for computational electromagnetics initially.

In the case of the DCT algorithm, only k iterative solutions need to converge in order for a farfield solution to be calculated. As was described in chapters 4 and 6, k realizations become k separate .pre files. Because I did not have access to the FEKO solver's source code, I was forced to manually feed the solver each of the k .pre files individually rather than run them concurrently as is done when the entire FEKO suite is utilized. This meant that I had to start the FEKO solver k separate times and therefore repeat the set-up process k times when the traditional FEKO method only needed to compute the set-up data once. This redundant computation at set-up time makes the simpler simulations less efficient when run using the DCT integration implementation.

The interesting aspect of the DCT integration implementation is that the savings it brings to the iterative solver aspect of the MLFMM method soon outweighs the additional overhead set-up costs. It becomes apparent that the most time-consuming aspect of any fairly complex simulations is the iterative process of converging upon a solution for each angle. This means that the DCT method will almost always reduce the runtime cost despite the redundancy of the set-up calculations.

Chapter 8

Future Advancements in Compressive Sensing

My second DCT implementation which incorporated the FEKO MLFMM solver was extremely successful in reducing the runtime costs of electromagnetic modeling problems. However, there are many improvements that can be made to this software in the future in order to further decrease the runtime and make it industry-friendly.

As the software stands at the moment it has a non-inconsequential flaw which makes it less robust than required for widespread use within the EM modeling industry. Any low frequencies or for simple models, my implementation that incorporates the DCT method will actually have a greater runtime than simply using the conventional FEKO suite. This means that my program has inconsistent results and will not be an acceptable industry standard. There are two possible solutions to this problem that could result in a more successful, consistent and complete software platform.

First, I want to investigate the development of my own MLFMM solver. This numerical analysis method appears to be the solver-of-choice for RCS computation problems and is extremely efficient at finding a solution. The development of my own MLFMM solver would enable the direct integration of the DCT algorithm rather than building a wrapper function that resulted in unnecessary redundancy as described in the previous chapter. This MLFMM solver would have to be equally as optimized as the FEKO solver. While creating my own MLFMM solver is an attractive solution in theory, I have many reservations about attempting to create such a complex piece of software. The MLFMM method is extremely complex and it could take years to develop something that works even half as well as the FEKO MLFMM solver. In addition, our users would then be confined to using the MLFMM method when other computational electromagnetic problems may benefit more from the Method of Moments of Fast Multipole Method techniques. My overall goal is to decrease runtime but not at the expense of computation flexibility. Therefore, I do not believe that creating my own MLFMM solver is the best solution for creating a consistent and successful compressive sensing-aided CEM solver.

The second possible solution to improving my DCT integration software is to make my software more intelligent. Rather than always using the DCT method, an intelligent implementation would choose a solver method best suited to the problem at hand. This software would choose the solver method for the problem by assessing different complexity aspects of the model. These aspects would include the frequency, the size and complexity of the geometry model, the number of elements in the mesh model, and the complexity of the user-requested solution. The software could then choose between any of the FEKO solver types and then choose whether or not to incorporate the DCT method as a wrapper to that particular solver. This intelligent software could be extremely efficient and consistent in its runtime and accuracy: the beginnings of a new industry standard.

FEKO and most other commercial software companies are now venturing into the domain of graphical processing units (GPUs) for increased runtime efficiency for their numerical analysis methods. GPUs offer a completely different architecture than traditional CPUs upon which to build a computational electromagnetics solver, such as MLFMM. GPUs are inherently structured with enormous parallel-processing capabilities on a single machine. The future of electromagnetic computation lies in the potential of GPU software. I believe that the best future advancement for runtime efficiency is to investigate implementation of compressive sensing in a GPU-specific language. In order for this technology to be effective, a commercial-quality CEM solver must be implemented first. This technology is currently being developed by companies such as FEKO. [7] Until this new technology is well accepted in the industry, compressive sensing integration will remain on-hold.

As discussed in this section, there remain numerous future advancements in the field of compressive sensing as applied to computational electromagnetic simulations. There is great potential to improve upon the software developed as a result of my current research. In addition, new research in the field of GPU technology holds a myriad of possibilities for simulation runtime reduction. My research has simply scratched the surface of the great potential that lies in compressive sensing and the possibilities for better runtime efficiency.

Chapter 9

Contributions

The theoretical benefits of compressive sensing algorithms have been understood and well-researched for decades. Unfortunately, that theory has yet to be incorporated into practice for problems in the field of computational electromagnetics. Throughout the course of my research I have made two important contributions to the theoretical and practical study of compressive sensing. Firstly, I have identified and implemented a specific compressive sensing algorithm well-suited for application in computational electromagnetics. This implementation proved that the DCT method was both a practically valid and relatively efficient method of compressive sensing. More importantly, my research proved that this method was also both a valid and effective method of electromagnetic modeling.

The second and most significant contribution of my research is the integration of the DCT compressive sensing method with the commercial FEKO solver software. This integration proved beyond a doubt that the DCT algorithm can change the way current electromagnetic modeling systems work. Reducing simulation runtime by two orders of magnitude dwarfed all predictions about the benefits of compressive sensing. Through my creative integration method, I was able to show how effective compressive sensing can be in a practical, industry setting. I also proved that integration creates no additional complexity cost since my software was as simple for the user as the traditional FEKO suite components. The research in this thesis is simply the beginning of my journey to explore the possibilities of compressive sensing and to change the way industry computation systems are created. These two major contributions highlight the great potential that lies ahead.

Appendix A

A.1: Runtime Data for Fortran-Based DCT Implementation

Computation Method	Frequency (MHz)	Runtime (Hrs)
FEKO	400	0.208
FEKO	600	1.257
FEKO	800	1.58
FEKO	1000	1.758
Fortran-Based DCT	400	1.617
Fortran-Based DCT	600	0.952
Fortran-Based DCT	800	2.019
Fortran-Based DCT	1000	3.176
FEKO-DCT	400	0.883
FEKO-DCT	600	0.687
FEKO-DCT	800	0.67
FEKO-DCT	1000	0.927

A.2: Runtime Data for FEKO-DCT Implementation

Computation Method	Frequency (MHz)	Runtime (Hrs)
FEKO	100	1.187
FEKO	200	38.537
FEKO	300	96.462
FEKO	400	164.801
Fortran-Based DCT	100	2.272
Fortran-Based DCT	200	12.629
Fortran-Based DCT	300	27.147
Fortran-Based DCT	400	68.56
FEKO-DCT	100	6.31
FEKO-DCT	200	0.731
FEKO-DCT	300	4.856
FEKO-DCT	400	8.664

Bibliography

[1] Baraniuk, Richard G., Volkan Cevher, Marco F. Duarte, and Chinmay Hegde. "Model Based Compressive Sensing." Rice University. Web. <http://www.ece.rice.edu/~vc3/model_based_CS.pdf>.

[2] Baron, Dror, Marco F. Duarte, Michael B. Wakin, Shriram Sarvotham, and Richard G. Baraniuk. "Distributed Compressive Sensing." *Distributed Compressive Sensing*. Israeli Institute of Technology, 2008. Web. <http://arxiv.org/PS_cache/arxiv/pdf/0901/0901.3403v1.pdf>.

[3] Candes, E.J., J. Romberg, and T. Tao. "Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information." *IEEE Transactions on Information Theory* 52.2 (2006): 489-509. Print.

[4] Carin, Lawrence, Dehong Liu, and Bin Guo. "In Situ Compressive Sensing." *IOP Science: Inverse Problems*. IOP Science, 2008. Web. <http://iopscience.iop.org/0266-5611/24/1/015023/pdf/0266-5611_24_1_015023.pdf>.

[5] Carin, Lawrence, Dehong Liu, Wenbin Lin, and Bin Guo. "Compressive Sensing for Multi-static Scattering Analysis." *Journal of Computational Physics* 228.9 (2009): 3464-477. Print.

[6] Donoho, D.L. "Compressed Sensing." *IEEE Transactions on Information Theory* 52.4 (2006): 1289-306. Print.

[7] "GPU Acceleration of Electromagnetic Solutions in FEKO." *FEKO - EM Simulation Software*. FEKO, Dec. 2009. Web. <http://www.feko.info/about-us/News/gpu-acceleration-of-electromagnetic-solutions-in-feko>.

[8] Ji, Shihao, Ya Xue, and Lawrence Carin. "Bayesian Compressive Sensing." *IEEE Transactions on Signal Processing* 56.6 (2008): 2346-356. Print.

[9] Mesecher, Dave, Larry Carin, Ron Pirich, and Ivan Kadar. "Exploiting Signal Sparseness for Reduced-Rate Sampling." IEEE, May 2009. Web. <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05031567>.

[10] "Multi-Level Fast Multipole Method." *FEKO - EM Simulation Software*. FEKO. Web. <http://www.feko.info/product-detail/numerical_methods/mlfmm>.

[11] "Numerical Methods — FEKO." *FEKO - EM Simulation Software*. Web. <http://www.feko.info/product-detail/numerical_methods/numerical-methods>.

[12] "Overview of FEKO — FEKO." *FEKO - EM Simulation Software*. Web. <http://www.feko.info/product-detail/overview-of-feko/overview-of-feko>.

[13] Rao, S. M., and N. Balakrishnan. "Computational Electromagnetics – A Review." *Indian Institute of Science*. Web. <http://www.iisc.ernet.in/~currsci/nov25/articles24.htm>.