2014

Bio-Inspired Optimization of Ultra-Wideband Patch Antennas Using Graphics Processing Unit Acceleration

Brian Vyhnalek
Cleveland State University

BIO-INSPIRED OPTIMIZATION OF ULTRA-WIDEBAND PATCH ANTENNAS USING GRAPHICS PROCESSING UNIT ACCELERATION

BRIAN VYHNALEK

Bachelor of Science in Physics

Bachelor of Science in Mathematics

Cleveland State University

December, 2009

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

April, 2014

We hereby approve the thesis for

Brian Vyhnalek

Candidate for the Master of Science in Electrical Engineering degree

This thesis has been approved for the

Department of

Electrical and Computer Engineering

and the

CLEVELAND STATE UNIVERSITY'S

College of Graduate Studies by

Dr. Dan Simon, Chairperson

Department & Date

Dr. Fuqin Xiong

Department & Date

Dr. Miron Kaufman

Department & Date

April 2, 2014

Date of Defense

To Liz and Alexander

ACKNOWLEDGEMENTS

I would like to thank my committee members: first Dr. Dan Simon for his willingness and guidance as my supervisor, Dr. Fuqin Xiong for his encouragement, and Dr. Miron Kaufman for his support over the years. In addition I would like to thank everyone at the NASA Glenn Research Center for the opportunities. I would also like to thank my wife and son for their support and understanding.

BIO-INSPIRED OPTIMIZATION OF ULTRA-WIDEBAND PATCH ANTENNAS USING GRAPHICS PROCESSING UNIT ACCELERATION

BRIAN VYHNALEK

ABSTRACT

Ultra-wideband (UWB) wireless systems have recently gained considerable attention as effective communications platforms with the properties of low power and high data rates. Applications of UWB such as wireless USB put size constraints on the antenna, however, which can be very difficult to meet using typical narrow band antenna designs. The aim of this thesis is to show how bio-inspired evolutionary optimization algorithms, in particular genetic algorithm (GA), particle swarm optimization (PSO) and biogeography-based optimization (BBO) can produce novel UWB planar patch antenna designs that meet a size constraint of a 10 mm $\times$ 10 mm patch. Each potential antenna design is evaluated with the finite difference time domain (FDTD) technique, which is accurate but time-consuming. Another aspect of this thesis is the modification of FDTD to run on a graphics processing unit (GPU) to obtain nearly a 20$\times$ speedup. With the combination of GA, PSO, BBO and GPU-accelerated FDTD, three novel antenna designs are produced that meet the size and bandwidth requirements applicable to UWB wireless USB systems.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation

Wireless technology has become an essential part of communications over the past century, especially in the last twenty years with the rapid growth of mobile telephones and networks. More recently there has been a tremendous development of networked computers in a variety of forms such as laptops, tablets, ebook readers, netbooks, and especially mobile smartphones. Along with the development of such devices, there have been advancements in network technology with the development of wireless sensor networks [1], personal area networks [2], body area networks [3], and ambient networks in general [4].

With this evolution there has been an enormous growth in both device connectivity and pervasive computing [5]. Wi-Fi, WiMAX, and other WPAN technologies such as Bluetooth, RFID, Z-Wave, Ultra-Wideband (UWB), and ZigBee have allowed for increasingly seamless operation and connectivity between devices in close proximity [6]. Furthermore, WPAN technology allows not only for device interconnectivity, but also for connectivity to higher level networks and the internet [7].

Related to the advancements in wireless personal area networking, there has also been development in wireless device peripherals. In the same way that USB technology has offered fast, interoperable and secure connections,

wireless USB (WUSB), based on UWB protocols, allows for the same features, but with the convenience and ease of use of wireless [8]. Computer peripherals such as wireless keyboards and mice have been on the market for some time, however the development of WUSB and advancements in WPAN technology have allowed for the development of wireless computer monitors, printers, hard disk drives, game controllers, printing of digital pictures from cameras, and efficient transfer of data from digital camcorders [9].

As wireless communication technology becomes more pervasive, the need for systems optimized for high data rates becomes increasingly more important. According to the Shannon-Hartley theorem [10], the maximum possible data rate, or capacity, for an idealized band-limited channel perturbed by additive white Gaussian noise (AWGN) is

$$C = B \log_2 \left(1 + \frac{S}{N} \right) \tag{1.1}$$

where C is the transmission data rate (capacity), B is the channel bandwidth, and S/N is the signal power to noise power ratio.

From (1.1) it can be understood that the channel capacity is related to both the bandwidth and transmission power, and that increasing either one will increase the maximum data rate. However, increasing power is costly, especially for wireless devices that are typically dependent upon battery power. Additionally, due to the linear relationship to bandwidth and logarithmic relationship to power, doubling the channel capacity would require doubling the bandwidth, but a four times increase in power. Therefore, the most efficient solution is to increase the available bandwidth. Unfortunately, increasing bandwidth may not always be possible due to the careful regulation of the radio frequency spectrum, which seeks to minimize the potential interference between transmissions from adjacent sections of the spectrum [11].

Ultra-wideband (UWB) technology has the potential to meet the requirements for low power and high data rates by spreading information over a very large bandwidth, 3.1-10.6 GHz. This is feasible due to the unique nature of UWB compared with traditional transmission systems. Typically associated with impulse radio, or high-speed spread-spectrum radio, UWB operates fundamentally different than traditional narrow band tranmission systems that transmit information by varying the power, frequency or phase of a signal [12]. Instead, UWB transmission systems operate at power levels at essentially the noise floor – or below, using low power ultra-short information bearing pulses [13]. In this way, UWB systems can operate simultaneously with other RF communications systems without interference.

Due to the short duration of the transmission pulses, UWB systems are able to achieve extremely high data rates, on the order of several Gbps at distances of a few meters, far exceeding the levels of comparable technology such as Bluetooth [14]. Besides the advantage of high data rates and low power, the operation of UWB at noise floor levels provides better security, lower potential radio frequency health hazards, and coexistence with narrowband systems [15]. In addition, single band direct sequence UWB signals do not suffer from multipath (Rayleigh) fading degradations that are seen in traditional narrowband signals [12].

One of the most critical issues in the design of a UWB system is the antenna component. Unlike typical narrowband antennas, in which the antenna is tuned to resonate at a specific frequency over a fractional bandwidth of less than a few percent, a UWB antenna must resonate well over the entire 3.1-10.6 GHz band – a fractional bandwidth of over 100 percent. Although broadband antennas have been in use for decades, even as early as the nineteenth century, current development has focused on smaller, planar antennas that can easily be integrated onto printed circuit boards [16].

Considerable attention has been given to planar monopole microstrip patch antennas. The main advantages of these antennas are their low profile, ease of fabrication, and simplicity of integration. However, the main disadvantage, in particular for UWB applications, is the relatively narrow impedance bandwidth. Several techniques for improving the impedance bandwidth have been reported, such as parasitic elements [17], beveling [18], multiple feeds [19], shorting pins [20], and semi-circular bases [21]. Typically these designs have resulted in antennas that are too large or unsuitable for circuit board integration. Several other studies have shown suitable designs, typically by cutting notches and adding slots in selective ways [22, 23, 24, 25, 26, 27, 28, 29].

The use of genetic algorithm (GA) optimization, and other related bio-inspired heuristic optimizers, such as particle swarm optimization (PSO), have generated some very novel antenna designs. In particular, the idea of a "pixelated" rectangular patch antenna in which the rectangular region is divided into small squares of either metalization or air, whose geometry is then optimized by a GA or PSO, has been successful in the design of antennas for specific frequencies [30], multi-resonances [31], bandwidth broadening [32], as well as UWB applications [33]. However, the GA optimized UWB antennas thus far have dimensions that are still too large for use in many of the intended applications of UWB such as wireless USB.

Optimization using methods such as GA have a distinct advantage in that they are relatively easy to configure for single objectives, such as bandwidth widening only, or multiple objectives such as bandwidth widening plus radiation pattern symmetry. By properly defining the objective function, a designer can generate solutions that are potentially globally optimal, as in the single objective case, or possibly globally optimal along a parameterized curve or surface, as in the multiobjective case. In this way, optimal designs can be determined without costly redesign and testing.

Although bio-inspired optimization in antenna design has been very successful, one of the major drawbacks is the amount of time needed to complete an optimization run. This is due to the fact that designs must be evaluated by full-wave electromagnetic simulation, such as the finite element method (FEM), method of moments (MoM), or the finite difference time domain (FDTD) method, which typically require several minutes for completion. However, when thousands of function calls are needed for an optimization run, the time needed can become prohibitive, on the order of several weeks [34]. Thus it is necessary for the success of the technique that suitable ways be determined to speed up evaluation time.

1.2 Organization

Chapter 2 provides some history and background on ultra-wideband technology, in particular how UWB technology began with the radio pioneers of the late nineteenth and early twentieth century. In addition, some of the applications of UWB, such as radar, through-wall imaging and sensor networks are described. Next a brief overview of microstrip patch antennas is given, along with some of the ways in which bandwidth has been increased. Finally chapter 2 concludes with a description and examples of UWB antennas found in the literature, as well as an explanation of the "pixelation" method for antenna design.

Next, Chapter 3 discusses the algorithmic formulation of the finite difference time-domain method for computational electromagetics, and how this is applied in particular to microstrip antennas. It begins with Maxwell's equations, and follows with how the equations are discretized on a uniform lattice according to the Yee algorithm. Next, a brief discussion of boundary conditions is given, in particular the uniaxial perfectly matched layer (UPML). After the section on boundary conditions, source conditions are described, techniques in which an electromagnetic excitation is simulated

and energy is added into the computational domain. Lastly, an example of an application of the method is detailed, showing how FDTD can be used to obtain information on antenna parameters, specifically impedance bandwidth, and showing how well the FDTD code developed for this study compares with a commercial MoM solver.

Chapter 4 gives an basic overview of generalized computing on graphic processor units (GPUs), particularly how the FDTD algorithm can be implemented using OpenCL (Open Compute Language). Some information is given regarding host programming and fundamental data structures, data transfer and partitioning, kernel programming, and device memory. Additionally, the algorithmic structure of the FDTD method implemented as an OpenCL kernel is outlined, showing how partitioning data correctly between GPU global memory and local memory can result in enormous speedups.

Chapter 5 introduces the concepts of bio-inspired optimization, how it differs from classical, calculus-based optimization, as well as other nonlinear methods, and outlines genetic algorithms (GA), particle-swarm optimization (PSO), and biogeography-based optimization (BBO). Beginning with a description of the binary genetic algorithm, the ideas of fitness, selection, and recombination are elaborated. Next, PSO is described, and then its binary variant is detailed, since unlike a GA or BBO, PSO is not trivially implemented in binary form. Following PSO, BBO is outlined, discussing the concpets of islands, immigration and emigration, and how the interplay of these lead to data exchange and generate optimal solutions. Finally, there is an example of the application of the GA, PSO, and BBO to a benchmark function.

Chapter 6 details how the GPU-accelerated FDTD and optimization methods were combined for the specific application of designing a UWB antenna. The best results obtained by each optimization method are pre-

sented. These are the main contributions of this study. Specifically, the application of BBO to a UWB antenna optimiation problem. The combination of GPU–accelerated FDTD method and bio-inspired optimization, which can reduce the time for antenna optimization by a large factor making the method more feasible. Lastly, three UWB patch antennas are designed with the patch constrained to be 10 mm $\times$ 10 mm, such that can be fit into a USB dongle, for example.

Chapter 7 concludes the thesis and discusses some possible extensions to the study. In particular, the most immediate extension would be a multi-objective optimization, and the calculation of Pareto-optimal solutions. There are several variables that are part of a complete UWB antenna design, with bandwidth being one of the most important, but also gain stability over the 3.1-10.6 GHz range, as well as phase linearity, and polarization, for example. Additional possibilites regarding antenna miniturization are mentioned, and also the implications of multi GPU-accelerated optimization using GPU clustering.

CHAPTER II

OVERVIEW OF ULTRA-WIDEBAND TECHNOLOGY

2.1 History

Ultra-wideband technology dates back to the original radio pioneers of the late nineteenth and early twentieth century. In particular, the short, pulsed transmissions of spark-gap radio were UWB. However, the usage of spark-gap radio was mainly due to the technological limitations of transmitters and receivers of the time, as radio pioneers had already considered the concept of narrowband, multi-channel systems [16]. As technology advanced, and narrowband communications became feasible, spark-gap UWB systems fell out of favor. Eventually, due to transmission interference and the lack of regulatory ability, by 1924 spark-gap radio was outlawed [35].

While narrowband radio technology evolved and flourished throughout the remainder of the twentieth century, patents for UWB related technology were being granted, most of the fundamental theoretical constructs of UWB signals and systems were developed, and several academic research programs were founded to specialize in UWB technology [36]. Much of the theoretical progress of UWB occurred in the 1960s and 1970s, largely due to the need to characterize the impulse response of microwave networks in the time-domain. By using short, pulsed signals, system responses over a large frequency range could be analyzed simultaneously using Fourier techniques, in contrast to measuring the system response to each individual

frequency [37]. Figure 2.1 shows an example of how a signal that is narrow in time is wide in frequency.

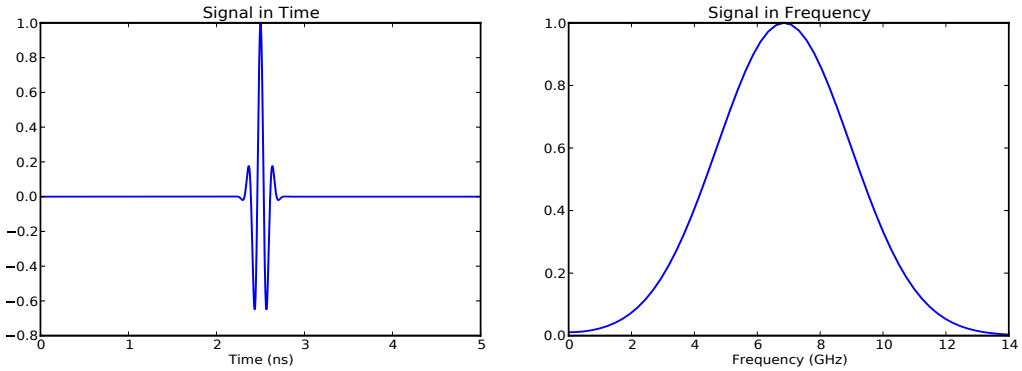


Figure 2.1: Pulsed signal in the time-domain and frequency-domain

Although the theoretical work indicated the utility of impulse techniques, it was not until advances in testing and measurement tools that UWB became practically feasible. Particularly important was the development in 1962 of the sampling oscilloscope, which allowed the direct measurement of microwave network impulse responses. Other UWB-enabling technologies developed during the 1960s and 70s were sample-and-hold receivers, useful for UWB signal averaging, short-pulsed radar, threshold receivers, pulse train generators and modulators, switching pulse train generators, detection receivers, wideband antennas, and the Hewlett-Packard network analyzer [38].

Much of the work in the 1980s was practical development, typically military radar applications, and the 1990s saw applications to commercial wireless communications systems as component technology improved and costs were reduced [12]. Although UWB had moved into the commercial sector by this time, there was not wide acceptance. Lack of an industry standard and implementation difficulty certainly contributed, but the most important reason for the stagnation of UWB was the lack of a specific FCC frequency allocation [38].

In February 2002, this situation changed. The FCC issued its First Report and Order, Revision 15, defining ultra-wideband, and allocating the 3.1 GHz to 10.6 GHz frequency band [39]. In this way, UWB radio could be operated simultaneously with existing RF systems, since one of the primary advantages of UWB radio is its ability to operate at extremely low power, essentially at the noise floor [13]. Since the FCC allocation of the 3.1-10.6 GHz frequency band for unlicensed use, a large research effort has been made to improve and commercialize UWB technology.

2.2 Ultra-Wideband Applications

There are several important applications of UWB technology. Historically, UWB systems have been utilized by the military for low probability of detection (LPD) radar. Unlike conventional radar, UWB radar demonstrates superior accuracy and range capabilities because of the shorter time durations of the pulses, and resulting shorter spatial lengths. Due to the increased resolution, target recognition is enhanced. Furthermore, UWB radar is relatively unaffected by typical atmospheric propagation elements such as rain, fog, or snow [40, 41], and is more secure becuase of the spectral broadening. Another advantage of UWB radar is that it can detect slowly moving or stationary objects [41].

Commercially, the most important and prevalent usage of UWB is as a physical layer for wireless personal area networks (WPANs), also known as in-home networks [13]. UWB technology is known for extremely high data rates over a short distance, while operating at power levels at or below the noise floor. These features are well suited for wireless personal area networks, which interconnect low power, battery operated, typically mobile devices, operating over distances of a few meters. Wireless peripherals, like computor monitors, USB, hard drives, scanners, etc., are all possible due to the very large data rates associated with UWB, virtually eliminating office

and household clutter due to wires. UWB implementation can provide data rates high enough for wireless transmission of high definition video [42, 43].

Similarly, UWB has found applications for wireless sensor networks (WSN) [44, 13]. Wireless sensor networks are arrangements of autonomous sensors grouped into nodes for the purposes of monitoring a large variety of environmental, medical, and industrial phenomena, such as temperature, pressure, air pollution content, heart rate, blood pressure, machine health, etc., [45, 46]. UWB technology is uniquely advantageous to such networks mainly due to low power consumption, since the sensors of a WSN typically are either battery powered, or powered through energy harvesting. Also, the large bandwidth and corresponding high transmission capacity enables the transfer of large amounts of data from diverse sensor architectures. Finally, the advantages of UWB in radar applications are also useful in the context of WSN applied to geolocation and remote sensing [47].

Other radar applications of UWB besides covert military operations are for imaging systems, due to short pulses whose lengths are less than the target dimensions, and the related sensitivity of scattering. This has been used for radar systems, as previously mentioned, but also for underground imaging [48, 49], through-wall imaging [50], ocean imaging [51, 52], and medical diagnosics [53].

2.3 Microstrip Antennas

A microstrip antenna, also called a patch antenna, consists of a patch of metal on top of a grounded substrate. The metal patch can be a variety of shapes, but typically rectangular or circular are most common. Figure 2.2 shows an illustration of a rectangular patch antenna.

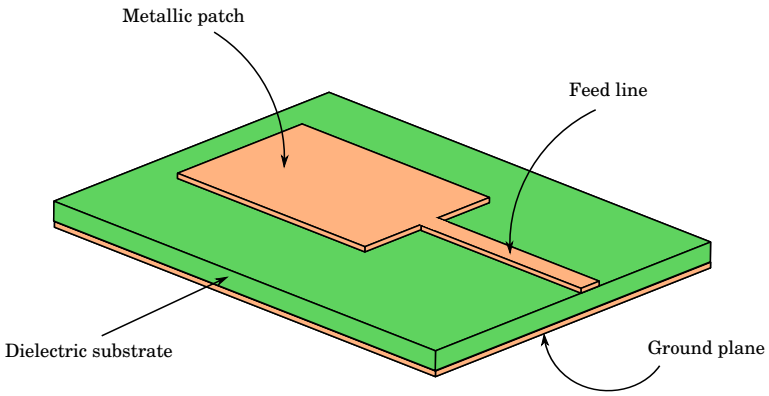


Figure 2.2: Rectangular patch antenna with microstrip feed line

Radiation occurs when the patch is excited by a feed, and a charge distribution is induced between the ground plane and the patch area. The largest charge density distributions are around the edges, resulting in fringing fields. It is the fringing fields that are responsible for the radiation.

There are many advantages of patch antennas, including light weight and low volume, low profile planar configuration – which can be made conformal – minimal fabrication cost, support of linear and circular polarization, easy integration with microwave integrated circuits, multi-frequency operation capability, and general mechanical robustness.

Microstrip patch antennas tend to have a number of disadvantages, however. Some of the shortcomings of a patch antenna are small bandwidth, low radiation efficiency, low gain, extraneous radiation from feeds and junctions, low power handling, and surface wave excitation [54].

There are a variety of feeding methods for patch antennas, each of which has an effect on the size, fabrication, impdedence and bandwidth. The most prominent methods are microstrip line, coaxial probe, aperture coupling and proximity coupling. A microstrip feed, consisting of a conducting strip extending from the edge of the patch, etched directly on the substrate, has the advantage of simplicity and a low profile configuration, but can suffer from spurious radiation. The coaxial and aperture coupled feed both produce narrow bandwidths. Unlike the coaxial and aperature feeds, the

proximity feed provides a high bandwidth, and is also free from spurious radiation. However, the proximity fed patch antenna has the disadvantages of fabrication complexity and overall increased thickness [55]. Thus, for applications requiring a low profile configuration, the microstrip feed line is essential.

Due to the complexity of obtaining analytical solutions of Maxwell's equations for the boundary conditions associated with the antenna geometries, two prominent models are used to characterize microstrip antennas – the transmission line model and the cavity model. The transmission line model is conceptually easier and less complex than the cavity model; however, the transmission line model is less accurate. Even so, useful results for certain patch antenna parameters have been obtained, such as a formula for the resonant frequency for any transverse magnetic (TM) propagation mode where there is only an electric field component along the direction of propagation [56],

$$f_0 = \frac{c}{2\sqrt{\epsilon_{r,eff}}}\left[\left(\frac{m}{L}\right)^2 + \left(\frac{n}{W}\right)^2\right]^{\frac{1}{2}} \tag{2.1}$$

where c is the speed of light in vacuum, L is the patch length, W is the patch width, m and n are the resonant mode numbers along L and W, respectively, and $\epsilon_{r,eff}$ is the effective (relative) dielectric constant, which is given by [57]

$$\epsilon_{r,eff} = \frac{\epsilon_r + 1}{2} + \frac{\epsilon_r - 1}{2}\left[1 + 12\frac{h}{W}\right]^{-\frac{1}{2}} \tag{2.2}$$

where ϵ_r is the relative dielectric constant and h is the dielectric thickness.

An effective dielectric constant is used to model the fact that the fringing fields are not fully contained within the dielectic substrate. From the

cavity model a useful formula for bandwidth is given by [54]

$$BW = \frac{16}{3\sqrt{2}} \frac{p}{e_r} \frac{1}{\epsilon_r} \frac{h}{\lambda_0} \frac{W}{L} q \tag{2.3}$$

where h is the substrate thickness, $\lambda_0 = 2\pi c/\omega$, and e_r is the radiation efficiency. Also,

$$p = 1 - \frac{0.16605}{20}(k_0 W)^2 + \frac{0.02283}{560}(k_0 W)^4 - 0.009142(k_0 L)^2 \tag{2.4}$$

where $k_0 = \omega/c$, and

$$q = 1 - \frac{1}{\epsilon_r} + \frac{2}{5\epsilon_r^2} \tag{2.5}$$

These formulas give physical insight into the dependence of the antenna characteristics on the patch geometry, as well as the thickness and dielectric constant of the substrate. Generally speaking, the bandwidth increases with the substrate thickness, the patch aspect ratio (W/L), and decreasing values of the substrate dielectic constant.

2.4 Bandwidth Widening Methods

Microstrip patch antennas are inherently narrowband, and accordingly, much effort has gone into determining special techniques to increase bandwidth. Although bandwidth is directly related to the thickness and material parameters of the dielectric substrate, increasing the thickness and using a low ϵ_r material is a limited approach due to increased surface wave power [54], resulting in poor radiation efficiency, spurious feedline radiation, impedance matching difficulty, distortions in radiation patterns and impedance characteristics.

Successful broadbanding of microstrip patch antennas typically is accomplished through the addition of coplanar multiresonating parasitic elements to the main patch configuration (Figure 2.3), stacking multires-

onators (Figure 2.4), or electromagnetically coupled multilayers [58].

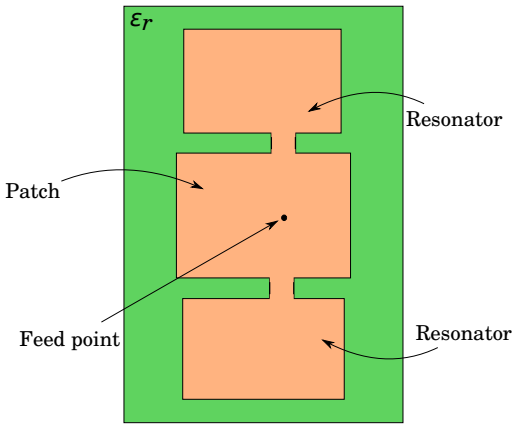


Figure 2.3: Top view of broadband microstrip antenna with directly coupled coplanar resonators. The main patch is coaxially fed from underneath

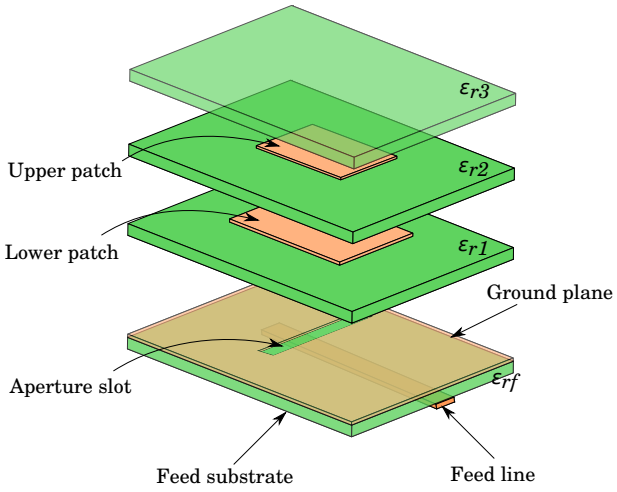


Figure 2.4: Exploded view of aperture-coupled stacked broadband microstip patch antenna. The upper patch and lower patch are slightly different in size in order to produce different resonant frequencies

The essential idea in each of these cases is to combine muliple radiating elements that are tuned to resonate at frequencies staggered across the band of interest, thereby increasing the overall bandwidth through their overlap and summation.

These methods have produced fractional bandwidths of up to 60%, significant improvements over the typical 1 – 3% of a standard configuration. However, an important drawback is that regardless of technique, the antenna size is increased tremendously, and therefore cannot be used for

applications requiring a compact geometry. Additionally, the bandwidth improvements are not enough for UWB applications.

For compact antennas, effective broadbanding solutions have been to cut a either a "U"-shaped slot into the patch (Figure 2.5 (a)) [59, 60], or use an "E"-shaped patch (Figure 2.5 (b)) [61].

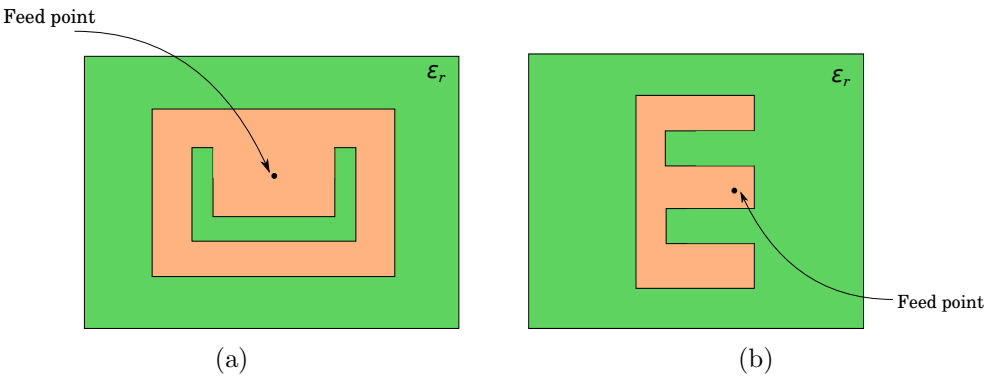
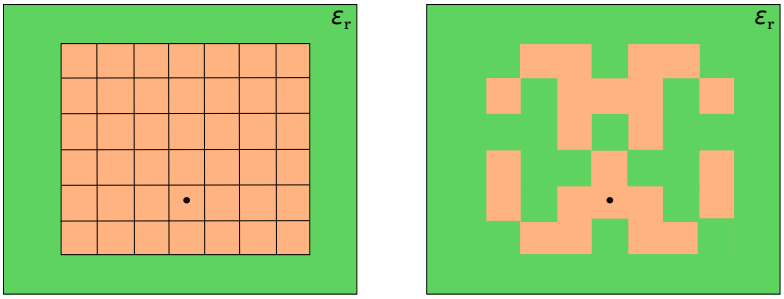


Figure 2.5: Top view of microstrip patch antennas with (a) U-slot, and (b) E-shaped patch

In these cases, the patch and the slot are designed to resonate at slightly different frequencies, to obtain overlapping multiresonances as in the previous cases. Fractional bandwidths of over 30 % have been reached with this design, while maintaining a compact profile.

A much more novel design that has produced bandwidth enhancements is the pixelated patch antenna. This type of antenna features a rectangular patch discretized into a grid of subsquares that are either metalized (ON), or air (OFF) (Figure 2.6).



(a) (b)

Figure 2.6: Top view of patch antennas (a) Metallic patch divided into 6x7 grid of ON/OFF pixels, before optimization, and (b) hypothetical design after genetic algorithm optimization

The binary description of the geometry lends itself to optimization by a genetic algorithm, in which trial geometries are evaluated and combined according to fitness. This novel technique has been used to design antennas for specific resonances, multi-resonances, increased bandwidth, or a combination [62, 63, 64, 65].

CHAPTER III

ANTENNA SIMULATION USING FINITE DIFFERENCE TIME DOMAIN

3.1 Introduction

Antenna analysis and design has been greatly improved by the use of full wave modeling. Analytical solutions of Maxwell's equations for patch antenna configurations have been unattainable, and general microstrip antenna analysis is accomplished using approximate models. These models have the advantage of closed-form solutions, numerical simplicity, and additive complexity.

However, the approximate models have several shortcomings, including limited domains of application, typically for very thin substrates, geometric and feeding method restrictions, lack of anisotropic substrate modeling, and overall accuracy [54]. Full wave modeling, providing numerical solutions to Maxwell's equations, overcomes all of the shortcomings of the approximate models.

The finite-difference time-domain (FDTD) scheme is a one of the most popular computational methods for microwave problems; it is relatively simple to program compared to finite- element based solvers, highly efficient, and easily adapted to deal with a variety of problems. The FDTD scheme is typically formatted on a structured Cartesian grid and it discretizes Maxwell's equations formulated in the time domain. This tech-

nique was first introduced by K.S. Yee in the 1960s [66], resulting in the *Yee algorithm* in which the electric and magnetic field components are defined in an interleaving way, both in three dimensional space, and in time.

3.2 The Yee Algorithm

The Yee Algorithm defines the electric and magnetic fields in an interleaving way, in a three-dimensional Cartesian space, as shown in Figure 3.1.

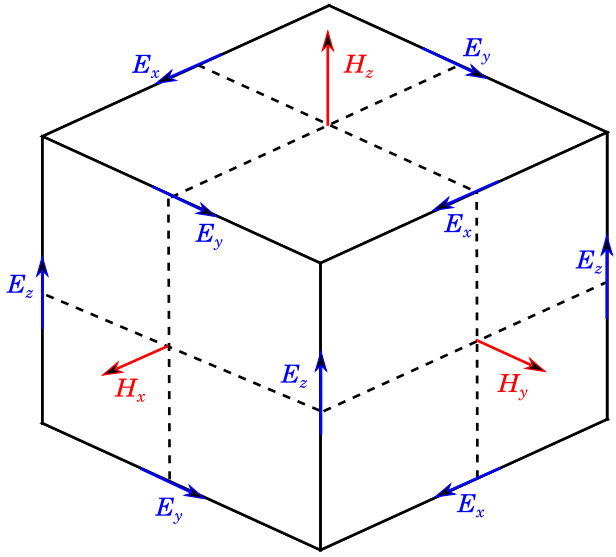


Figure 3.1: Yee cell

Each magnetic field component is surrounded by four electric field components, and similarly each electric field component is surrounded by four magnetic field components. Thus, the partial derivative of an electric or magnetic field component with respect to time at a particular point on the grid can be approximated by the central differences of the surrounding magnetic or electric field components, according to Maxwell's curl equations:

$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t} \tag{3.1}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \epsilon \frac{\partial \mathbf{E}}{\partial t} \tag{3.2}$$

where **E** is the electric filed, **H** is magnetic field, μ is the magnetic permeability of the medium, **J** is the current density, and ϵ is the electric permittivity of the medium.

Consider an arbitrary funciton $u(x)$, and its Taylor's series exapansion about the point x_0 to the points $x_0 + \Delta x$ and $x_0 - \Delta x$:

$$u(x_0 + \Delta x) \approx u(x_0) + \Delta x \frac{\partial u}{\partial x}\bigg|_{x_0} + \frac{(\Delta x)^2}{2}\frac{\partial^2 u}{\partial x^2}\bigg|_{x_0} + \dots \tag{3.3}$$

$$u(x_0 - \Delta x) \approx u(x_0) - \Delta x \frac{\partial u}{\partial x}\bigg|_{x_0} + \frac{(\Delta x)^2}{2}\frac{\partial^2 u}{\partial x^2}\bigg|_{x_0} - \dots \tag{3.4}$$

Combining (3.3) and (3.4) and rearranging gives the second-order central-difference approximation to the first partial derivative of u,

$$\frac{\partial u}{\partial x} \approx \frac{u(x_0 + \Delta x) - u(x_0 - \Delta x)}{2\Delta x}. \tag{3.5}$$

To apply this to Maxwell's equations, first denote a space point on a uniform rectangular grid at a particular time as

$$(i, j, k, n) = (i\Delta x, j\Delta y, k\Delta z, n\Delta t) \tag{3.6}$$

where $\Delta x, \Delta y, \Delta z$ are the spatial increments in the *x, y,* and *z* coordinate directions, Δt is the time increment, and *i, j, k* and *n* are integers. Furthermore, denote a function of space and time, u, and its derivative, as

$$u(i\Delta x, j\Delta y, k\Delta z, n\Delta t) = u^n_{i,j,k} \tag{3.7}$$

$$\frac{\partial u}{\partial x}(i\Delta x, j\Delta y, k\Delta z, n\Delta t) = \frac{u^n_{i+1/2,j,k} - u^n_{i-1/2,j,k}}{\Delta x}. \tag{3.8}$$

Note that in the Yee formulation, Δx is replaced by $\Delta x/2$.

Applying (3.8) to the x-component of (3.2), and assuming no current

sources ($\mathbf{J}$=0) gives

$$\frac{E_x\big|_{i,j+1/2,k+1/2}^{n+1/2} - E_x\big|_{i,j+1/2,k+1/2}^{n-1/2}}{\Delta t} =$$

$$\frac{1}{\epsilon_{i,j+1/2,k+1/2}}\left(\frac{H_z\big|_{i,j+1,k+1/2}^{n} - H_z\big|_{i,j,k+1/2}^{n}}{\Delta y} - \frac{H_y\big|_{i,j+1/2,k+1}^{n} - H_y\big|_{i,j+1/2,k}^{n}}{\Delta z}\right) \tag{3.9}$$

and rearranging produces the update equation for E_x

$$E_x\big|_{i,j+1/2,k+1/2}^{n+1/2} = E_x\big|_{i,j+1/2,k+1/2}^{n-1/2} +$$

$$\frac{\Delta t}{\epsilon_{i,j+1/2,k+1/2}}\left(\frac{H_z\big|_{i,j+1,k+1/2}^{n} - H_z\big|_{i,j,k+1/2}^{n}}{\Delta y} - \frac{H_y\big|_{i,j+1/2,k+1}^{n} - H_y\big|_{i,j+1/2,k}^{n}}{\Delta z}\right). \tag{3.10}$$

Analogously, for the z-component of the magnetic field we have

$$H_z\big|_{i,j+1,k+1/2}^{n+1} = H_z\big|_{i,j+1,k+1/2}^{n} + \frac{\Delta t}{\mu_{i,j+1,k+1/2}} \times$$

$$\left(\frac{E_x\big|_{i,j+3/2,k+1/2}^{n+1/2} - E_x\big|_{i,j+1/2,k+1/2}^{n+1/2}}{\Delta y} - \frac{E_y\big|_{i+1/2,j+1,k+1/2}^{n+1/2} - E_y\big|_{i-1/2,j+1,k+1/2}^{n+1/2}}{\Delta x}\right). \tag{3.11}$$

The basic update equations for $E_y, E_z, H_x,$ and H_y can be derived similarly. Figure 3.2 illustrates the spatial relationship between the components in the update equations for E_x and H_z, and shows a geometrical interpretation of the electric and magnetic fields as "chain-linked" arrays.

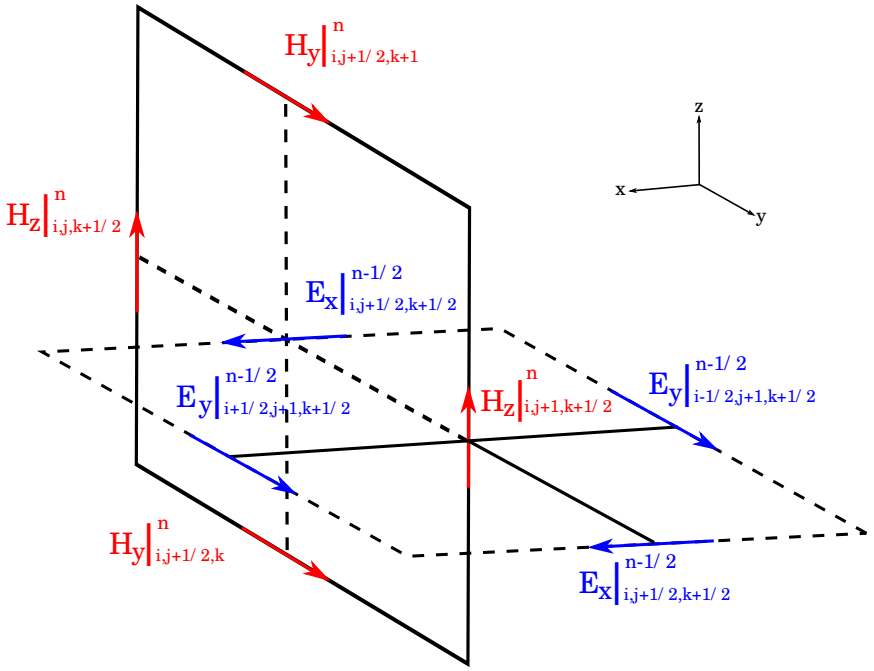


Figure 3.2: Intersecting electric and magnetic field contours

3.3 Boundary Conditions

Computing resources are limited, and therefore so is the size of the computational domain. Eventually the simulated electric and magnetic fields will propagate to the edge of the domain, and reflect back into the computational space. For antenna simulation and scattering problems, field reflections from the boundaries are unphysical and very undesirable. A number of formulations have been proposed and utilized, however the most successful has been Berenger's Perfectly Matched Layer (PML).

The idea of the PML is is to surround the computational volume with a layer that is both impedance matched to the interior, so that minimal reflection occurs, and is also lossy, so that the electric and magnetic fields that impinge upon it decay to negligible amplitudes before encountering the lattice edge. As diagrammed in Figure 3.3, this is accomplished by adding fictitious conductivities in the absorbing layer which are directionally dependent, and overlapping at the corners.

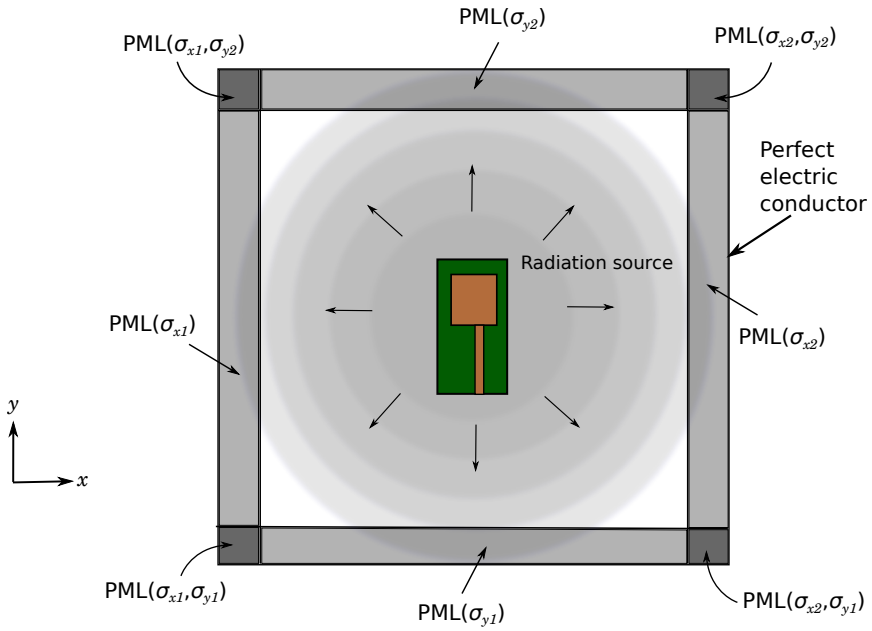


Figure 3.3: Diagram of a perfectly matched layer

In this way reflections are minimized for both normal and oblique incidence [67],[68].

The implementation of the PML splits each electromagnetic field component into two subcomponents along orthogonal directions. For example, the x-component of the electric field would be written as $E_x = E_{xy} + E_{xz}$, and Maxwell's curl equations would double to 12 when written in component form.

An alternative approach to the PML, called the *uniaxial* PML (UPML), avoids this field splitting by introducing the loss parameters into a general constitutive tensor, thereby utilizing a physical model rather than a mathematical model. The UPML also has the advantage of reduced complexity in that the boundary conditions are incorporated into the field update equations, instead of as a special case along the boundaries. To see how this works, (3.1) and (3.2) are written in time-harmonic form as [69]

$$\nabla \times \tilde{\mathbf{E}} = -j\omega\mu\bar{\bar{s}}\tilde{\mathbf{H}} \tag{3.12}$$

and

$$\nabla \times \tilde{\mathbf{H}} = j\omega\epsilon\bar{\bar{s}}\tilde{\mathbf{E}} \tag{3.13}$$

where

$$\bar{\bar{s}} = \begin{pmatrix} s_y s_z s_x^{-1} & 0 & 0 \\ 0 & s_x s_z s_y^{-1} & 0 \\ 0 & 0 & s_x s_y s_z^{-1} \end{pmatrix} \tag{3.14}$$

and

$$s_x = \kappa_x + \frac{\sigma_x}{j\omega\epsilon} \tag{3.15a}$$

$$s_y = \kappa_y + \frac{\sigma_y}{j\omega\epsilon} \tag{3.15b}$$

$$s_z = \kappa_z + \frac{\sigma_z}{j\omega\epsilon} \tag{3.15c}$$

Additionally, there are also the constitutive relations

$$\tilde{D_x} = \epsilon\frac{s_z}{s_x}\tilde{E_x} \tag{3.16a}$$

$$\tilde{D_y} = \epsilon\frac{s_x}{s_y}\tilde{E_y} \tag{3.16b}$$

$$\tilde{D_z} = \epsilon\frac{s_y}{s_z}\tilde{E_z} \tag{3.16c}$$

and

$$\tilde{B_x} = \mu\frac{s_z}{s_x}\tilde{H_x} \tag{3.17a}$$

$$\tilde{B_y} = \mu\frac{s_x}{s_y}\tilde{H_y} \tag{3.17b}$$

$$\tilde{B_z} = \mu\frac{s_y}{s_z}\tilde{H_z} \tag{3.17c}$$

Substituting (3.15a,b,c) and (3.16a,b,c) into (3.13), and using only the D_x component as an example gives

$$j\omega s_y\tilde{D_x} = \frac{\partial\tilde{H_z}}{\partial y} - \frac{\partial\tilde{H_y}}{\partial z} \tag{3.18}$$

Transforming to the time domain $j\omega \rightarrow \partial/\partial t$,

$$\kappa_y \frac{\partial D_x}{\partial t} + \frac{\sigma_y}{\epsilon} D_x = \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \tag{3.19}$$

Similarly, the E_x equation can be derived by substituting (3.15a) and (3.15b) into (3.16a), and then transforming to the time domain,

$$\epsilon \left[\kappa_z \frac{\partial E_x}{\partial t} + \frac{\sigma_z}{\epsilon} E_x \right] = \kappa_x \frac{\partial D_x}{\partial t} + \frac{\sigma_x}{\epsilon} D_x \tag{3.20}$$

The discretized update equation for D_x is obtained by applying (3.8) to (3.19) producing,

$$\kappa_y \left(\frac{D_x\big|^{n+1/2}_{i,j+1/2,k+1/2} - D_x\big|^{n-1/2}_{i,j+1/2,k+1/2}}{\Delta t} \right) +$$

$$\frac{\sigma_y}{\epsilon} \left(\frac{D_x\big|^{n+1/2}_{i,j+1/2,k+1/2} + D_x\big|^{n-1/2}_{i,j+1/2,k+1/2}}{2} \right) \tag{3.21}$$

$$= \left(\frac{H_z\big|^{n}_{i,j+1,k+1/2} - H_z\big|^{n}_{i,j,k+1/2}}{\Delta y} - \frac{H_y\big|^{n}_{i,j+1/2,k+1} - H_y\big|^{n}_{i,j+1/2,k}}{\Delta z} \right)$$

where the time approximation $D_x^n \approx (D_x^{n+1/2} + D_x^{n-1/2})/2$ was used. Rearranging gives

$$D_x\big|^{n+1/2}_{i,j+1/2,k+1/2} = \left(\frac{2\epsilon\kappa_y - \sigma_y \Delta t}{2\epsilon\kappa_y + \sigma_y \Delta t} \right) D_x\big|^{n-1/2}_{i,j+1/2,k+1/2} + \left(\frac{2\epsilon \Delta t}{2\epsilon\kappa_y + \sigma_y \Delta_t} \right) \times$$

$$\left(\frac{H_z\big|^{n}_{i,j+1,k+1/2} - H_z\big|^{n}_{i,j,k+1/2}}{\Delta y} - \frac{H_y\big|^{n}_{i,j+1/2,k+1} - H_y\big|^{n}_{i,j+1/2,k}}{\Delta z} \right) \tag{3.22}$$

The E_x update follows similarly from (3.20),

$$E_x\big|_{i,j+1/2,k+1/2}^{n+1/2} = \left(\frac{2\epsilon\kappa_z - \sigma_z\Delta t}{2\epsilon\kappa_z + \sigma_z\Delta t}\right) E_x\big|_{i,j+1/2,k+1/2}^{n-1/2} +$$

$$\left[\frac{2\epsilon\kappa_x + \sigma_x\Delta t}{(2\epsilon\kappa_z + \sigma_z\Delta t)\epsilon}\right] D_x\big|_{i,j+1/2,k+1/2}^{n+1/2} - \left[\frac{2\epsilon\kappa_x - \sigma_x\Delta t}{(2\epsilon\kappa_z + \sigma_z\Delta t)\epsilon}\right] D_x\big|_{i,j+1/2,k+1/2}^{n-1/2} \tag{3.23}$$

An analogous procedure is used to derive the B and H-component update equations. A full listing can be found in Appendix A.

The form of the components of $\overline{\overline{\mathbf{s}}}$, $s_x, s_y, \text{and} s_z$ guarantee the condition of perfect impedance continuity [70]. However, for the method to succeed, the values of κ and σ must be chosen appropriately. It follows that in the interior region of the simulation space, where there is no absorption, $\kappa_{xyz} = 1$ and $\sigma_{xyz} = 0$, but in the absorbing region κ_{xyz} and σ_{xyz} are increasing functions along the direction normal to the interface. This way, the incident fields gradually decay before impringing on the edge of the computational lattice, thereby reducing reflections to a minimum.

Typically, σ and κ use either polynomial grading or geometric grading [69]. Polynomial grading is given by

$$\sigma_x(x) = (x/d)^m \sigma_{max}, \quad \sigma_{max} = -\frac{(m+1)\ln(\Gamma_{err})}{2\eta d} \tag{3.24}$$

and

$$\kappa_x(x) = 1 + (\kappa_{max} - 1)(x/d)^m \tag{3.25}$$

where d is the length of the PML, η is the impedance of the medium, Γ_{err} is the reflection error tolerance, and κ_{max} and m are parameters. Similarly, geometric grading is defined by

$$\sigma_x(x) = (g^{1/\Delta x})^x \sigma_{x,0}, \quad \sigma_{x,0} = -\frac{\ln(\Gamma_{err})\ln(g)}{2\eta\Delta x(g^{d/\Delta x} - 1)} \tag{3.26}$$

and

$$\kappa_x(x) = (g^{1/\Delta x})^x \tag{3.27}$$

where as before d is the length of the PML, η is the impedance of the medium, Γ_{err} is the reflection error tolerance, and g is a scaling factor.

3.4 Source Conditions

Another important topic regarding the FDTD method is the question of how energy is introduced into the space-time lattice. Generally, this can be accomplished in three different ways: either as a hard-source, an additive-source, or the total field/scattered field (TFSF) formulation. Hard-sources and additive sources are easier to implement, but are limited in scope whereas the TFSF source is much more generally applicable.

As a hard-source, a time dependent function is specified at a point which represents either electric field components or current distributions. This function is typically Gaussian, or a Gaussian derivative, which takes advantage of the wide-band nature of the FDTD simulation, but can also be sinusoidal for steady state applications. However, the major drawback of the hard-source, especially if a sinusoidal function is specified, is that the evolution of the fields at the source point is not subject to Maxwell's equations and therefore unphysical. This has the effect of causing spurious reflections, polluting the solution.

The additive source is essentially the same as the hard-source, only the field or current distribution is added each time step to the source location. Most importantly, the electric and magnetic fields at the source evolve according the Yee scheme, avoiding unphysical reflections and corruption. Unfortunately both the additive source and hard source cannot simulate incident plane wave sources effectively [71].

For situations when plane waves are needed, such as for scattering prob-

lems, radar cross section calculations, or to more accurately simulate propagation along a microstrip feed line, for example, the total field/scattered field source formulation is the most useful. In this case the computational domain is divided into two areas, the total field region and the scattered field region as shown in Figure 3.4. In addition, a one dimensional lattice is constructed on which the plane wave is generated.

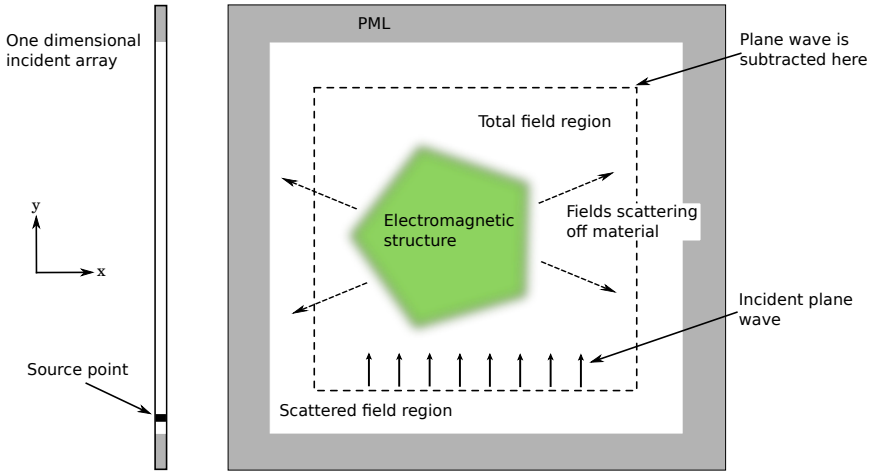


Figure 3.4: Schematic of total field/scattered field source [71]

The reason for the one dimensional buffer is that in one dimension the discretized wave equation is an exact solution of the continuous wave equation, not just an approximation [69], and also the boundary conditions are perfect [71]. At each time step, the incident wave is calculated as in additive source on the auxiliary one dimensional lattice, the results of which are then added into the three dimensional domain at one end of the total field/scattered field interface. The plane wave then propagates through the total field region using the three dimensional update equations as previously described, interacts with the structures of interest, and then is subtracted out at the other end of the TFSF interface using the field values from the one dimensional buffer.

3.5 Microstrip Patch Antenna Simulation

To demonstrate the utility of the FDTD method, as well as test the accuracy of the developed code, a microstrip patch antenna was simulated. The simulated antenna [72] is rectangular with a 12.45 mm width, 16.0 mm length, and a 0.794 mm height. Additionally, the feedline is 2.46 mm wide, 20.0 mm long and is offset from the patch edge by a width of 2.09 mm. The conductive elements, patch, feedline, and ground plane, are considered to have a thickness equal to a single discretization unit along the z direction. Figure 3.5 shows the geometry of this antenna.

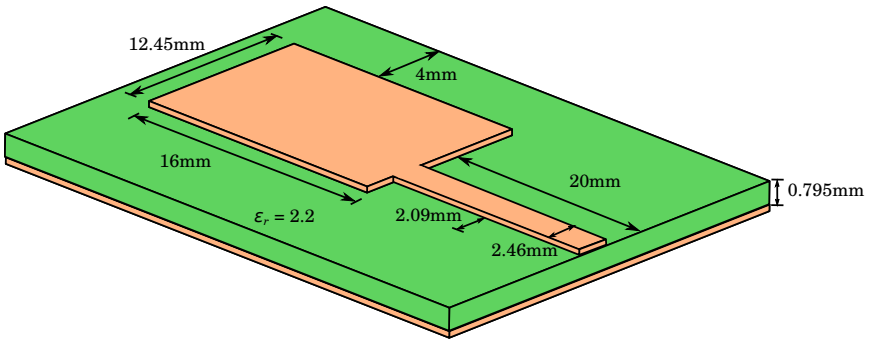


Figure 3.5: Rectangular microstrip antenna

The discretization used was $\Delta x = 0.149$ mm, $\Delta y = 0.149$ mm, and $\Delta z = 0.1$ mm. For numerical stability, the time increment must satisfy [69],

$$\Delta t \leq \frac{1}{c\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}}} \tag{3.28}$$

where c is the speed of light in vacuum. In this case the time incrememt upper limit is $\Delta t \leq 0.2148$ ps, however the actual time increment used was $\Delta t = \Delta z/2c = 0.1667$ ps. Additionally, the source function was a Gaussian monocycle,

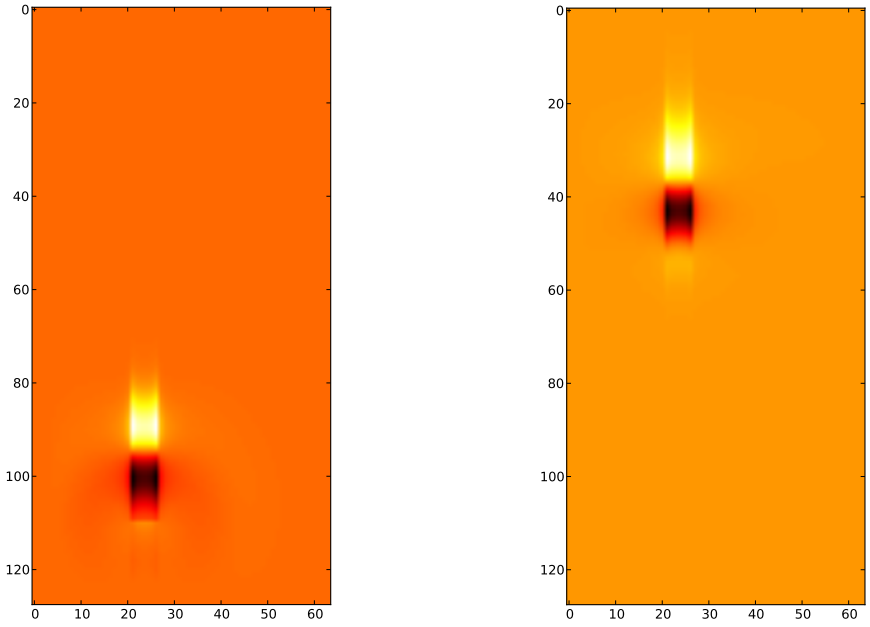
$$E(t) = -\left(\frac{t - t_0}{\sigma^2}\right) e^{-\frac{1}{2}(t-t_0)^2/\sigma^2} \tag{3.29}$$

with the time offset $t_0 = 150\Delta t$ and the pulse width $\sigma = 20\Delta t$.

To characterize the antenna, it is necessary to determine the *return loss*, which is the ratio of the reflected voltage to the input voltage at the input port, given by

$$S_{11}(f) = -20 \log \left(\frac{V_{ref}(f)}{V_{in}(f)} \right) \tag{3.30}$$

Therefore, it is necessary to perform two simulation runs. The first run is performed without the patch included, instead extending the strip line through to the end of the computational domain. Figure 3.6 shows 2-D plots on the xy plane of the E_z component at a height just below the top conducting layer after 200 time steps, and 400 time steps.



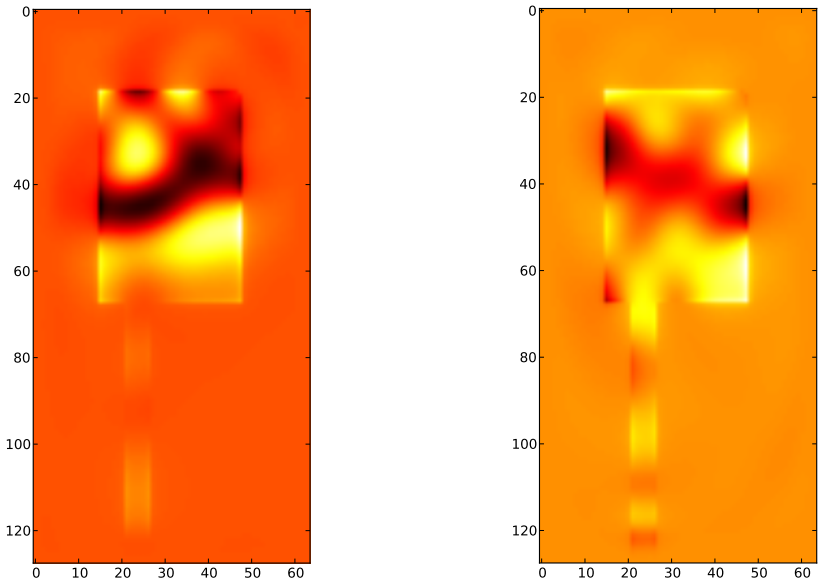
(a) (b)

Figure 3.6: Incident pulse electric field z-component, top view at (a) t = 200, and (b) t = 400 time steps

The E_z component of the electric field is recorded at a single point on the source plane at the location just underneath the top conductive strip, at each time step.

Similarly, on the second run the E_z component is once again recorded at each time step at the same location, but with the conductive patch

included in the geometry. Figure 3.7 shows 2-D plots on the xy plane of the E_z component with the patch included after 600 and 1000 time steps. These graphs illustrate the complex geometrical patters of the electric field as it resonates in the antenna.



(a) (b)

Figure 3.7: Electric field z-component with patch, top view at (a) t = 600, and (b) t = 1000 time steps

The field obtained from the second simulation can be considered a superposition of the incident field and the reflected field. To obtain the reflected field, the incident field calculated in the initial simulation run is simply subtracted.

Once the reflected field is determined, the frequency dependence of the incident and reflected fields are calculated through the fast Fourier transform. Equation 3.30 is then applied, slightly modified to use electric fields instead of voltages, which gives the same result since it is a ratio. The result plotted versus frequency is shown in Figure 3.8 along with the results using a commercial software IE3D.

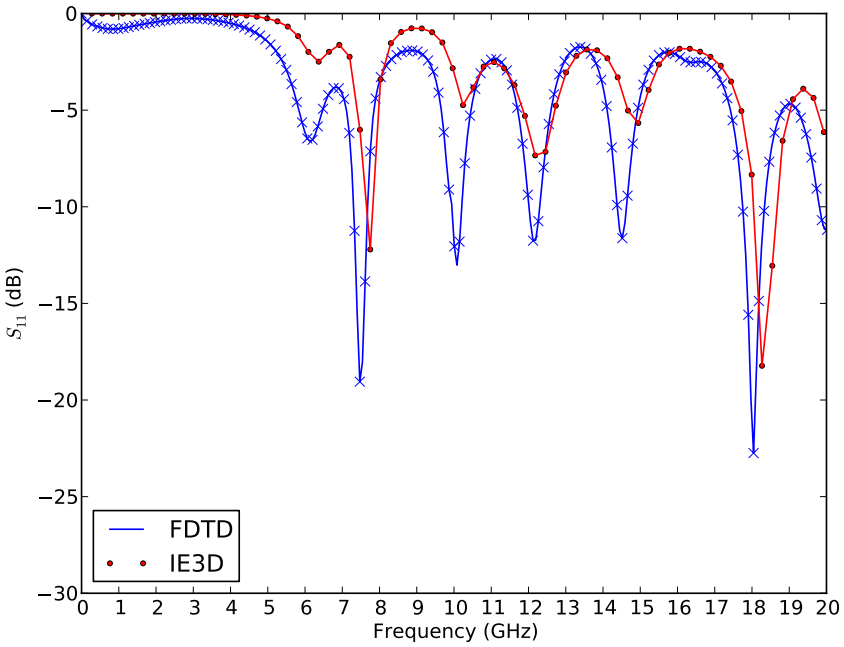


Figure 3.8: Return loss for retangular patch

It can be seen that there is some notable differences in the curves, but some discrepancy is expected since IE3D simulates electromagnetic systems in a completely different manner. IE3D is a *method–of–moments* (MoM) solver, which solves boundary–value integral equations through matrix inversion. This is a frequency domain solution method, as the solution must be calculated at specific frequencies over a range. In contrast, the FDTD method produces direct solutions to Maxwell's equations in the time domain, and obtains frequency information through Fourier transformation.

From Figure 3.8, there would appear to be less than a 10% difference in the predicted locations of the S_{11} peaks from the two different methods. Therefore it can be said that the agreement between the two is very good. Additionally, the cavity model for microstrip antennas predicts a resonant mode at a frequency of 7.6 GHz, also in good agreement.

CHAPTER IV

GENERALIZED COMPUTING ON GRAPHICS PROCESSING UNITS USING OPENCL

4.1 Overview of Computation on Graphics Processing Units

Graphics processing units (GPUs) have become increasingly recognized as a powerful platform on which to perform scientific computing. GPU computing has found its way into a wide variety of applications such as digital image and signal processing [73, 74], climate research [75], molecular modeling [76], bioinformatics [77], ray tracing [78], etc., essentially anywhere where typical CPU cluster-based supercomputing has been employed. The main advantage of using a GPU for computations that are traditionally handled by a central processing unit (CPU) is the potentially enormous accelerated computation time, for a small cost, and little system complexity. Speed-ups on the order of $30\times$ – $200\times$ CPU implementations, have been reported [79, 80], encouraging continued research in this exciting nascent field.

In an effort to understand how a single GPU can achieve the same, or greater, level of performance as a CPU cluster, it is important to have an awareness of the basic architectures of both CPUs and GPUs. For example, CPUs are designed for minimum latency, in order to respond to user action (keystrokes, mouse clicks, etc.), whereas GPUs are designed for maximum throughput, so as to able to process millions of pixels simultaneously. For

the same size chip, basic models of CPU and GPU architecture are shown in Figure 4.1.

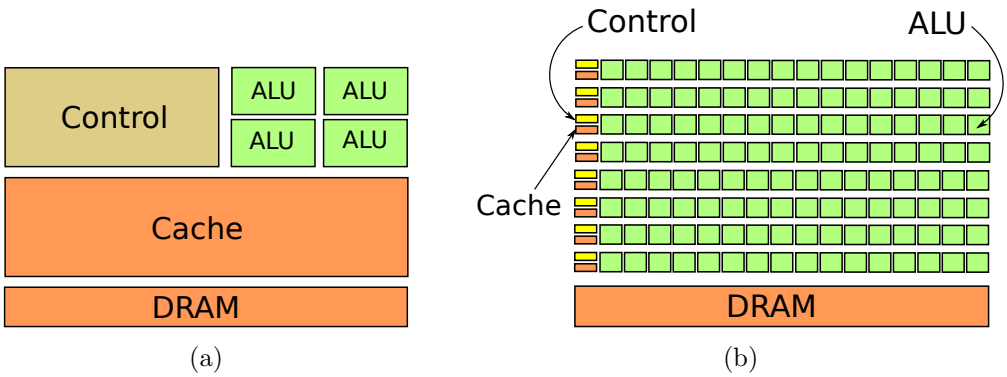


Figure 4.1: Functional schematics of (a) CPU and (b) GPU

A CPU requires a large cache, for instruction pre-fetch, out-of-order execution, etc., in order to reduce the average time to access memory. In addition, a control unit is needed to implement and manage various instructions. The remaining space on a CPU chip not being used by either the cache or control unit is filled with arithmetic logic units (ALUs), which perform the actual computations.

Graphics processors, as mentioned previously, are designed for high throughput. Since latency can be tolerated, unlike a CPU, a large memory cache is not necessary. The result is the extra chip space that is not being used for cache can be used for more computing units.

For parallel processing, CPUs use *task parallelism* in which different instructions (tasks) are run on different threads. Each thread must be individually programmed, and each thread must also be explicitly managed and scheduled. GPUs use *data parallelism*, however, the Single Instruction Multiple Data (SIMD) model, in which the same instructions are copied onto each of the processors, only each processor operates on different sets of data. In this way, programming is done for batches of threads instead of for each thread individually, and threads are managed and scheduled by the hardware instead of explicity by the programmer.

4.2 Introduction to OpenCL

Although examples GPU accelerated computations have been in the literature for at least ten years [81], little progress was made due to the complexity of programming graphics processors. The complexity results from the fact that prior to the release of the Compute Unified Device Architecture (CUDA) language by NVIDIA in 2007, and the Open Computing Language (OpenCL) by the Khronos Group in 2008, researchers had to recast scientific calculations in such a way as to be implemented through the GPU graphics pipeline. This amounted to mapping computational algorithms to geometrical objects such as polygons and triangles, and executing in terms of vertex assembly, shading, rasterization, pixel shading, blending, etc. In addition, this had to be accomplished through a device-dependent assembly language interface.

With the release of CUDA by NVIDIA, general purpose GPU programming could be done with relative ease. This is because CUDA is a high level software platform that abstracts from the GPU hardware, and also maintains a low learning curve due to its syntactic relationship and compatibility with the C programming language. The problem with CUDA, however, is that it can only be run on NVIDIA graphics cards.

Alternatively, OpenCL was developed for *heterogenous computing* applications. The concept of heterogenous computing is a framework for code that is written once, but can be run on any platform – CPU, GPU, digital signal processor (DSP), field programmable gate array (FPGA), etc., or all available platforms depending on the application. Released in 2008, OpenCL is maintained by the Khronos Group, a consortium of companies such as AMD/ATI, Apple, Intel, NVIDIA, etc., promoting open standards and dynamic media application programming interfaces (APIs).

For use with GPUs, OpenCL application code can be divided into two

separate but interdependent regimes. First, there is the *host code*, which is written in standard C/C++, and executed in serial on the CPU. Secondly, there is the *device code*, called a *kernel*, which is written in an OpenCL-specific C, and executes in parallel on the GPU. Figure 4.2 diagrams the relationship between host and device.

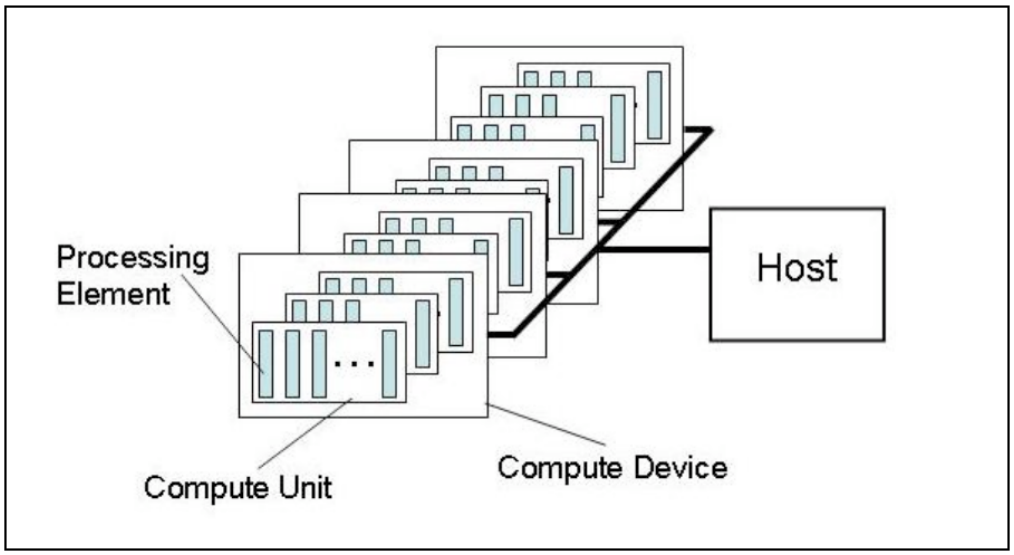


Figure 4.2: OpenCL platform model [82]

4.2.1 OpenCL Program Flow

All OpenCL applications begin with a series of steps that set up an interface between the CPU and GPU. First, a *platform* is determined, so that the OpenCL runtime is specified for a particular vendor's implementation. Next, a *context* is created, an abstraction layer to which specific operations are associated with specific devices, such as memory allocation, or compiling and running programs. After a context is created, the next initialization step is the creation of a *device*, which is basically a handle with which to reference a particular CPU, GPU, etc. Program compilation and kernel execution happen on a per device basis. Figure 4.3 shows the relationship between platforms, contexts, and devices.

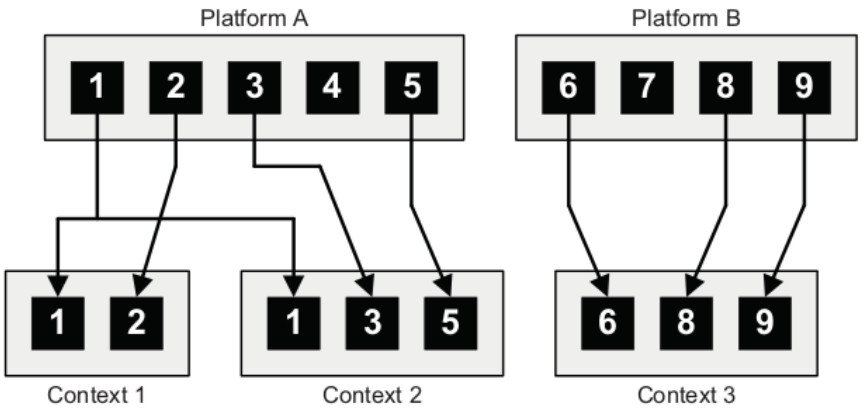


Figure 4.3: Platforms, devices, and contexts [83]

The next step in an OpenCL application is to create a *program* object. In OpenCL, a program contains either source code or binaries loaded from disk, which consists of the actual code that is intended to be run. This code can either be a single file with several functions, or from multiple source files. Additionally, the program contains a list of target devices, as well as build options. Next, the program is compiled for a specific device, or devices, on which it is intended to be run. This build process happens at runtime, using a particular OpenCL command.

A program contains entry points, called *kernels*, which are the functions intended to be executed on the device(s). Each kernel must be associated with a specific kernel object, so as to facilitate the transfer of function arguments. However, unlike function calls in the C language, each individual argument of a kernel must be separately set with a call to `kernel.setArg()`, which takes the index and value to the particular argument.

Kernel execution is first queued through a *command queue*, which is a virtual interface between the host and a specific device. Actual execution of a kernel proceeds with a call to `queue.enqueueNDRangeKernel()`, which takes several arguments specifying the dimension of the data to be processed by the kernel, as well as information regarding how the kernel is to be processed.

The process just described is executed on the host, and is essentially the same for all OpenCL applications. Particular modifications would need to be made to kernel arguments and command queue arguments for a specific application. Other modification involving synchronization would have to be made if multiple devices and command queues were being utilized.

4.2.2 Memory Management and Kernel Optimization

To achieve the best performance of a GPU, it is essential to carefully manage the usage of memory. As implemented in OpenCL, a GPU arranges its memory as shown in Figure 4.4.

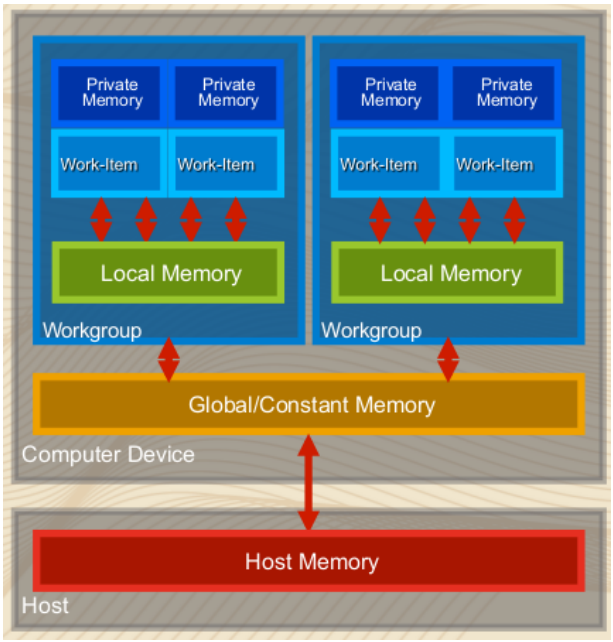


Figure 4.4: OpenCL memory model [83]

Data is passed from host to device by allocating memory buffers on the device. In terms of the model, the data that is passed from the host to the device memory buffers sits in the device's *global memory*. Each computation on data performed by a kernel is called a *work–item*. When a kernel executes, work–items execute in parallel, up to some maximum number that is device–dependent, and data is accessed from global memory.

However, the latency involved in global memory read/write operations can significantly reduce the potential gains of parallel operation performed by the GPU.

Optimizing GPU computation involves prodigious use of *local memory*. This can be taken advantage of by first dividing the work–items into *work–groups*. Typically the maximum number of work–items that can be placed into a work–group is 128, 256, 512, etc. This is the total number of work–items that can be executed simultaneously. Local memory is specific to a work–group, and read/write operations are on the order of 100x faster than global memory. All memory management is explicit in OpenCL, and the key to fast parallel processing on a GPU is to move data from global to local memory when possible, so that computations are not burdened by memory latency.

4.3 Application of OpenCL to the Finite Difference Time Domain Method

The FDTD method is inherently parallel in nature. This is due to the fact that the update equations for a particular field component are only dependent upon the component's value at a previous time, as well as spatial values of other field components. Thus, there is not a spatial relationship of a component with itself.

Parallel implementation of the FDTD on CPU distributed memory clusters has been in practice for many years, [84, 85, 86, 87]. However, there is a limiting factor in the number of processors that can be used simultaneously, and the possible speedups are less than proportional. Also, memory management and data distribution becomes extremely complex.

Due to the highly parallel nature of computation on a GPU, there has been much recent attention to the application of GPU processing to the FDTD method [79, 80, 81, 88, 89]. This is especially important as the

need grows for increasing accuracy of 3–D simulations of finely detailed structures at high frequencies. In order to obtain better accuracy and numerical stability, a fine spatial grid and small time increment is required. However, since the time increment must be made smaller, the number of iterations must be increased, and with the addition of a fine spatial grid the computational burden becomes very demanding.

Effective GPU implementation of FDTD requires careful GPU memory management in order to maximize computational time while minimizing memory reading and writing. Reported implementations have seen speedups of 12× over CPU clustering environments [88], to over 200× a CPU implmentation [80]. However, it is important to note that expected speedups depend heavily on the problem level of detail and requirements for accuracy, the hardware used, and the amount of implementation optimization.

In applying OpenCL to the FDTD method, each field component array is allocated on the host, and memory buffers are allocated on the GPU to facilitate data transfer. The host code program flow previously described is implemented, with the creation of a platform, context, command queue, etc.

For the electric and magnetic field time-stepping, each component update is called as a separate kernel function. This is to simplify the passing of function arguments, since each kernel argument must be set as a separate function call. Next, the data is partitioned into work–groups in order to take advantage of the GPU parallelism.

Specifically, for this study, the patch antenna from Chapter 3 was simulated on a $128 \times 256 \times 32$ grid, using an AMD Radeon HD 6450 GPU. This GPU is limited to 256 work–items that can be processed in parallel. Therefore, the data was partitioned in the x,y plane into 256 work–groups of size 8×16. At each time step, for a particular component, each work–

group transfers data from global memory to local memory, and executes its 256 work–items in parallel, although each work–group itself executes sequentially over the x,y plane, as shown in Figure 4.5. This procedure then repeats at each z location.

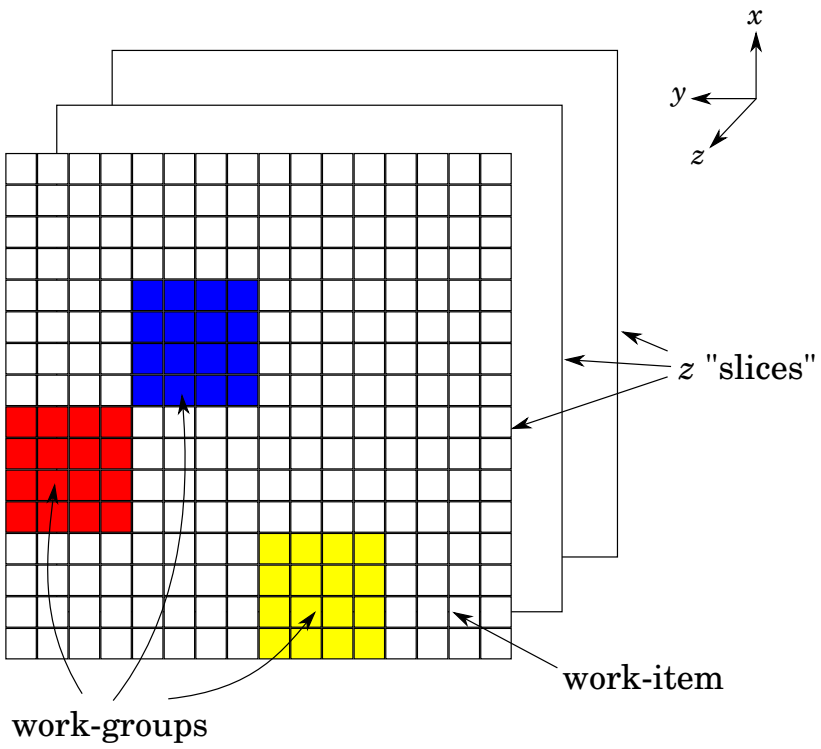


Figure 4.5: Diagram of data partitioning

For the plane wave source, using the TFSF method, the 1–dimensional plane wave propagates along the y–axis. Since the y–axis size is 256, the entire axis fits into GPU local memory. Thus, the 1D plane wave is updated everywhere simultaneously. However, the process of injecting the 1D plane wave into the 3D grid of the antenna is not as easily partitioned, so this part of the calculation was carried out sequentially.

A comparison of the 1D temporal output of the FDTD method between the CPU and GPU implementation is shown in Figure 4.6.

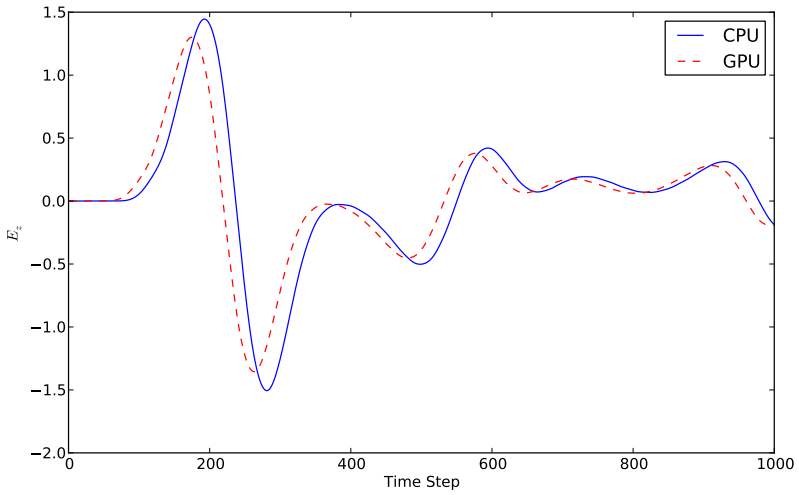


Figure 4.6: Output of FDTD method as run on a CPU and a GPU (first 1000 time steps)

Here it can be observed that the output from the GPU is about 90% of the CPU curve, and shifted left by about 9%. Overall, this is a systematic difference, probably due to the limits of the GPU as a single-precision processor. The time to complete a single run on the CPU was about 1542 seconds, while on the GPU the calculation took nearly 76.6 seconds. Thus, the GPU FDTD method was able to process about 20x faster than a single CPU.

CHAPTER V

BIO-INSPIRED OPTIMIZATION

5.1 The Genetic Algorithm

Genetic algorithms are a class of adaptive stochastic optimization algorithms which attempt to produce optimal solutions by mimicking basic evolutionary processes observed in nature. In a typical genetic algorithm (GA), a *population* of some fixed size N is defined, in which each member of the population is a binary sting called a *chromosome*. The chromosome population represents possible solutions for a particular problem, and through evolutionary operations such as *selection*, *recombination*, and *mutation*, new and potentially better solutions are generated.

From biology, strings of DNA form chromosomes which function as a "blueprint" for an organism. Furthermore, chromosomes can be subdivided into *genes*, as shown in Figure 5.1(a), each of which encode a particular protein that controls a trait, such as eye color. In a genetic algorithm, each chromosome is a string of numbers, usually fixed in length, where each gene is represented by a grouping of numbers as shown in Figure 5.1(b), usually binary, but can be integers or floating point.

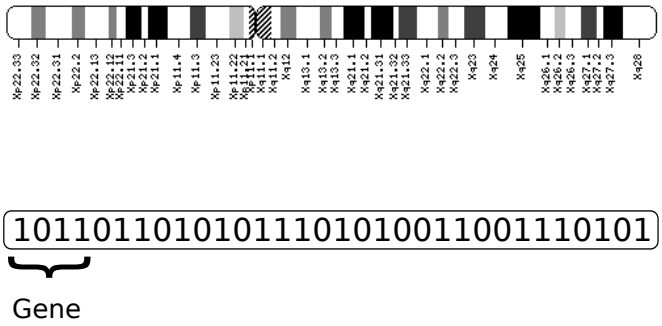


Figure 5.1: Diagram of (a) human X-chromosome [90] and (b) GA chromosome

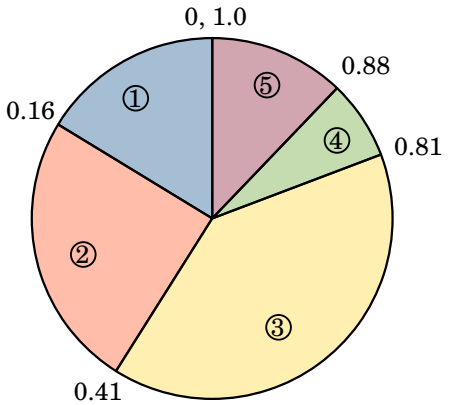
In a way similar by which genes control trait expression, each "gene" of a GA represents the value of a parameter. With an appropriate decoding scheme, the genes of a chromosome are converted from binary strings to elements of a real-valued vector. This vector is then evaluted by a *fitness function*, which is the optimization function of interest. Each chromosome is assigned a level of fitness, depending on its evaluation by the fitness function.

A population of potential solutions (chromosomes) produces new candidate solutions by combining elements in a way similar to that of genetic recombination. The new solutions are evaluated, and then replace the old solutions. However, often the best solutions from the previous generation are retained and replace the worst solutions from the current generation. Known as *elitism*, this aspect of a GA mimics a basic form of natural selection. Thus, an initial population of candidate solutions, through simulated mating and natural selection, "evolve" towards increasingly better solutions.

5.1.1 Selection

The process of recombination begins with *selection*. Chromosomes are selected from the population for reproduction, in such a way that chromosomes with a higher fitness level are more likely to be chosen. A very com-

mon method of selection is *fitness-proportionate selection*, which is simply implemented with "roulette–wheel sampling" [91]. In roulette–wheel sampling, each chromosome is assigned a probability of selection based on its fitness level by $p_i = f_i / \sum_{j=1}^{N} f_j$, where f_i is the fitness of the ith chromosome, and N is the population size. It can be imagined that each chromosome has a slice of a circular "roulette wheel", and that the size of each slice is proportional to each chromosome's probability of selection as illustrated in Figure 5.2.



No	Chromosome	*X*	Fitness *f*(*X*)	% of Total
1	010101010010	(2.56, 0.865)	2.53	16
2	001011101001	(1.876, 5.16)	3.85	25
3	111011001011	(15.12, 1.59)	6.17	40
4	010110001011	(8.67, 10.35)	1.1	7
5	100010010110	(22.5, 0.26)	1.89	12
Totals			15.54	100

Figure 5.2: Roulette–wheel selection example

A spin of the wheel is simulated by generating a random number r between 0 and 1. Next, the slice in which r resides is determined from the cumulative selection probabilities, $\sum_{j=1}^{i-1} p_j \leq r \leq \sum_{j=1}^{i} p_j$. The ith chromosome corresponding to this slice is then selected for mating. This process is then repeated until all of the selections have been made.

While roulette wheel sampling is easy to implement and has an intuitive sense of fitness-proportionate selection, in practice chromosomes are not always selected according to their probabilities [92]. This is due to the fact that GA populations are typically small, less than 50 individuals in

most cases, and roulette wheel selection probabilities are the statistical expectation values that would be obtained after many samples from a large population. Therefore, there is a possibility that less fit chromosomes could be selected disproportionately.

Another fitness-proportionate selection method is *stochastic universal sampling* [93]. In stochastic universal sampling (SUS), a random number is generated, and a chromosome is selected as in roulette–wheel sampling. However, instead of generating a new random number every time a chromosome is to be selected, a predetermined fixed amount is cumulatively added to the random number for each selection. In this way, less fit members are neither disproportionally dominated by more fit solutions, nor disproportionally selected due to a statistically insufficient population size, as in roulette–wheel sampling. However, SUS as well as roulette–wheel sampling have a drawback in that population variances can decrease quickly, resulting in a stagnating evolution before an optimal solution is reached.

Besides roulette–wheel sampling, SUS, and other finess-proportionate selection methods, there are also alternative selection methods such as *rank selection* and *tournament selection*. Rank selection assigns a value from 1 to N, to each member of the population based on increasing fitness. Selection probabilies are then linearly assigned according to

$$p_i = Min + (Max - Min)\frac{i-1}{N-1} \tag{5.1}$$

where Min is the reproduction rate of the worst individual, and $Max = 2 - Min$. Probabilities can also be assigned by exponential ranking, according to

$$p_i = \frac{c^{N-i}}{\sum_{j=1}^{N} c^{N-j}} \tag{5.2}$$

where c is a parameter, and $0 < c < 1$. The advantage of rank selection is that diversity is maintained, resulting in a larger sampling of the solution

space. However there are speed disadvantages since the entire population must be sorted to produce the rankings.

Tournament selection involves randomly selecting M members of the population, $1 \leq M \leq N$, and then selecting the most fit individual from this group. The procedure is then repeated N times. Selection pressure is easily adjusted by changing the tournament size, M. With a small tournament size, less fit individuals have more opportunity, but as $M \to N$, better fit solutions dominate. Note that $M = 1$ is equivalent to purely random selection.

5.1.2 Recombination and Mutation

After the selection process, the selected individuals exchange portions of their binary strings in process called *recombination* or *crossover*. The biological process of recombination occurs during the first stage of meiosis, in which chromosomes form pairs and exchange different segments of genetic material. New combinations of DNA are created, which are a significant source of genetic variation, and may result in beneficial new combinations of alleles (variations of a gene). Finally, these offspring are subject to *mutation*, where single elementary bits of DNA (nucleotides) are changed, often as a result of copying errors.

Similarly, in a GA recombination is implemented typically through *single–point crossover*, *double–point crossover*, or *uniform crossover*. Single–point crossover is the simplest method and entails generating a random number, r, $r < M - 1$, where M is the length of the parents' binary string, to be the crossover position, and then exchanging the respective parts of the parents (Figure 5.3(a)).

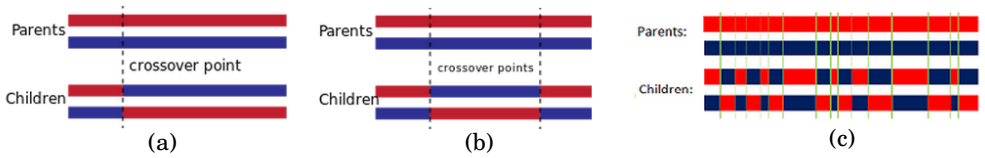


Figure 5.3: Diagrams of GA crossover methods, (a) single–point, (b) double–point, and (c) uniform

This method does have problems, however, in particular "positional bias" and "endpoint effect" [92], in which the length of the strings being exchanged can disrupt potentially good solutions, and the strings exchanged always contain the endpoints.

Using double–point crossover, two crossover positions are selected at random, and everything between the two points in the parent chromosomes is swapped to create the child chromosomes (Figure 5.3(b)). In this way, some of the biasing observed in single–point crossover can be mitigated. The single–point and double–point crossover methods can be regarded as special cases of a generalized n–point crossover method. Increasing the number of crossover points can potentially result in a more exploratory search. However too many crossover points can be highly disruptive of good solutions, resulting in a very slow convergence, or no convergence to an optimal solution [92].

In the case of uniform crossover (Figure 5.3(c)), each position in a parent string is assigned a probability for exchange. A random number r is generated, and if $r \leq p_i$, where p_i is the probability for exchange at chromosome position i, then an exchange occurs. Uniform crossover explores the solution space more effectively than single or double–point crossover, however can be disruptive to strings that form solutions with high fitness.

After selection and crossover, offspring are subject to *mutation*. Mutation in biology is a small change in DNA; similarly in a GA, mutation is implemented as a bit flip at a random position in a string. The effect is to maintain some amount of diversity in the population in order to avoid

solutions converging to a local extremum. However, mutation is considered as a background operator to the main operation of recombination [94]. The process of selection, recombination and mutation repeats until either a set number of iterations is reached, or a minimum fitness level is attained. Figure 5.4 illustrates the process.

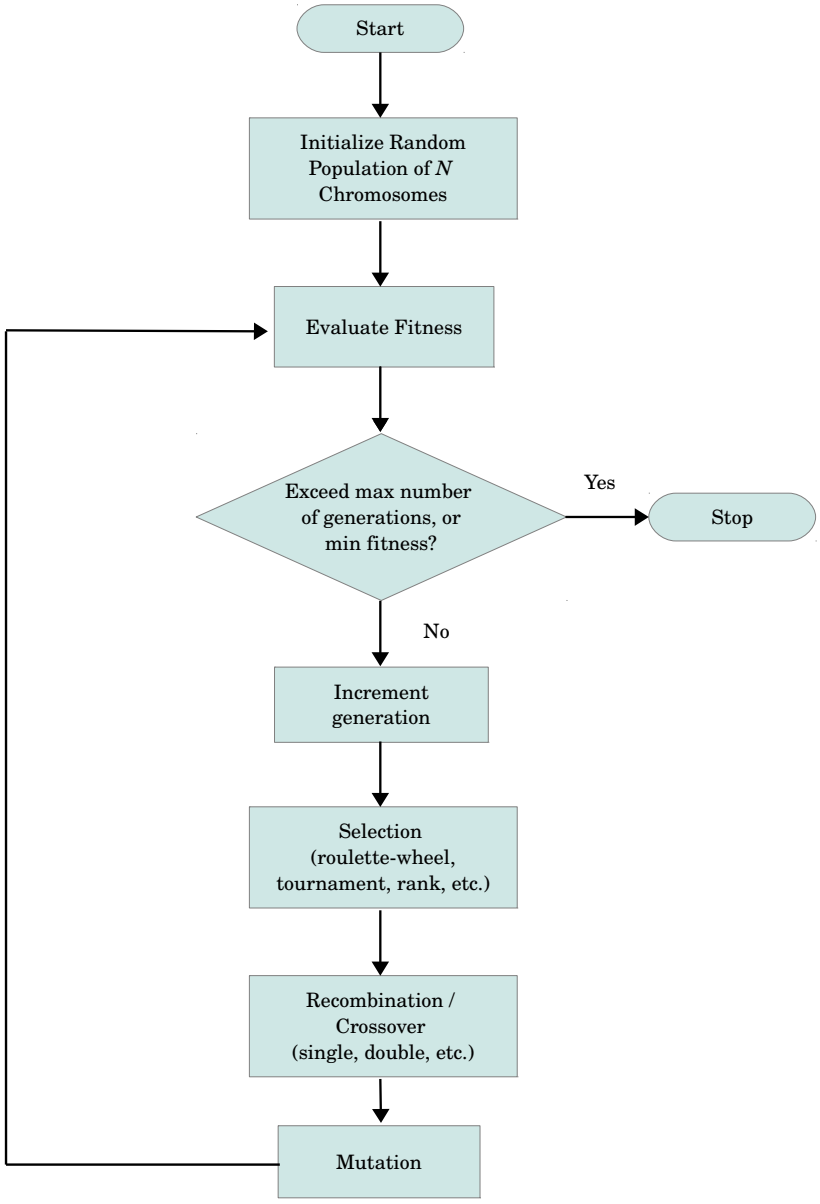


Figure 5.4: Diagram of a GA process

5.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is another population–based adaptive stochastic optimization method like the GA. However, unlike the GA, there are no evolutionary operators such as selection, crossover, or mutation. Instead, PSO is based on the collective swarm intelligence observed in the social behavior of bees, birds, fish, etc., [95], as they search for the best feeding locations. Potential solutions are referred to as *particles*, and instead of evolving towards optimal solutions, they "fly" through the problem space along trajectories determined by the best positions encountered by both an individual particle, and the swarm as a whole.

In PSO, each particle represents a possible solution to the optimization problem, with an associated position or location, and velocity. A fitness function is used to evaluate the potential solutions, and determines which are the best locations. Each particle keeps track of its own personal best location (*pBest*), as well as the global best (*gBest*), the best location visited by the entire swarm. If a particle's current position has a better fitness value than it's *pBest*, its current position replaces its *pBest*, and if any of the $pBest_i$ are better than the current *gBest*, it is also replaced.

A particle's position and velocity are updated according to

$$\mathbf{v}_i^{k+1} = w^k \mathbf{v}_i^k + c_1 r_{1,i}^k (\mathbf{p}_i^k - \mathbf{x}_i^k) + c_2 r_{2,i}^k (\mathbf{g}^k - \mathbf{x}_i^k) \tag{5.3}$$

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^{k+1} \Delta t \tag{5.4}$$

where $\mathbf{v}_i^k$ is the velocity, w^k is the inertial weight, which determines the influence of the particle's previous trajectory, p_i^k is the location of the ith particle's *pBest*, and g^k is the location of *gBest*. The superscript k refers to the kth iteration. The parameters c_1 and c_2 are the *cognitive weight*, and the *social weight*, which determine whether a particle has a tendency

towards *pBest* or *gBest*. Finally, $r_{1,i}^k$ and $r_{2,i}^k$ are uniformly distributed random numbers between 0 and 1, and Δt is the time step, where it is standard for $\Delta t = 1$. The basic algorithm is shown in Figure 5.5.

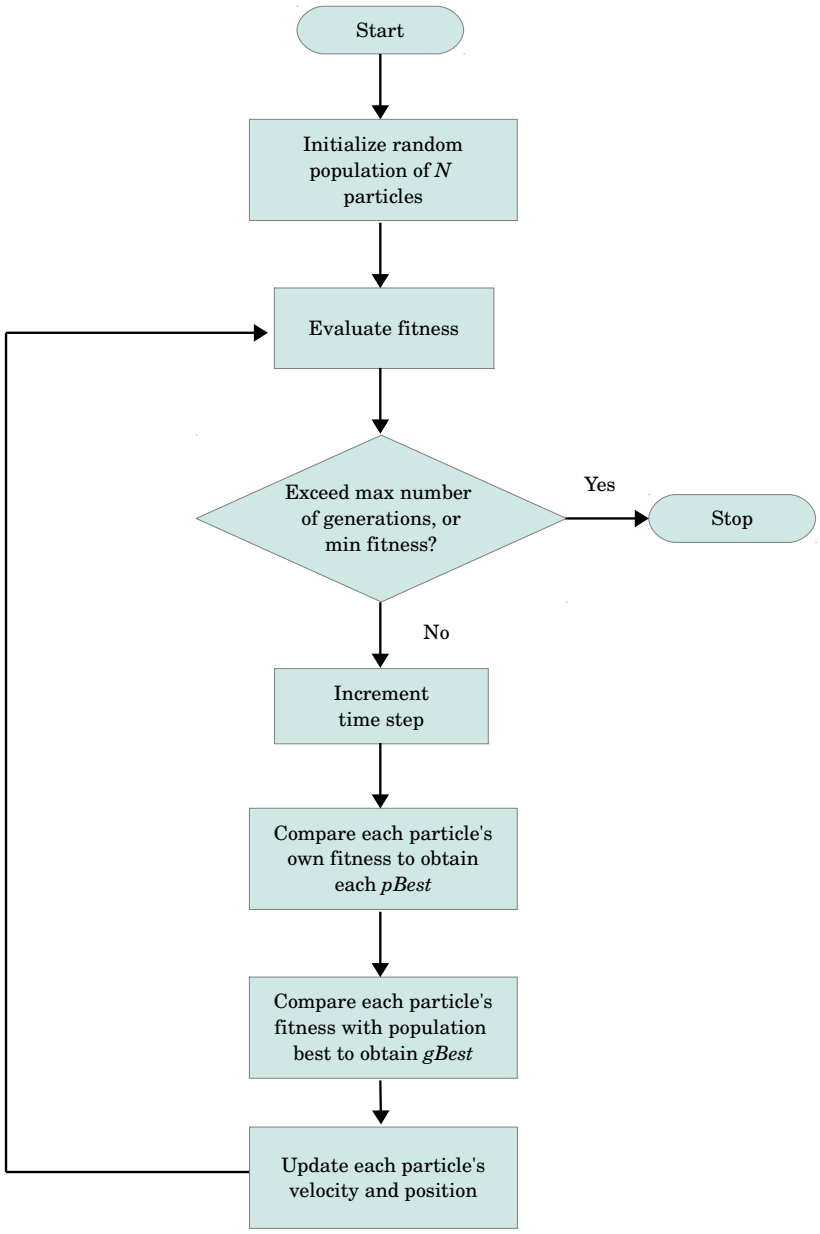


Figure 5.5: PSO algorithm

Unlike a typical GA, PSO is usually implemented using floating point representations of solutions, instead of binary codings. However, PSO does have binary variants. In binary PSO (BPSO), [96], the velocity update equation (5.3) remains the same, but is given a new interpretation. Instead

of a velocity of a particle, the v_i represent the probability that the bit x_i will take the value 1. Thus, from (5.3), $\mathbf{v}_i^k$ is constrained to the interval [0.0, 1.0], and $\mathbf{x}_i^k$, $\mathbf{p}_i^k$, and $\mathbf{g}_i^k$ are binary strings. The other parameters remain the same.

The other modification to PSO for its binary variant is the position update equation, as (5.4) is no longer valid since velocities are now probabilites. This is accomplished by use of a sigmoid function to normalized the velocities,

$$Sig(v) = \frac{1}{1 + e^{-v}} \tag{5.5}$$

and then to update the binary string representing position,

$$x_{ij}^{k+1} = \begin{cases} 1, & \text{if } r_{ij} < Sig(v_{ij}^{k+1}) \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

where r_{ij} is a uniformly distributed random number over the interval [0.0, 1.0].

5.3 Biogeography–Based Optimization

Biogeography is the study of the distribution of plant and animal species and ecosystems in both space and time. Combining research from ecology, evolutionary biology, geology and geography, biogeography is an integrative field that attempts to understand how species are distributed across geographical areas, and in particular how environmental dynamics influence evolution.

Like the GA and PSO, biogeography–based optimization (BBO) is a stochastic–adaptive, population–based, heuristic optimizer that models the process of optimization on the natural dynamics of biogeography, specifically island biogeography. Island biogeography studies species diversification not only of actual islands, but any isolated natural environment such

as mountains or lakes surrounded by deserts, isolated forests, etc. The main tenet of island biogeography is that the number of species found on an undisturbed island is determined by immigration, emigration, and extinction [97]. Furthermore, immigration and emigration are affected by distance to neighboring islands or the mainland, while the size of the island itself has an effect on the extinction rate. In addition to environmental and geographical effects, there is also the genetic evolution of the species themselves.

BBO attempts to find optimal solutions to problems by simulating the dynamics of species immigration and emigration. Similar to a GA, potential solutions are encoded as binary strings, where in this case, each string is considered an *island*. The bits themselves can be envisioned as species of plants or animals living on a particular island, or some quality of habitability. In the language of BBO, a bit, or island element, is referred to generally as a *suitability index variable* (SIV). All of the potential solutions together form a population of islands, or an archipelago.

As in the case of other population–based heuristic optimizers, the quality of a potential solution is determined by a fitness function. For BBO, the quality of a solution is known as its *habitat suitability index* (HSI). The optimization process is then analogous to finding an island that has a high HSI, and is therefore very habitable.

Each potential solution, or island, has an associated immigration rate λ, and emigration rate μ, determined by

$$\lambda_{i-1} = \frac{i}{N} \tag{5.7a}$$

$$\mu_{i-1} = 1 - \lambda_{i-1} \tag{5.7b}$$

where $i = 1, 2, ...N$, N is the size of the population, and the population is sorted from best to worst. Thus, islands with a high HSI have a high

emigration rate and a low immigration rate, and correspondingly, islands with a low HSI have a low emigration and a high immigration rate.

Information is exchanged between islands through a selection process similar to a GA, where immigrating and emigrating islands are chosen probabilistically using roulette–wheel selection. Next, elements of an island are selected to be modified according to immigration rates, and the emigration rates of the other islands are used to determine which solutions migrate a randomly selected element [98]. After all island updates are completed, mutation can be applied like in a GA (Figure 5.6).

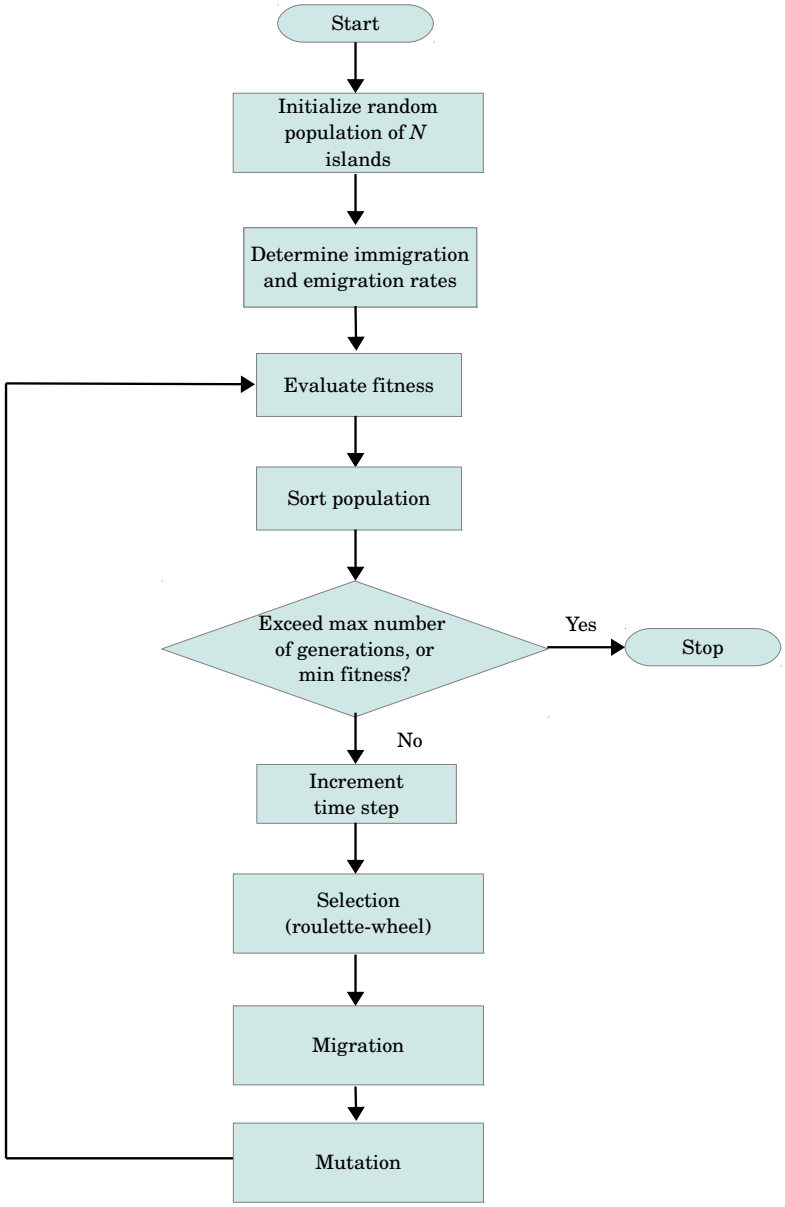


Figure 5.6: BBO flowchart

5.4 Optimization Example

5.4.1 Rastrigin Function Minimization

As a demonstration of the effectiveness of the GA, BBO, and PSO, consider the Rastrigin function [99],

$$f(\mathbf{x}) = 10N + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) \right) \tag{5.8}$$

where the topology of this function is shown in Figure 5.7 for $n = 2$.

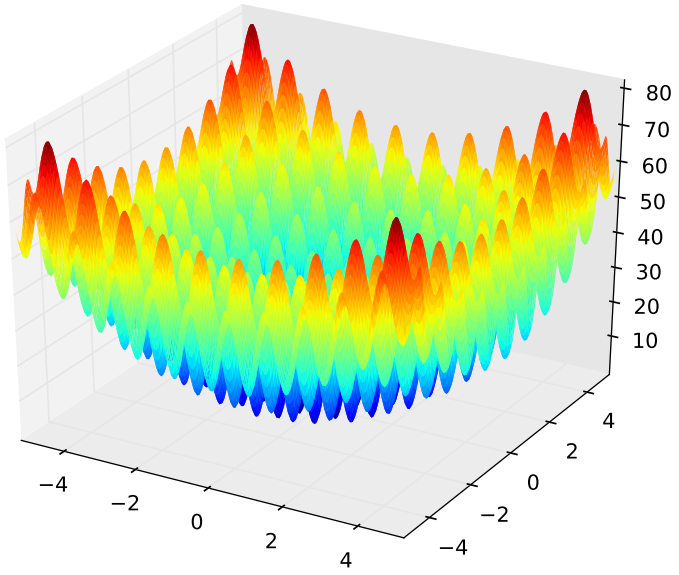


Figure 5.7: Rastrigin function with $n = 2$

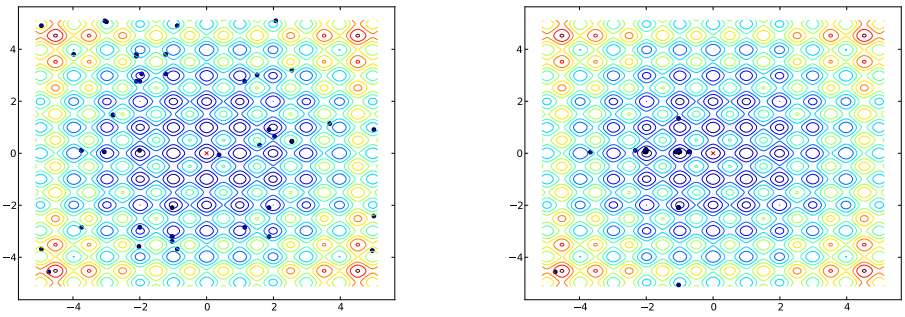
The Rastrigin function is highly multimodal, with frequent local minima. Its global minimum is $f(\mathbf{x}) = 0$, for $x_i = 0$, $i = 1, ..., n$. Also, the search area is restricted to $-5.12 \leq x_i \leq 5.12$.

For the two dimensional case, $N = 2$, a GA was run with a population size of 45, 16 bits per variable, 0.50 elitism rate, and 0.008 mutation rate.

Each set of 16 bits was decoded to a real number through the mapping

$$g(\mathbf{x}) = Min + \left(\frac{Max - Min}{2^L - 1}\right) \sum_{j=0}^{L-1} 2^{L-1-j} x_j \tag{5.9}$$

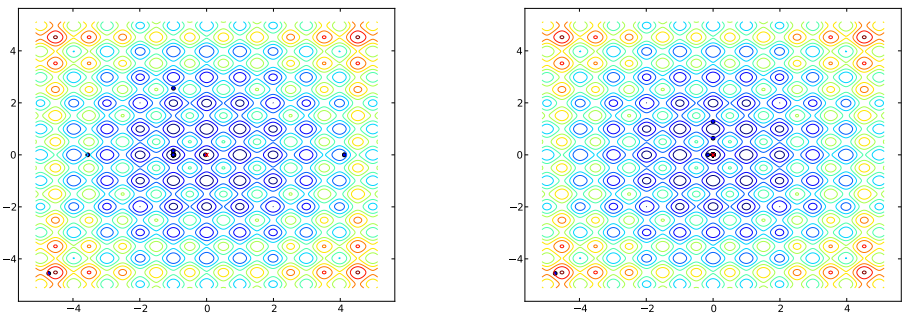
where $(Min, Max) = (-5.12, 5.12)$, and $L = 16$, the number of bits per variable. Figure 5.8(a) shows the initial random population distribution, and (b) shows the population after 5 generations, where some of the solutions have gathered around a few local minima.



(a) (b)

Figure 5.8: Rastrigin function contour with GA population distribution (a) initial (b) after 5 generations

As seen in Figure 5.9(a), after 27 generations some of the solutions have converged on the global minimum, and after 50 generations (Figure 5.9(b)), most of the solutions have converged.



(a) (b)

Figure 5.9: Rastrigin function contour with GA population distribution (a) after 27 generations (b) after 50 generations

Now consider an example of the Rastrigin function where the problem dimension is 30. Here, the GA, PSO, and BBO were run for 10000 iterations. The population size for each method was 45, and the number of bits was 480, so that each variable had a 16–bit representation. The GA parameters were an elitism rate of 0.4, and a mutation rate of 0.008. Also, roulette–wheel sampling and double–point crossover were used. For BBO, the same elitism and mutation rates were used. The PSO paramters were $w = 0.75$, $c_1 = 2.0$, and $c_2 = 1.75$. Figure 5.10 shows the convergence of the best solutions and Figure 5.11 shows the average fitness values.

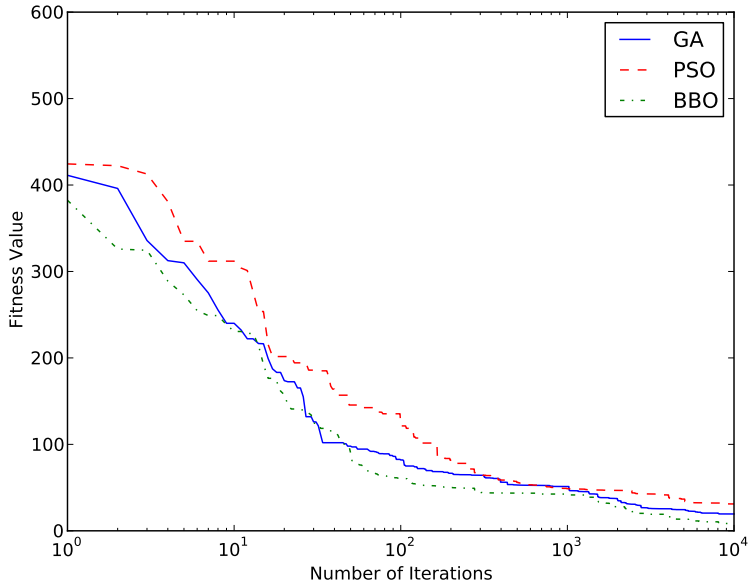


Figure 5.10: Rastrigin function convergence, for $n = 30$. Best solutions.

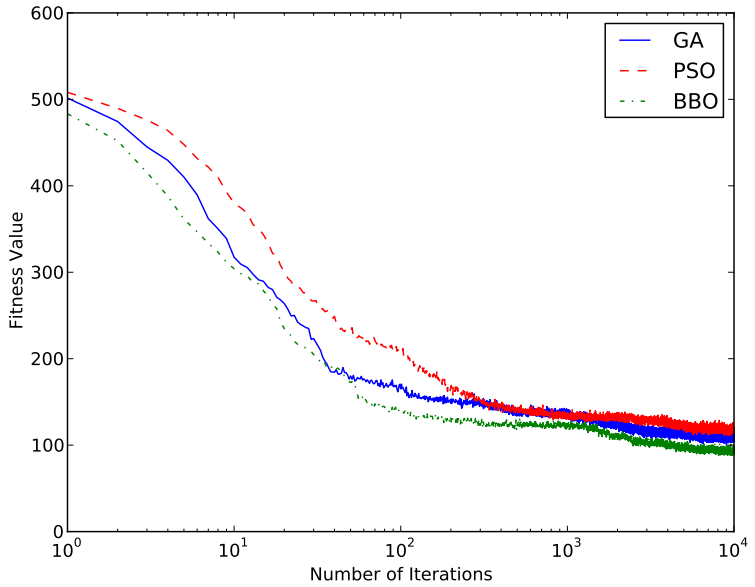


Figure 5.11: Rastrigin function convergence, for $n = 30$. Average solutions.

From Figures 5.10 and 5.11 it can be observed that the best and average solutions obtained by each method increasingly improve each iteration. However, the rate of improvement slows down considerably after an initial period. For example, the GA had an initial fitness of 411.233 which improved by almost 80% to 97.396 after only 50 iterations, but another approximately 80% improvement from 97.396 to 19.947 took 9,857 iterations. PSO and BBO show similar fitness improval rates. The final fitness values after 10,000 iterations were 19.479 for the GA, 31.064 for PSO and 8.415 for BBO, so in this case BBO outperformed both.

5.4.2 Parameter Considerations

Evolutionary optimization algorithms such as the GA, BBO, or PSO have several parameters that must be selected at the start of the optimization process. The choice of population size, elitism and mutation rate, etc., all affect the ability of the algorithm to converge to optimal values. In order to understand some of the effects of parameter values the previous example of the Rastrigin function optimization was repeated, but with different

parameters.

Figure 5.12 shows the results of GA optimization for variable elitism rate, mutation rate, and population. Since the ranges of the parameters values differ significantly, all values were rescaled from 0 to 1 so that trends could be easily compared.

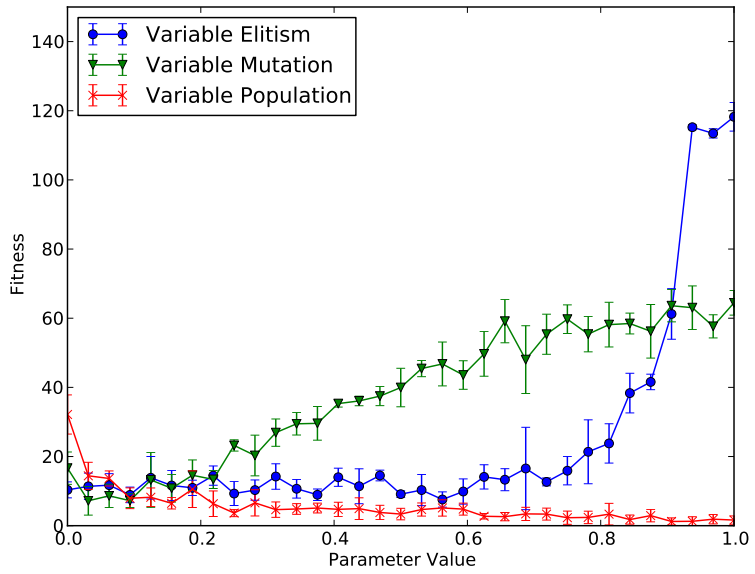


Figure 5.12: GA average best fitness for variable parameters

In each of the three cases the parameters that were not varying were fixed at the values used in the previous example. The fitness values are the average of 25 optimization runs using a particular set of parameters, and the error bars are 1 standard deviation. The elitism parameter ranged from 0.15 to 0.95, mutation from 0.001 to 0.2, and population from 10 to 330, with 32 data points total for each.

It is clear from Figure 5.12 that in general low mutation rates and larger populations give the best results. Although the average fitness improves as the population size increases, most improvement is seen as the population increases from 10 to 30, and only slowly for larger values. Mutation rates between 0.007 and 0.02 gave the best results, and gave increasingly worse results as mutation rate increased. As elitism rate increases and

more solutions are used in recombination, the average fitness does not vary appreciably until about an elitism value of 0.725, afterwhich the average fitness rapidly worsens. Also, the sizes of the error bars indicate that overall the optimization results are consistent for a certain set of parameters, except for a few isolated cases.

Figure 5.13 shows the results of an identical procedure to test BBO.

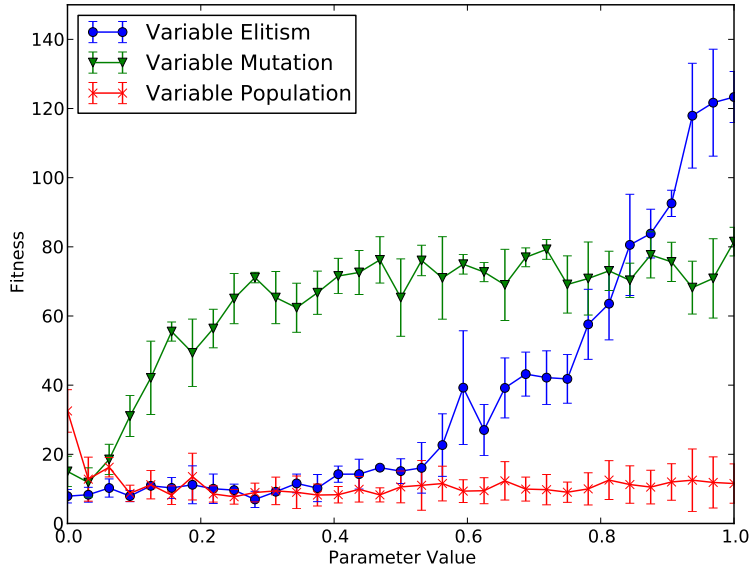


Figure 5.13: BBO average best fitness for variable paramters

Generally for BBO there is more variability in the average fitness values, and a greater sensitivity to mutation than the GA. Also, fitness improved as population was increased from 10 to 30, as for the GA, but fitness did not continue to improve for increasing populations. The lowest mutation rates, 0.001 and 0.007, gave the best results, while results quickly worsened as mutation increased until about a value of about 0.06, where the fitness reached a plateau with respect to increased mutation. The effects of elitism were minimal until an elitism value of 0.475, afterwhich solutions rapidly became less fit with greater variability.

PSO has a different set of parameters than either the GA or BBO, so a direct comparison of the effects of elitism and mutation is not possible.

Figure 5.14 shows the output of varying PSO parameters on the Rastrigin function, using the same procedure as for the GA and BBO.

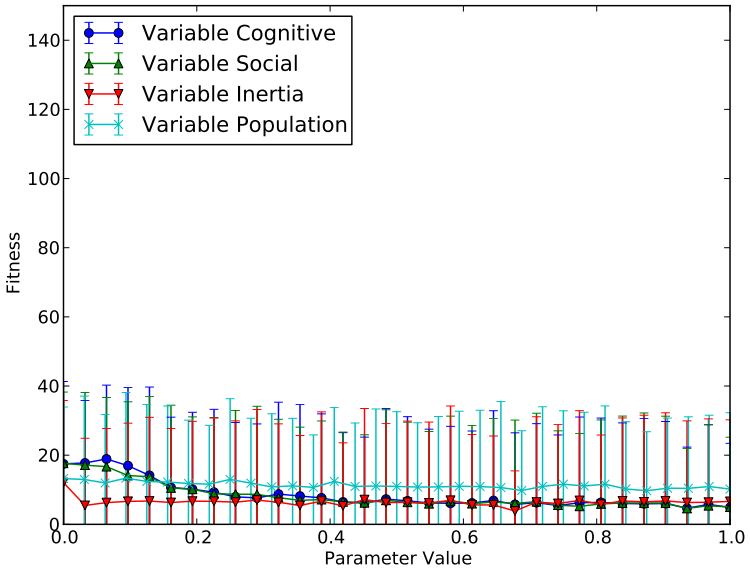


Figure 5.14: PSO average best fitness for variable paramters

In this case, the cognitive weight c_1 and social weight c_2 varied from 1.0 to 9.0, the inertial weight w from 0.5 to 8.5, and the population from 10 to 330. Unlike the GA or BBO there are not as obvious discernable trends, other than some slow improvement as c_1 or c_2 is increased from 2.25 to 4.25. From Figure 5.14 it can be observed that on average, PSO consistently outperforms the GA and BBO with respect to variation of parameters. However, the error bars show a large variability, therefore PSO would not necessarily outperform the GA or BBO in isolated cases.

CHAPTER VI

DESIGN OF UWB ANTENNAS

6.1 Design Strategy

The main constraint in compact antenna design is the size, which limits the bandwidth and gain. However, using optimization techniques such as the genetic algorithm, or particle swarm optimization, it has been shown that novel geometries are possible that can overcome apparent limitations [65]. The overall strategy in antenna design using a GA, or related method, is to first determine the parameters for a given design goal, such as antenna dimensions. Next, the GA (PSO, BBO, etc.) generates a set of possible solutions. Each of these possible solutions is evaluated by electromagnetic simulation, such as with the FDTD method, to determine the performance of the antenna for the given paramters. If a solution meets the design criteria, the optimization loop ends. Otherwise, new potential solutions are generated according to the evolution rules of the chosen optimization method, and the process repeats.

In designing antennas with a GA, the optimal geometry is completely unknown. The designer specifies certain constraints and design objectives, but allows the optimization routine to determine the specifics of the geometry. Therefore it is possible for the optimizer to generate novel designs that meet the performance objectives, but are not likely to have been discovered through traditional design methods.

For the application of a compact ultra–wideband patch antenna, the design method is to fix certain size constraints, but then allow the optimization method to explore uncommon patch geometries. In this case, the fixed parameters were patch width $W = 10.0$ mm, length $L = 10.0$ mm, and substrate thickness $H = 0.794$ mm. Also, the feedline width $W_f = 1.2$ mm, feedline length $L_f = 20.0$ mm, the feedline offset from the patch *offset* = 5.0 mm, and the substrate dielectric constant $\epsilon_r = 2.2$. Figure 6.1 shows the layout of the antenna configuration.

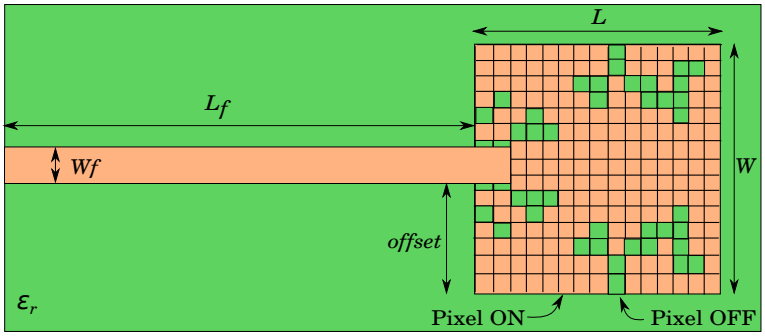


Figure 6.1: Example patch antenna configuration

The patch is divided into a 15×15 grid of pixels where a pixel is either "ON" (metalized) or "OFF" (air). Each pixel is 0.66 mm $\times$ 0.66 mm. For optimization with a GA, PSO, or BBO, a population of potential geometries is generated randomly. Every member of the population is a binary string of length $15 \times 8 = 120$ bits, representing a patch geometry where a 1 means a pixel is "ON" and a 0 means a pixel is "OFF". Only 120 bits are needed instead of $15 \times 15 = 225$ bits is because the geometry is mirrored across the central line, lengthwise.

In the design of an UWB antenna, there are several antenna parameters that can be optimized including bandwidth, radiation pattern, etc. However, the focus was only for the optimization of bandwidth, since the bandwidth of patch antennas is very small, and for UWB applications, potentially the most difficult optimization parameter. The fitness function

for the optimization routines was defined as

$$f(\mathbf{x}) = \sum_{n=0}^{N-1} (S_{11}(\mathbf{x})|_{f_n} < -10\text{dB}) \tag{6.1}$$

where $\mathbf{x}$ is a binary string representing a patch geometry, $S_{11}(\mathbf{x})$ is the return loss for the particular geometry, f_n is the nth frequency in the range 3.1 GHz to 10.6 GHz, and N is the number of frequency points in that interval. The sum is for $S_{11} < -10dB$ because that is the value below which an antenna is generally consdered to be resonant. The bandwidth is then the difference between the frequencies for which S_{11} is below -10 dB.

The return loss, S_{11} is evaluated using the FDTD method, as described in Chapter 3. Initially, a pulse is sent down the antenna feedline without the patch antenna present. The electric field z–component $E_{z,inc}$ is recorded for every time step at some location near the entry point to the microstrip feed line. Then, a patch geometry is generated from a binary string representing a potential optimal geometry. A pulse is again sent down the feedline and $E_{z,total}$ is recorded at the same location. The reflected electric field is $E_{z,ref} = E_{z,total} - E_{z,inc}$. Fourier transforms are taken of the incident field and reflected field, and

$$S_{11}(f) = -20 \log \left(\frac{E_{z,ref}(f)}{E_{z,inc}(f)} \right) \text{dB} \tag{6.2}$$

The entire process repeats for each trial solution, with the exception of the calculation of the incident pulse. This needs only happen once, since it is the same for all possible solutions. New solutions are generated according to the rules of the optimization method.

6.2 Results of Computer Simulation and Optimization

The optimization process begins by first setting the FDTD paramters. In this case, the cell discretization was $\Delta_{xyz} = 0.132$ mm, and $\Delta t = 0.22$ ps. Thus, the total number of cells along each axis was $X = 128$, $Y = 256$, and $Z = 28$. However, the z–direction was then zero padded such that $Z = 32$, to ensure proper computation using the GPU. Thus, the total number of cells was $128 \times 256 \times 32 = 1048576$.

The incident pulse,

$$E(t) = -\left(\frac{t - t_0}{\sigma^2}\right) e^{-\frac{1}{2}(t-t_0)^2/\sigma^2} \tag{6.3}$$

has $t_0 = 400\Delta t$ and $\sigma = 100\Delta t$. To map a trial solution to the FDTD grid, each bit represented a pixelated section of the patch antenna. On the FDTD grid each pixel contained 25 grid points, to reduce numerical inaccuracies. Simulations were run for 10000 timesteps to ensure that the total energy in the grid had reached a steady state minimum.

6.2.1 Genetic Algorithm Optimization

For optimization with a GA several parameters were chosen. First, the population size was 30, and the number of bits/variables was 120. Since the antenna geometries were represented directly by binary numbers, no decoding of the bit strings to real decimal numbers was necessary. An elitism rate of 0.45 was used, along with a 0.005 mutation rate. Roulette–wheel selection and double–point crossover were used. The optimization was run for 200 iterations.

Running on an AMD Radeon HD 6450 GPU, a single fitness function evaluation took approximately 1.25 min, and an entire GA optimization run took about 5 days. Three GA runs were completed and the best result

is presented. The normalized convergence of the GA is shown in Figure 6.2, the optimized antenna geometry is shown in Figure 6.3, and the antenna S_{11} is shown in Figure 6.4.

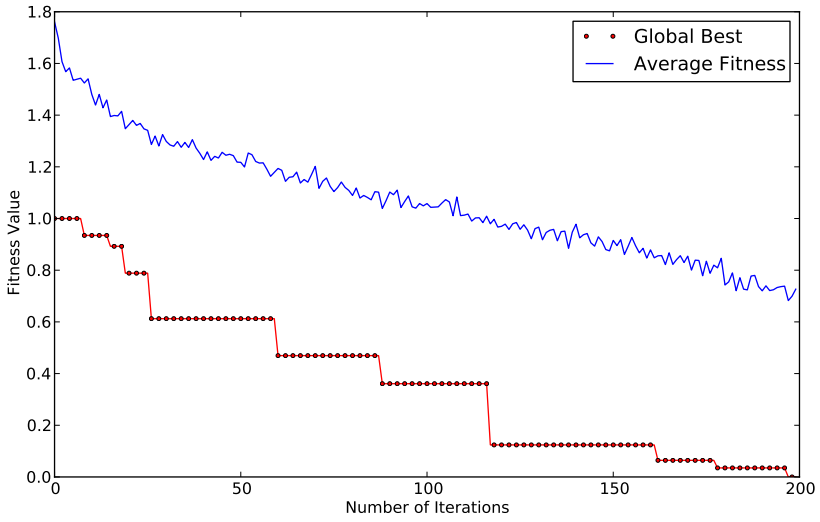


Figure 6.2: GA convergence

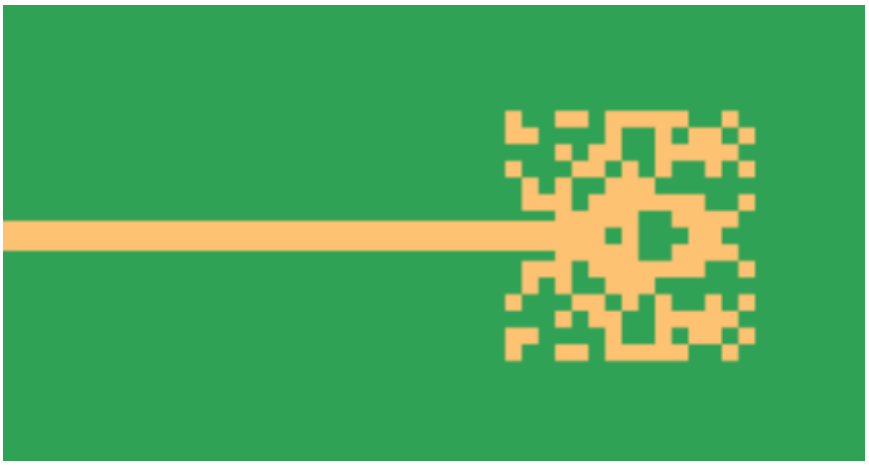


Figure 6.3: GA optimized UWB patch antenna

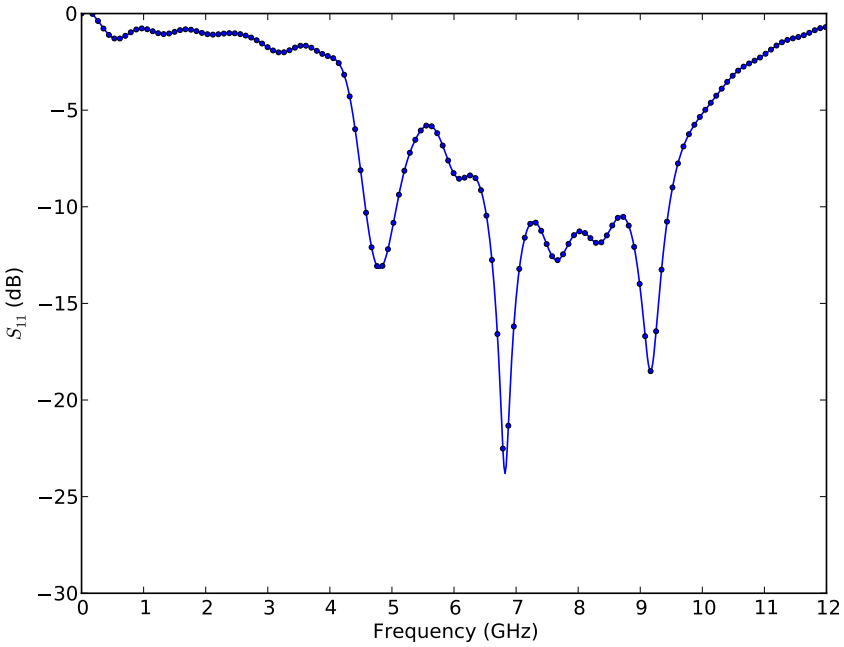


Figure 6.4: Return loss for GA optimized patch

The fitness of the GA steadily converged to the final design, as shown in Figure 6.2. From Figure 6.4 it can be seen that where S_{11} drops below -10 dB, the antenna bandwidth does not cover the entire range of 3.1 GHz – 10.6 GHz, but still covers a large portion, from about 6.5 GHz – 9.5 GHz, as well as a lower band around 4.5 GHz – 5 GHz.

It is not always necessary for an UWB antenna to cover the entire 3.1 GHz - 10.6 GHz frequency band. UWB systems that utilize multiband orthogonal frequency division multiplexing (MB-OFDM) divide the 3.1 GHz - 10.6 GHz spectrum into nonoverlapping subbands of 528 MHz each. These subbands are then grouped into 5 band groups, and UWB applications may only band groups. For example, in an MB-OFDM system band group #4 comprises the frequency band 6.6 GHz - 7.7 GHz and band group #5 comprises 8.2 GHz - 9.24 GHz [38]. The GA designed antenna which covers the 6.5 GHz - 9.5 GHz range could therefore find applications in an MB-OFDM system in which devices were using band groups #4 and #5.

6.2.2 Particle Swarm Optimization

As with the GA, the population size was 30, and the number of bits/variables was 120. For the PSO, the social parameter was set to 1.4, the cognitive paramter was 1.6, and the inertial weight was 0.65. The optimization was run for 200 iterations, as before, with the best results of 3 runs presented.

The normalized convergence of the PSO is shown in Figure 6.5, the optimized antenna geometry is shown in Figure 6.6, and the antenna S_{11} is shown in Figure 6.7.

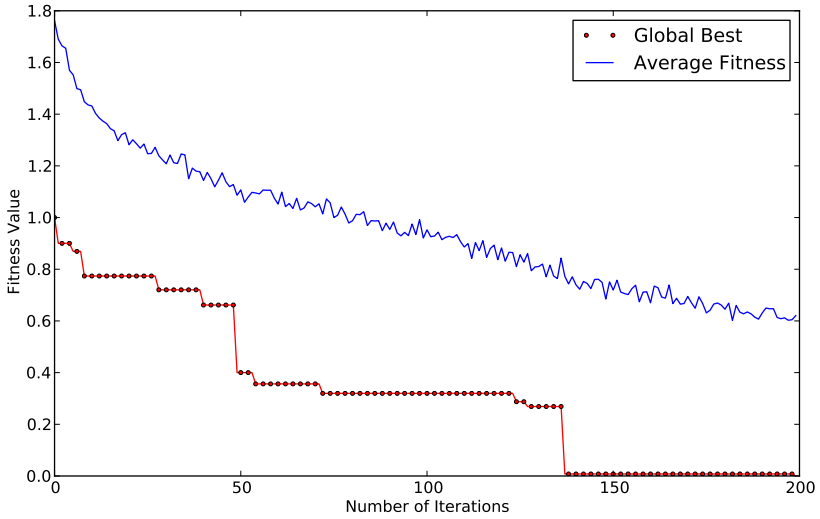


Figure 6.5: PSO convergence

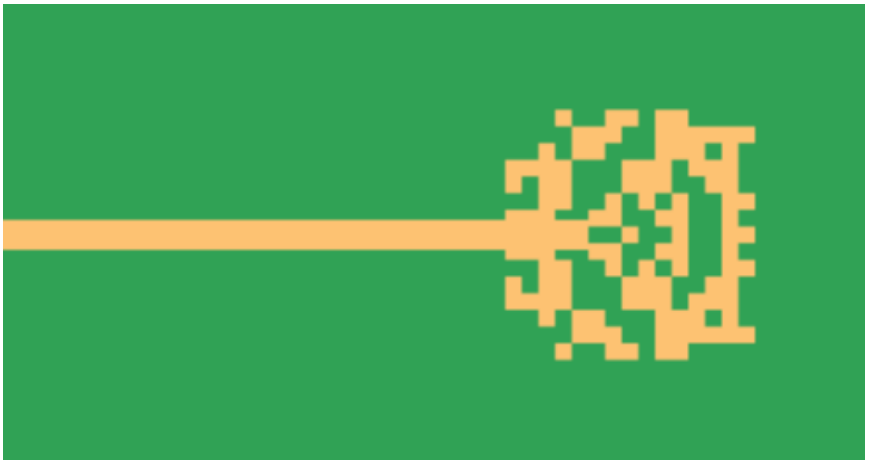


Figure 6.6: PSO optimized UWB patch antenna

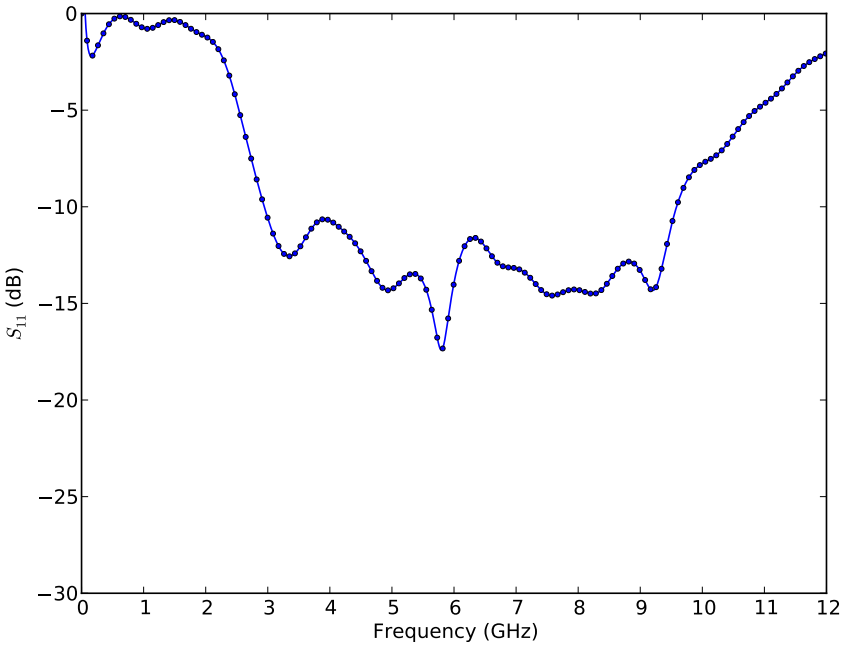


Figure 6.7: Return loss for PSO optimized patch

Unlike the GA which steadily improved its fitness levels, PSO improved in two large jumps–after 50 iterations and again after 137 iterations–as shown in Figure 6.5. From Figure 6.7 it can be observed that the bandwidth of the antenna designed covers most of the 3.1 GHz to 10.6 GHz UWB range, specifically from 2.94 GHz – 9.57 GHz. In terms of total bandwidth PSO outperformed the GA with a bandwidth of 6.63 GHz versus 2.95 GHz–more than double.

6.2.3 Biogeography–Based Optimization

The BBO parameters were once again a population of 30, with 120 variables, the elitism rate was 0.38, the mutation rate was 0.008, and the number of iterations was 200.

After 3 optimizations over the course of 15 days, the normalized convergence of the best result of BBO is shown in Figure 6.8, the optimized antenna geometry is shown in Figure 6.9, and the antenna S_{11} is shown in Figure 6.10.

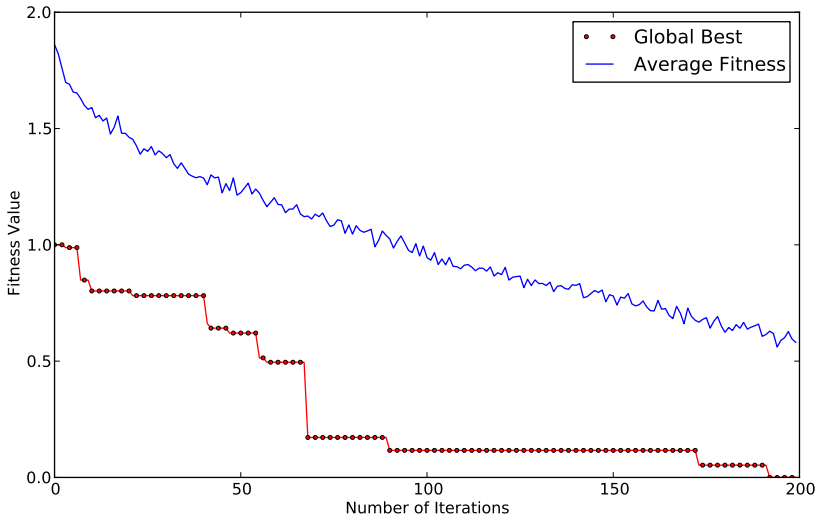


Figure 6.8: BBO convergence

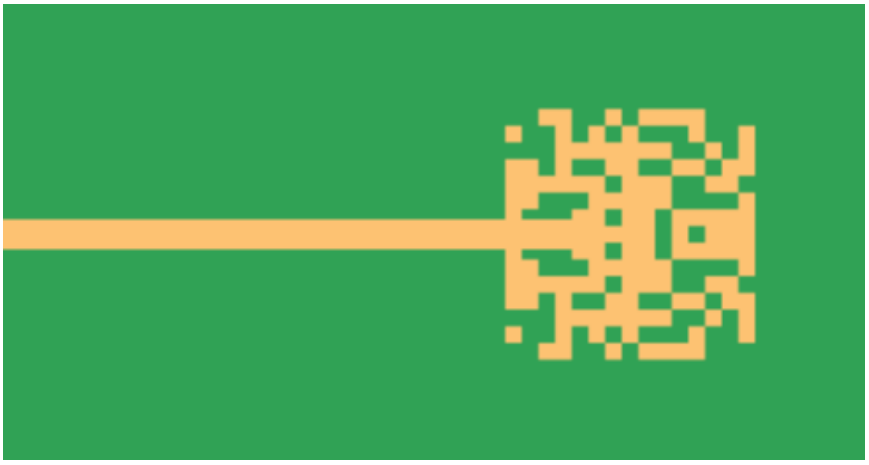


Figure 6.9: BBO optimized UWB patch antenna

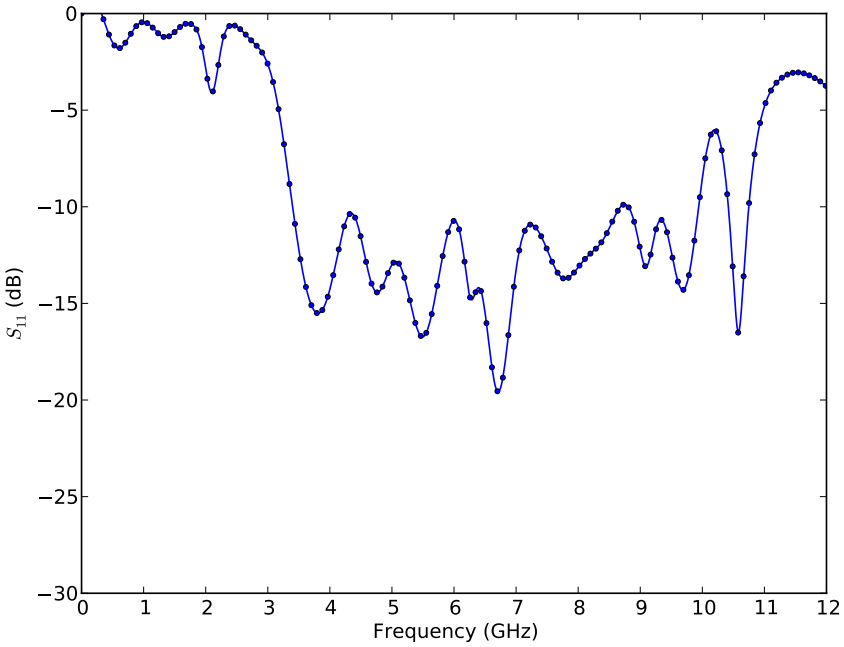


Figure 6.10: Return loss for BBO optimized patch

Like PSO the BBO design covers most of the UWB range from about 3.4 GHz to 9.92 GHz, with a separate 300 MHz peak centered at 10.6 GHz. The total bandwidth of the BBO design was 6.52 GHz, slighly less than the PSO design and more than twice the GA optimized design.

6.2.4 Discussion of Results

The results of the optimization processes were very favorable, with each of the methods producing designs that could potentially be used in UWB systems. PSO and BBO found antenna configurations with bandwidths of over 6.5 GHz that covered most of the UWB range, and therefore could be used for a variety of UWB applications. The GA, however, produced a design with a smaller bandwidth of approximately 3 GHz which could be used in an MB-OFDM USB system. It should be noted that these results do not necessarily imply that PSO and BBO are superior to the GA for this application, due to the effects of parameter variation. Therefore it is quite possible that the GA could find other optimal geometries given

certain parameters settings, number of iterations, etc.

Typical wireless system antenna design begins with a standard design which is then modified through educated guesswork, trial and error, or by accident to arrive at a suitable design for a given system [100]. This is a time-consuming process, the results of which cannot be guaranteed to meet fabrication or manufacturing constraints. Additionally, a design methodology successful in one particular situation cannot necessarily be reused in another design.

Bio-inspired evolutionary optimization procedures can overcome most of these difficulties, as the results show. Constraints are built into the optimization process, and the same process can be used for differenct problems. It is only a matter of redefining the goals and constraints. The results are nonintuitive designs that still result in excellent performance.

However, these optimization algorithms can also be time-consuming which potentially outweighs the benefits. On a single Intel Core i5-2320 3.0 GHz CPU, a fitness function took nearly 26 min, and with the population sizes used would have taken nearly 108 days to complete a single optimized design. Clearly this is not acceptable, since it is possible that several optimization runs maybe required.

Parallel processing using a GPU can significantly speedup the time for computation, however, without the need for a large investment in hardware. As was shown an AMD Radeon HD 6450, a low-end GPU costing less than $40, provided enough computational ability to speedup the process so that it could complete in 5 days instead of 108. With increased computational power further speedups could be realized, resulting in a very practical and relatively simple design method that can be applied in potentially any situation to produce novel designs in an evolutionary way.

CHAPTER VII

CONCLUDING REMARKS AND FUTURE POSSIBILITIES

7.1 Conclusion

This study has shown the benefits of using bio-inspired optimization for the development of UWB antennas, in particular BBO, which has not been applied to this type of problem before. A typical patch antenna has a very narrow bandwidth, and traditional geometries must be modified to achieve the bandwidth needed for UWB applications. Usually this must be done through a process of trial and error, where a basic design is created and then incrementally modified. This process can be very time consuming and potentially expensive, however. By using optimization algorithms such as a GA, PSO, or BBO that makes no assumptions about the design, very unorthodox geometries can be generated to meet certain design criteria.

Although bio-inspired optimization can produce effective results that are very unlike typical design methods, they are usually very time consuming as well. Especially for antenna or microwave designs where each potential design must be evaluted with a three dimensional computational electromagnetic solver. However, through effective use of GPU programming, some of the time pitfalls can be alleviated. Specifically, it was shown by the development of a GPU-accelerated FDTD method coupled with an evolutionary optimizer, the time for an optimized result can be reduced

substatially.

The overall time for an optimized result was about 5 days, which is about a 20x improvement over what could be expected from a single CPU performing the same computations. It is possible to achieve a similar speed up by using parallel CPUs, however that increases complexity and would require a dedicated system. By utilizing a GPU, which most machines come equipped with, no additional special hardware is required and complexity is lessened.

Each optimizer considered performed about equally well in generating an antenna geometry that can be used for UWB applications. In particular, the sizes of the patch antennas were constrained to 10 mm x 10 mm, a size that could be fit inside a USB dongle, for instance. This is another contribution, since patch antennas for UWB applications are typically larger. In summary, bio-inspired optimization methods can be very useful for generating nontraditional antenna geometries that perform in applications where a typical design would fail, but is only really practical when used with a method for accelerating computations. GPU acceleration has great potential for designers, since they are available on most systems without modification.

7.2 Future Work

The most immediate extension to this study would be a multi-objective evolutionary optimization (MOEA). In MOEA, more than one fitness function is defined, and the optimization algorithm seeks to find solutions that give the best fitness for each simultaneously. Alternatively, a single fitness function can be defined that is a weighted sum of the individual fitness functions.

Several variables are part of a complete UWB antenna design, not only bandwidth, but also gain stability over the 3.1-10.6 GHz range, as well as

phase linearity, and polarization, for example. In an MOEA, however, it is not always possible to find a solution that is optimal for all variables simultaneously. Instead, a distribution of solutions are found, where each solution is more or less optimal for a particular variable.

Another possible extension is regarding patch antenna miniturization. Instead of fixing the lengths and width of the patch and feedline, each of those parameters could also be variable, but with upper bounds for design constraints. Then it would be a matter of scaling the pixelating process accordingly.

Finally, it would be worth looking into the potential of multi GPU-accelerated optimization using GPU clustering. A typical motherboard and power supply can accomodate up to 4 GPUs. In the single GPU case, the fitness function (FDTD) is processed in parallel, but each potential solution is processed serially. Using multiple GPUs, trial solutions could be processed simultaneously. Assuming linear speedup, the 5 days needed to complete an optimization could be reduced to slighly over 1 day.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, no. 9, pp. 102–114, August 2002.

[2] A. Molisch, J. Foerster, and M. Pendergrass, "Channel Models for Ultrawideband Personal Area Networks," *IEEE Wireless Communications*, vol. 10, no. 6, pp. 14–21, 2003.

[3] D. Smith, D. Miniutti, T. Lamahewa, and L. Hanlen, "Propagation Models for Body–Area Networks: A Survey and New Outlook," *IEEE Antennas and Propagation Magazine*, vol. 55, no. 5, pp. 97–117, 2013.

[4] N. Niebert, A. Schieder, H. Abramowicz, G. Malmgren, J. Sachs, U. Horn, C. Prehofer, and H. Karl, "Ambient Networks: An Architecture for Communications Networks Beyond 3G," *IEEE Wireless Communications*, vol. 11, no. 2, pp. 14–22, April 2004.

[5] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," *IEEE Personal Communications*, vol. 8, no. 4, pp. 10–17, August 2001.

[6] C. Park and T. Rappaport, "Short–Range Communications for Next–Generation Networks: UWB, 60 GHz, Millimeter–Wave WPAN, and ZigBee," *IEEE Wireless Communications*, vol. 14, no. 4, pp. 70–78, August 2007.

[7] I.-H. Kim, H.-J. Kim, J.-T. Ihm, G.-M. Jeong, and K.-D. Chung, "WPAN Platform Architecture and Application Design for Handset," *2008 International Conference on Consumer Electronics (ICCE 2008)*, pp. 1–2, January 2008.

[8] K. Kang, D. Kang, K. Ha, and J. Lee, "Android Phone as Wireless USB Storage Device Through USB/IP Connection," *2011 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 289–290, January 2011.

[9] W. Jones, "No Strings Attached," *IEEE Spectrum*, vol. 43, no. 4, pp. 16–18, April 2006.

[10] C. Shannon, "A Mathematical Theory of Communication," *Bell Systems Technology Journal*, vol. 27, pp. 379–423 and 623–656, July and October 1948.

[11] J. E. Nuechterlein and P. J. Weiser, *Digital Crossroads: Americal Telecommunications Policy in the Internet Age*. MIT Press, 2005.

[12] G. R. Aiello and G. D. Rogerson, "Ultra-Wideband Wireless Systems," *IEEE Microwave Magazine*, pp. 36–47, June 2003.

[13] L. Yang and G. B. Giannakis, "Ultra-Wideband Communications, An Idea Whose Time Has Come," *IEEE Signal Processing Magazine*, pp. 26–54, November 2004.

[14] R. Rashid and R. Yusoff, "Bluetooth Performance Analysis in Personal Area Network (PAN)," *2006 International RF and Microwave Conference*, pp. 393–397, 2006.

[15] H. Lau, "High-Speed Wireless Personal Area Networks: An Application of UWB Technologies," in *Novel Applications of the UWB Technologies*, D. B. Lembrikov, Ed. InTech, 2011.

[16] H. G. Shantz, "Three Centuries of UWB Antenna Development," *ICUWB*, pp. 506–512, 2012.

[17] X. Jiang, S. Li, and G. Su, "Broadband Planar Antenna With Parasitic Radiator," *Electronics Letters*, vol. 39, no. 23, 2003.

[18] X. Qiu, H. Chiu, and A. Mohan, "Symmetrically Beveled Ultra-Wideband Planar Monopole Antenna," *2005 IEEE Antennas and Propagation Society International Symposium*, vol. 2A, pp. 504–507, 2005.

[19] E. Antonino-Daviu, M. Cabedo-Fabres, M. Ferrando-Bataller, and A. Valero-Nogueira, "Wideband Double-Fed Planar Monopole Antennas," *Electronics Letters*, vol. 39, no. 23, November 2003.

[20] D. Guha and Y. Antar, "Circular Microstrip Patch Loaded With Balanced Shorting Pins for Improved Bandwidth," *IEEE Antennas and Wireless Propagation Letters*, vol. 5, no. 1, pp. 217–219, 2006.

[21] V. Sadeghi, C. Ghobadi, and J. Nourinia, "Design of UWB Semi-Circle-Like Slot Antenna With Controllable Band-Notch Function," *Electronics Letters*, vol. 45, no. 25, pp. 1282–1283, 2009.

[22] Z. Low, J. Cheong, and C. Law, "Low-Cost PCB Antenna for UWB Applications," *IEEE Antennas and Wireless Propagation Letters*, vol. 4, pp. 237–239, 2005.

[23] J. Liang, C. C. Chiau, X. Chen, and C. Parini, "Study of Printed Circular Disc Monopole Antenna for UWB Systems," *IEEE Transactions on Antennas and Propagation*, vol. 53, no. 11, pp. 3500–3504, November 2005.

[24] Y. Lin and K. Hung, "Compact Ultrawideband Rectangular Aperture Antenna and Band-Notched Designs," *IEEE Transactions on*

Antennas and Propagation, vol. 54, no. 11, pp. 3075–3081, November 2006.

[25] Y. Cho, K. Kim, D. Choi, S. Lee, and S. Park, "A Miniture UWB Planar Monopole Antenna With 5-GHz Band-Rejection Filter and the Time-Domain Characteristics," *IEEE Transactions on Antennas and Propagation*, vol. 54, no. 5, pp. 1453–1460, May 2006.

[26] P. Li, J. Liang, and X. Chen, "Study of Printed Elliptical/Circular Slot Antennas for Ultrawideband Applications," *IEEE Transactions on Antennas and Propagation*, vol. 54, no. 6, pp. 1670–1675, June 2006.

[27] Z. Chen, T. See, and X. Qing, "Small Printed Ultrawideband Antenna With Reduced Ground Plane Effect," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 2, pp. 383–388, February 2007.

[28] A. Abbosh and M. Bialkowski, "Design of Ultrawideband Planar Monopole Antennas of Circular and Elliptical Shape," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 1, January 2008.

[29] K. G. Thomas and M. Sreenivasan, "A Simple Ultrawideband Planar Rectangular Printed Antenna With Band Dispensation," *IEEE Transactions on Antennas and Propagation*, vol. 58, no. 1, January 2010.

[30] R. Azadegan and K. Sarabandi, "A Novel Approach for Miniaturization of Slot Antennas," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 3, pp. 421–429, 2003.

[31] B. Porter, G. Noakes, and S. Gearhart, "Design on Dual-Band Dual-Polarized Wire Antennas Using a Genetic Algorithm," *IEEE Anten-*

nas and Propagation Society International Symposium, 1999, vol. 4, pp. 2706–2709, 1999.

[32] G. Zhao, W. Shen, and M. Wu, "Monopole Antenna Design Using a Genetic Algorithm With Dual-Band and Widebad Operations," *2008 China-Japan Joint Microwave Conference*, pp. 241–244, 2008.

[33] L. Lizzi, F. Viani, R. Azaro, and A. Massa, "A PSO-Driven Spline-Based Shaping Approach for Ultrawideband (UWB) Antenna Synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 56, no. 8, pp. 2613–2621, 2008.

[34] M. John and M. Ammann, "Wideband Printed Monopole Design Using a Genetic Algorithm," *IEEE Antennas and Wireless Propagation Letters*, vol. 6, 2007.

[35] H. G. Shantz, "A Brief History of Ultra-Wideband Antennas," *IEEE Conference on UWBST*, pp. 209–213, November 2003.

[36] T. Barrett, "History of Ultra Wideband (UWB) Radar and Communications: Pioneers and Innovators," *Proceedings of Progress in Electromagnetics Symposium 2000(PIERS2000)*, July 2000.

[37] C. Bennett and G. F. Ross, "Time-Domain Electromagnetics and Its Applications," *Proceedings of the IEEE*, vol. 66, no. 3, March 1978.

[38] R. Aiello and A. Barta, *Ultra Wideband Systems: Technologies and Applications.* Elsevier, 2006.

[39] *First Report and Order, Revision of Part 15 of the Commission's Rules Regarding Ultra-Wideband Transmission Systems*, FCC, February 2002.

[40] R. J. Fontana, "Recent System Applications of Short-Pulse Ultra-Wideband (UWB) Technology," *IEEE Transactions on Microwave Theory and Applications*, vol. 52, no. 9, September 2004.

[41] I. Immoreev and D. Fedotov, "Ultra Wideband Radar Systems: Advantages and Disadvantages," *Proceedings of the IEEE Ultra Wideband Systems and Technology Conference*, May 2002.

[42] M. Ho, L. Taylor, and G. Aiello, "UWB Technology for Wireless Video Networking," *ICCE. International Conference on Consumer Electronics*, 2001.

[43] J. Kim, S. Lee, Y. Jeon, and S. Choi, "Residential HDTV Distribution System Using UWB and IEEE 1394," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 1, 2006.

[44] Z. Jinyun, P. Orlik, Z. Sahinoglu, and A. Molisch, "UWB Systems for Wireless Sensor Networks," *Proceedings of the IEEE*, vol. 97, no. 2, 2009.

[45] S. Gezici, T. Zhi, G. Giannakis, H. Kobayashi, A. Molisch, H. Poor, and Z. Sahinoglu, "Localization Via Ultra-Wideband Radio: a Look At Positioning Aspects for Future Sensor Networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, 2005.

[46] V. Gungor and G. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, 2009.

[47] D. Dardari, A. Conti, U. Ferner, A. Giorgetti, and M. Win, "Ranging with Ultrawide Bandwidth Signals in Multipath Environments," *Proceedings of the IEEE*, vol. 97, no. 2, 2009.

[48] S. Dai, L. Liu, and G. Fang, "A Low–Cost Handheld Integrated UWB Radar For Shallow Underground Detection," *2010 IEEE International Conference on Ultra–Wideband (ICUWB)*, pp. 1–4, Sept. 2010.

[49] G. Ji, X. Gao, H. Zhang, and T. Gulliver, "Subsurface Object Detection Using UWB Ground Penetrating Radar," *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*, pp. 740–743, August 2009.

[50] M. H. Ucar, A. Sondas, and Y. E. Erdemli, "Design of 2 × 2 UWB Printed Antenna Array for See-Through-Wall Imaging," *2013 Computational Electromagnetics Workshop (CEM)*, pp. 26–27, 2013.

[51] P. Hansen, K. Scheff, and E. Mokole, "Dual Polarized, UWB Radar Measurements of the Sea at 9 GHz," *1998 Ultra–Wideband Short–Pulse Electromagnetics 4*, pp. 335–348, June 1998.

[52] M. Levy, D. Kumar, and A. Dinh, "A Novel Fractal UWB Antenna for Earthquake and Tsunami Prediction Application (LETPA)," *2013 26th Annual IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4, May 2013.

[53] E. Pancera, L. Xuyang, M. Jalilvand, and T. Zwick, "UWB Medical Diagnostics: In–Body Transmission Modeling and Applications," *Proceedings of the 5th European Conference on Antennas and Propagation (EUCAP)*, pp. 2651–2655, April 2011.

[54] R. Garg, P. Bhartia, I. Bahl, and A. Ittipiboon, *Microstrip Antenna Design Handbook*. Artech House, 2001.

[55] D. M. Pozar and B. Kaufman, "Increasing the Bandwidth of a Microstrip Antenna by Proximity Coupling," *Electronics Letters*, vol. 23, no. 8, pp. 368–369, 1987.

[56] J. James, P. Hall, and C. Wood, *Microstrip Antenna Theory and Design.* Peter Peregrinus Ltd, 1981.

[57] C. A. Balanis, *Antenna Theory: Analysis and Design.* John Wiley and Sons, Inc, 2005.

[58] G. Kumar and K. Ray, *Broadband Microstrip Antennas.* Artech House, 2003.

[59] Y. Jang, "Broadband T-shaped Microtrip–Fed U-slot Coupled Patch Antenna," *Electronics Letters*, vol. 38, no. 11, pp. 495–496, May 2002.

[60] J. Ansari, N. Yadav, A. Mishra, K. Singh, and A. Singh, "Broadband Rectangular Microstrip Antenna Loaded With a Pair of U-Shaped Slots," *2010 International Conference on Power, Control and Embedded Systems (ICPCES)*, pp. 1–5, Nov. 2010.

[61] A. Deshmukh and G. Kumar, "Compact Broadband E-shaped Microstrip Antennas," *Electronics Letters*, vol. 41, no. 18, pp. 989–990, Sept. 2005.

[62] J. Kovitz and Y. Rahmat-Samii, "Micro–Actuated Pixel Patch Antenna Design Using Particle Swarm Optimization," *2011 IEEE Symposium on Antennas and Propagation*, pp. 2415–2418, July 2011.

[63] H. Choo and H. Ling, "Design of Multiband Microstrip Antennas Using a Genetic Algorithm," *IEEE Microwave and Wireless Components Letters*, vol. 12, no. 9, pp. 345–347, Sept. 2002.

[64] F. Villegas, T. Cwik, Y. Rahmat-Samii, and M. Manteghi, "Parallel Genetic–Algorithm Optimization of a Dual–Band Patch Antenna for Wireless Communications," *2002 Antennas and Propagation Society International Symposium*, pp. 334–337, 2002.

[65] Y. Rahmat-Samii, J. M. Kovitz, and H. Rajagopalan, "Nature–Inspired Optimization Techniques in Communication Antenna Designs," *Proceedings of the IEEE*, vol. 100, no. 7, pp. 2132–2144, July 2012.

[66] K. Yee, "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations In Isotropic Media," *IEEE Transactions on Antennas and Propagation*, vol. 14, pp. 302–307, 1966.

[67] J. Berenger, "A Perfectly Matched Layer for the Absorption of Electromagnetic Waves," *Journal of Computational Physics*, vol. 114, pp. 185–200, 1994.

[68] ——, "Three-Dimensional Perfectly Matched Layer for the Absorption of Electromagnetic Waves," *Journal of Computational Physics*, vol. 127, pp. 363–379, 1996.

[69] A. Taflove and S. C. Hagness, *Computational Electrodynamics: The Finite-Difference Time-Domain*, 2nd ed. Artech House, 2000.

[70] Z. Sacks, D. Kingsland, R. Lee, and J. Lee, "A Perfectly Matched Anisotropic Absorber for Use as an Absorbing Boundary Condition," *IEEE Transactions on Antennas and Propagation*, vol. 7, pp. 1460–1463, December 1995.

[71] D. M. Sullivan, *Electromagnetic Simulation Using the FDTD Method*. IEEE Press, 2000.

[72] D. Sheen, S. Ali, M. Abouzahra, and J. Kong, "Application of the Three-Dimensional Finite-Difference Time-Domain Method to the Analysis of Planar Microstrip Circuits," *IEEE Transactions on Microwave Theory and Techniques*, vol. 38, no. 7, pp. 849–857, July 1990.

[73] W. Bozejko, A. Dobrucki, and M. Walczynski, "Parallelizing of Digital Signal Processing With Using GPU," *Signal Processing Algorithms, Architectures, Arrangements, and Applications Conference Proceedings (SPA)*, pp. 29–33, Sept 2010.

[74] N. Zhang, Y. shan Chen, and J.-L. Wang, "Image Parallel Processing Based on GPU," *2010 2nd International Conference on Advanced Computer Control (ICACC)*, March 2010.

[75] W. Vanderbauwhede and T. Takemi, "An Investigation Into the Feasibility and Benefits of GPU/Multicore Acceleration of the Weather Research and Forecasting Model," *2013 International Conference on High Performance Computing and Simulation (HPCS)*, July 2013.

[76] C. Yang, Q. Wu, J. Chen, and Z. Ge, "GPU Acceleration of High-Speed Collision Molecular Dynamics Simulation," *Ninth IEEE International Conference on Computer and Information Technology*, Oct. 2009.

[77] A. Bustamam, K. Burrage, and N. Hamilton, "Fast Parallel Markov Clustering in Bioinformatics Using Massively Parallel Graphics Processing Unit Computing," *2010 Ninth International Workshop on Parallel and Distributed Methods in Verification*, Sept. 2010.

[78] D. Barboza and E. Clua, "GPU-Based Data Structure for a Parallel Ray Tracing Illumination Algorithm," *2011 Brazilian Symposium on Games and Digital Entertainment (SBGAMES)*, pp. 11–16, Nov. 2011.

[79] C. Y. Ong, M. Weldon, S. Quiring, L. Maxwell, M. Hughes, C. Whelan, and M. Okoniewski, "Speed It Up," *IEEE Microwave Magazine*, pp. 70–78, April 2010.

[80] Z. Bo, X. Zheng-hui, R. Wu, L. Wie-ming, and S. Xin-qing, "Accelerating FDTD Algorithm Using GPU Computing," *2011 IEEE International Conference on Microwave Technology and Computational Electromagnetics (ICMTCE)*, pp. 410–413, May 2011.

[81] S. E. Krakiwsky, L. E. Turner, and M. M. Okoniewski, "Graphics Processor Unit (GPU) Acceleration of Finite-Difference Time-Domain (FDTD) Algorithm," *Proceedings of the 2004 International Symposium on Circuits and Systems*, May 2004.

[82] J. Hensley, "OpenCL Specification Overview." SIGGRAPH Asia 2009, 2009.

[83] M. Scarpino, *OpenCL in Action*. Manning Publications Co., 2012.

[84] S. Gedney, "Finite–Difference Time–Domain Analysis of Microwave Circuit Devices on High Performance Vector/Parallel Computers," *IEEE Transactions on Microwave Theory and Techniques*, vol. 43, no. 10, pp. 2510–2514, October 1995.

[85] W. Yu, Y. Liu, Z. Su, N.-T. Hunag, and R. Mittra, "A Robust Parallel Conformal Finite–Difference Time–Domain Processing Package Using the MPI Library," *IEEE Antennas and Propagation Magazine*, vol. 47, no. 3, pp. 39–59, June 2005.

[86] Y. Lu and C. Chen, "A Domain Decomposition Finite–Difference Method for Parallel Numerical Implementation of Time–Dependent Maxwell's Equations," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 3, pp. 556–562, March 1997.

[87] C. Guiffaut and K. Mahdjoubi, "A Parallel FDTD Algorithm Using the MPI Library," *IEEE Antennas and Propagation Magazine*, vol. 43, no. 2, pp. 94–103, April 2001.

[88] P. Sypek, A. Dziekonski, and M. Mrozowski, "How to Render FDTD Computations More Effective Using a Graphics Accelerator," *IEEE Transactions on Magnetics*, vol. 45, no. 3, pp. 1324–1327, 2009.

[89] S. Adams, J. Payne, and R. Boppana, "Finite Difference Time Domain (FDTD) Simulations Using Graphics Processors," *2007 DoD High Performance Computing Modernization Program Users Group Conference*, pp. 334–338, 2007.

[90] "Ncbi map viewer," http://www.ncbi.nlm.nih.gov/projects/mapview, accessed: 01-18-2014.

[91] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st ed. Addison-Weasley Professional, 1989.

[92] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[93] J. Baker, "Reducing Bias and Inefficiency in the Selection Algorithm," *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, pp. 14 – 21, 1987.

[94] S. Sivanandam and S. Deepa, *Introduction to Genetic Algorithms*. Springer, 2008.

[95] Eberhart and Kennedy, "Particle Swarm Optimization," *IEEE International Conference on Neural Networks*, 1995.

[96] J. Kennedy and R. Eberhart, "A Discrete Binary Version of the Particle Swarm Optimization," *IEEE International Conference on Systems, Man, and Cybernetics*, 1997.

[97] R. MacArthur and E. O. Wilson, *The Theory of Island Biogeography*. Princeton University Press, 1967.

[98] D. Simon, "Biogeography–Based Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702 – 713, December 2008.

[99] X. Yao, Y. Liu, and G. Lin, "Evolutionary Programming Made Faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, July 1999.

[100] J. M. Johnson and Y. Rahmat-Samii, "Genetic Algorithm and Method of Moments (GA/MOM) for the Design of Integrated Antennas," *IEEE Transactions on Antennas and Propagation*, vol. 47, no. 10, pp. 1606–1614, October 1999.

APPENDIX A

FINITE DIFFERENCE TIME DOMAIN UPDATE EQUATIONS

This appendix contains the full listing of the electromagnetic field and constitutive field update equations, as derived based on the discussion in Section 3.3. The examples from that section are repeated for completeness.

A.1 D-field Update

$$D_x\big|_{i,j+1/2,k+1/2}^{n+1/2} = \left(\frac{2\epsilon\kappa_y - \sigma_y\Delta t}{2\epsilon\kappa_y + \sigma_y\Delta t}\right) D_x\big|_{i,j+1/2,k+1/2}^{n-1/2} + \left(\frac{2\epsilon\Delta t}{2\epsilon\kappa_y + \sigma_y\Delta_t}\right)\times$$

$$\left(\frac{H_z\big|_{i,j+1,k+1/2}^{n} - H_z\big|_{i,j,k+1/2}^{n}}{\Delta y} - \frac{H_y\big|_{i,j+1/2,k+1}^{n} - H_y\big|_{i,j+1/2,k}^{n}}{\Delta z}\right) \tag{A.1}$$

$$D_y\big|_{i,j+1/2,k+1/2}^{n+1/2} = \left(\frac{2\epsilon\kappa_z - \sigma_z\Delta t}{2\epsilon\kappa_z + \sigma_z\Delta t}\right) D_y\big|_{i,j+1/2,k+1/2}^{n-1/2} + \left(\frac{2\epsilon\Delta t}{2\epsilon\kappa_z + \sigma_z\Delta_t}\right)\times$$

$$\left(\frac{H_x\big|_{i,j+1/2,k+1}^{n} - H_x\big|_{i,j+1/2,k}^{n}}{\Delta z} - \frac{H_z\big|_{i+1/2,j+1/2,k+1/2}^{n} - H_z\big|_{i-1/2,j+1/2,k+1/2}^{n}}{\Delta x}\right) \tag{A.2}$$

$$D_z\big|_{i,j+1/2,k+1/2}^{n+1/2} = \left(\frac{2\epsilon\kappa_x - \sigma_x\Delta t}{2\epsilon\kappa_x + \sigma_x\Delta t}\right) D_z\big|_{i,j+1/2,k+1/2}^{n-1/2} + \left(\frac{2\epsilon\Delta t}{2\epsilon\kappa_x + \sigma_x\Delta_t}\right)\times$$

$$\left(\frac{H_y\big|_{i+1/2,j+1/2,k+1/2}^{n} - H_y\big|_{i-1/2,j+1/2,k+1/2}^{n}}{\Delta x} - \frac{H_x\big|_{i,j+1,k+1/2}^{n} - H_x\big|_{i,j,k+1/2}^{n}}{\Delta y}\right)$$

(A.3)

A.2 E-field Update

$$E_x\big|_{i,j+1/2,k+1/2}^{n+1/2} = \left(\frac{2\epsilon\kappa_z - \sigma_z\Delta t}{2\epsilon\kappa_z + \sigma_z\Delta t}\right) E_x\big|_{i,j+1/2,k+1/2}^{n-1/2} +$$

$$\left[\frac{2\epsilon\kappa_x + \sigma_x\Delta t}{(2\epsilon\kappa_z + \sigma_z\Delta t)\epsilon}\right] D_x\big|_{i,j+1/2,k+1/2}^{n+1/2} - \left[\frac{2\epsilon\kappa_x - \sigma_x\Delta t}{(2\epsilon\kappa_z + \sigma_z\Delta t)\epsilon}\right] D_x\big|_{i,j+1/2,k+1/2}^{n-1/2}$$

(A.4)

$$E_y\big|_{i,j+1/2,k+1/2}^{n+1/2} = \left(\frac{2\epsilon\kappa_x - \sigma_x\Delta t}{2\epsilon\kappa_x + \sigma_x\Delta t}\right) E_y\big|_{i,j+1/2,k+1/2}^{n-1/2} +$$

$$\left[\frac{2\epsilon\kappa_y + \sigma_y\Delta t}{(2\epsilon\kappa_x + \sigma_x\Delta t)\epsilon}\right] D_y\big|_{i,j+1/2,k+1/2}^{n+1/2} - \left[\frac{2\epsilon\kappa_y - \sigma_y\Delta t}{(2\epsilon\kappa_x + \sigma_x\Delta t)\epsilon}\right] D_y\big|_{i,j+1/2,k+1/2}^{n-1/2}$$

(A.5)

$$E_z\big|_{i,j+1/2,k+1/2}^{n+1/2} = \left(\frac{2\epsilon\kappa_y - \sigma_y\Delta t}{2\epsilon\kappa_y + \sigma_y\Delta t}\right) E_z\big|_{i,j+1/2,k+1/2}^{n-1/2} +$$

$$\left[\frac{2\epsilon\kappa_z + \sigma_z\Delta t}{(2\epsilon\kappa_y + \sigma_y\Delta t)\epsilon}\right] D_z\big|_{i,j+1/2,k+1/2}^{n+1/2} - \left[\frac{2\epsilon\kappa_z - \sigma_z\Delta t}{(2\epsilon\kappa_y + \sigma_y\Delta t)\epsilon}\right] D_z\big|_{i,j+1/2,k+1/2}^{n-1/2}$$

(A.6)

A.3 B-field Update

$$B_x\big|_{i,j+1,k+1/2}^{n+1} = \left(\frac{2\epsilon\kappa_y - \sigma_y\Delta t}{2\epsilon\kappa_y + \sigma_y\Delta t}\right) B_x\big|_{i,j+1,k+1/2}^{n} - \left(\frac{2\epsilon\Delta t}{2\epsilon\kappa_y + \sigma_y\Delta_t}\right) \times$$

$$\left(\frac{E_z\big|_{i,j+3/2,k+1/2}^{n} - E_z\big|_{i,j+1/2,k+1/2}^{n}}{\Delta y} - \frac{E_y\big|_{i,j+1,k+1}^{n} - E_y\big|_{i,j+1,k}^{n}}{\Delta z}\right) \tag{A.7}$$

$$B_y\big|_{i,j+1,k+1/2}^{n+1} = \left(\frac{2\epsilon\kappa_z - \sigma_z\Delta t}{2\epsilon\kappa_z + \sigma_z\Delta t}\right) B_y\big|_{i,j+1,k+1/2}^{n} - \left(\frac{2\epsilon\Delta t}{2\epsilon\kappa_z + \sigma_z\Delta_t}\right) \times$$

$$\left(\frac{E_x\big|_{i,j+1,k+1}^{n} - E_x\big|_{i,j+1,k}^{n}}{\Delta z} - \frac{E_z\big|_{i+1/2,j+1,k+1/2}^{n} - E_z\big|_{i-1/2,j+1,k+1/2}^{n}}{\Delta x}\right) \tag{A.8}$$

$$B_z\big|_{i,j+1,k+1/2}^{n+1} = \left(\frac{2\epsilon\kappa_x - \sigma_x\Delta t}{2\epsilon\kappa_x + \sigma_x\Delta t}\right) B_z\big|_{i,j+1,k+1/2}^{n} - \left(\frac{2\epsilon\Delta t}{2\epsilon\kappa_x + \sigma_x\Delta_t}\right) \times$$

$$\left(\frac{E_y\big|_{i+1/2,j+1,k+1/2}^{n} - E_y\big|_{i-1/2,j+1,k+1/2}^{n}}{\Delta x} - \frac{E_x\big|_{i,j+1/2,k+1/2}^{n} - E_x\big|_{i,j-1/2,k+1/2}^{n}}{\Delta y}\right) \tag{A.9}$$

A.4 H-field Update

$$H_x\big|^{n+1}_{i,j+1,k+1/2} = \left(\frac{2\epsilon\kappa_z - \sigma_z\Delta t}{2\epsilon\kappa_z + \sigma_z\Delta t}\right) H_x\big|^{n}_{i,j+1/2,k+1/2} +$$

$$\left[\frac{2\epsilon\kappa_x + \sigma_x\Delta t}{(2\epsilon\kappa_z + \sigma_z\Delta t)\mu}\right] B_x\big|^{n+1}_{i,j+1/2,k+1/2} - \left[\frac{2\epsilon\kappa_x - \sigma_x\Delta t}{(2\epsilon\kappa_z + \sigma_z\Delta t)\mu}\right] B_x\big|^{n}_{i,j+1/2,k+1/2} \tag{A.10}$$

$$H_y\big|^{n+1}_{i,j+1,k+1/2} = \left(\frac{2\epsilon\kappa_x - \sigma_x\Delta t}{2\epsilon\kappa_x + \sigma_x\Delta t}\right) H_y\big|^{n}_{i,j+1/2,k+1/2} +$$

$$\left[\frac{2\epsilon\kappa_y + \sigma_y\Delta t}{(2\epsilon\kappa_x + \sigma_x\Delta t)\mu}\right] B_y\big|^{n+1}_{i,j+1/2,k+1/2} - \left[\frac{2\epsilon\kappa_y - \sigma_y\Delta t}{(2\epsilon\kappa_x + \sigma_x\Delta t)\mu}\right] B_y\big|^{n}_{i,j+1/2,k+1/2} \tag{A.11}$$

$$H_z\big|^{n+1}_{i,j+1,k+1/2} = \left(\frac{2\epsilon\kappa_y - \sigma_y\Delta t}{2\epsilon\kappa_y + \sigma_y\Delta t}\right) H_z\big|^{n}_{i,j+1/2,k+1/2} +$$

$$\left[\frac{2\epsilon\kappa_z + \sigma_z\Delta t}{(2\epsilon\kappa_y + \sigma_y\Delta t)\mu}\right] B_z\big|^{n+1}_{i,j+1/2,k+1/2} - \left[\frac{2\epsilon\kappa_z - \sigma_z\Delta t}{(2\epsilon\kappa_y + \sigma_y\Delta t)\mu}\right] B_z\big|^{n}_{i,j+1/2,k+1/2} \tag{A.12}$$