Finite-difference time-domain simulation of nanostructured metal films using parallel computers

by

Matthew Charles Hughes
B.Sc., University of Calgary, 2003

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Supervisor: Dr. Maria Stuchly

Abstract

A hybrid parallel finite-difference time-domain code utilizing OpenMP and MPI and embedding the Python scripting language is described. The code is used to study the transmission of light through nano-structured metal films. Simulations showing the increased transmission of light through rectangular apertures of decreasing size and the effect of aperture shape on the transmission are presented. The effects of polarization angle and basis rotation on the transmission through an array of ellipses and double holes are explored. The maximum transmission through an array of ellipses occurs when the polarization coincides with the basis vector of the unit cell. In contrast, the maximum transmission through an array of double holes occurs when the polarization vector coincides with the lattice vector of the array. As an example of a large problem which exercises the parallel components of the code, the transmission of light through an aperiodic set of randomly positioned apertures was simulated. This problem is useful experimentally as a way to estimate the light transmitted through a single hole. The simulations confirmed that the code can handle large problems, and that random holes could be used to estimate the transmissive of a single aperture to within approximately 18%.

Contents

List of Tables

List of Figures

Acknowledgments

Thanks to my supervisor, Maria Stuchly, for allowing me the freedom to pursue a rather large and risky project, and for the guidance and support. Thanks to Kris Caputa, who introduced me to OpenMP and Neil Gaiman, among other things. Thanks to Donna Shannon, who not only helped with all the administrative stuff, but who also remembered life in Alberta. Thanks to Reuven Gordon, for agreeing to be co-supervisor and for having such enthusiasm and energy for the work. Luis Netter is owed a debt of gratitude for bringing coffee and conversation, and for lending me the ergonomic keyboard I am currently typing on. My wrists thank you. Thanks of course to my colleagues in the bio-electromagnetics lab, the microwave group, and the optics group. Thanks to my family for all the love and support over the years (and years) of school. Thanks to the HEPCats, the physics and astronomy Ultimate team, who kept me from getting too lethargic. And of course, thanks to my wife Tamara.

Chapter 1

Introduction

Computers have been used practically since they were invented to model physical processes. The increasing capability of computers, particularly in terms of computation speed and memory capacity, means that larger and more complicated physical systems can now be modeled, even those that were computationally intractable a year ago.

Parallel computers have previously been programmed using a wide variety of vendor specific methods which tended to be incompatible with one another. Efforts to standardize programming models for shared memory and distributed parallel computers have emerged in the last few years, leading to the promise of increased code portability between high performance platforms. Even standard PC desktops can now be targeted using the same code base that is used to target supercomputers.

A variety of computational methods are now used to solve electromagnetics problems. While approximations and simplifications can be made to reduce the computational burden, increasing simulation accuracy and the exploration of new problems continue to drive a need for increasing amounts of computer power.

One area of interest, which requires tremendous amounts of computer time to simulate, is the interaction of light with nanostructured metallic structures. The form of the fields near the metal surface and how the fields behave in and near the metal is of particular interest.

1.1 Background

The research undertaken here is motivated by an interest in parallel computation and software engineering principles and practices, and the need to visualize and understand the processes involved in enhanced optical transmission through thin metallic films. This interesting effect was first identified in 1998, but the exact mechanism responsible for increased light transmission through the film is not fully understood [7].

The finite different time domain (FDTD) method introduced by Yee [8] solves Maxwell's equations using a second-order discretization in time and space. Since the FDTD method simulates the electromagnetic field in a volume of space, it can simulate an arbitrarily complex structure and reveal details about the fields that would be impossible to measure experimentally. The time domain nature of FDTD means that it is possible to obtain a wideband result through the use of the Fourier transform.

1.2 Research Objectives

The main objectives of the research presented in this thesis are two-fold:

- Development of a portable high speed parallel implementation of the FDTD

method.

  - Increase the flexibility and extensibility of the FDTD implementation by embedding a general purpose scripting language while minimizing the performance impact of such a modification.
  - Quantify the performance of the code on various architectures.
  - Develop problem set up guidelines for achieving good performance.
  - Identify possible problem areas and suggest future improvements.

- Modeling of the interaction of electromagnetic waves with thin metallic films.
  - Account for the existence of electron plasma within metallic films and model the effects this plasma has on the fields in the vicinity of the film.
  - Compare simulations to experimental data for the transmission of light through a single isolated sub-wavelength aperture.
  - Compare simulations to experimental data for the transmission of light through an array of sub-wavelength apertures.
  - Examine the effects of incident wave polarization on the transmission of light through an array of double holes and through an array of elliptical holes.
  - Evaluate the transmission of light through a set of randomly place apertures of equal size.

1.3 Thesis Contributions

This thesis describes the implementation of Phred, a portable parallel hybrid finite-difference time-domain code. The novel features of the code are:

- Message Passing Interface (MPI) distributed computing.
- OpenMP symmetrical multiprocessing.
- Embedding the Python scripting language for scripting and to increase the flexibility, readability, and re-usability of FDTD problem definitions.

The application of Phred to several nano-scale optics problems is also described in this thesis. Two small problems are evaluated and compared to experimental data to show that the code correctly simulates the enhanced transmission phenomenon. The effects of polarization on the transmission of light through an array of apertures of various shapes are confirmed through simulation. A problem which is difficult to simulate numerically with other methods, the transmission of light through a set of randomly placed apertures, is simulated. The power of the parallel FDTD method is demonstrated by the sheer size of the problem, which would be intractable using a serial method.

1.4 Thesis Outline

Chapter 2 consists of a review of nano-scale optical phenomenon, including an overview of the electrical properties of metals at optical frequencies.

Chapter 3 discusses high speed parallel supercomputers and the common programming models used to implement high performance software. Previous parallel implementations of the FDTD algorithm are outlined.

Chapter 4 details the data structures and algorithms that have been implemented in the code discussed in this thesis. The choice of C++ and Python as the implemen-

tation languages is discussed, particularly with respect to reuse and maintainability. The general execution path of the code is illustrated and explained.

Chapter 5 describes the performance of the code. Performance tests and optimization procedures are described. Of particular interest is the speedup obtained as a problem is executed on an increasingly large number of processors. The selection of problem and machine parameters, such as the size of the computational domain and the optimal number of processors to use, is discussed.

Chapters 6 and 7 present the results of a variety of simulations modeling the enhanced transmission phenomenon using the code described in this thesis.

Conclusions and suggestions for further work are given in chapter 8.

Chapter 2

Literature Review: Enhanced transmission

"There's plenty of room at the bottom." - Richard Feynman

Miniaturization is the driving force behind many recent technological developments. Increasingly fine lithography techniques enable smaller, more powerful integrated circuits. This has led to smaller consumer devices such as cellphones, which in turn has led to a demand for the development and refinement of display technologies. Miniaturization has also enabled research into handheld detectors and analysis equipment, which depends on the further development of fields such as nonlinear optics, surface enhanced Raman spectroscopy, nano-lithography, and system-on-chip.

In order to proceed further, it is necessary to understand how light interacts with physical structures that are smaller than the wavelength. This problem is very similar to what has already been studied by scientists and engineers interested in the microwave range of the electromagnetic spectrum for communication and radar applications. The difference lies partly in the types of structures that can be built,

and in the properties of materials at optical frequencies.

This chapter reviews the experimental results, which show the non-intuitive behavior of light interacting with sub-wavelength structures, the proposed theories to explain the observed behavior, and examines similar effects which occur in the microwave band.

2.1 Definitions

The enhanced transmission phenomenon has been researched by two largely separate groups. The effect was first observed by physicists, but was quickly adopted as field of interest by the electrical engineering community. Each group has focused on different aspects of the phenomenon. Physicists are concerned with how light interacts with the electrons within the metal, while engineers in general are content to abstract this complexity away using a complex permittivity function which varies with frequency.

The terms used by each group reflect their focus. Physicists use the term *surface plasmon polariton* to describe an electromagnetic wave interacting with electrons at the surface of a metal, placing emphasis on the particle nature of the interaction. Engineers use the term *surface wave*, focusing on the wave nature. This section defines terms that are commonly used in the discussion of enhanced transmission phenomenon. The field where the definition is normally used is specified in parenthesis.

Bloch wave The wave-function of a particle, such as an electron, in a periodic potential. It consists of a propagating wave, $e^{-i\mathbf{kr}}$, and a periodic function with the same period as the potential. Bloch waves may also be used to describe an electromagnetic wave interacting with a periodic structure. (Physics)

dynamical diffraction A process in which scattered waves exchange energy back and forth with diffracted waves. (Physics)

Fabry-Pérot resonator A cavity with parallel reflecting surfaces in which a wave propagating in a direction normal to the surfaces resonates at a frequency determined by the distance between the surfaces.

leaky wave A wave, which exponentially increases in amplitude away from the interface between two materials. Such a wave cannot exist without another type of wave present to support it. (Engineering)

localized surface plasmon Non-propagating surface plasmons in the neighborhood of a certain geometrical feature, such as a single metallic nano-particle. The electron gas with which the electromagnetic field interacts is confined to a certain volume, and the surface plasmon is therefore restricted to the surface bounding that volume. (Physics)

plasmon A quantized packet of energy carried by the motion of charged particles in a plasma. Bulk plasmons have longitudinal components only. (Physics)

plasmon A wave propagating along the interface between metal and another material where the real part of the permittivity of the metal is negative due to the existence of an electron plasma in the metal. May also refer to a wave propagating in the ionosphere where the real part of the permittivity is negative due to the motion of charged particles. (Engineering)

polariton An interaction between an electromagnetic wave and charged particles, where changes in particle position and field intensity occur on the same timescale, causing a long term entanglement that can be treated as a quantized particle [9]. (Physics)

surface plasmon A plasmon, which has both longitudinal and transverse components, due to the interaction of the charged particles with a confining surface. (Physics)

surface plasmon polariton An electromagnetic surface wave interacting with a surface plasmon. (Physics)

trapped surface wave A wave, which propagates along the interface between two materials with a phase velocity lower than the speed of light in free space. The energy of the wave is carried close to the interface. (Engineering)

Wood's anomaly A minimum in the diffraction pattern of light from a grating where the diffracted light is tangential to the plane of the grating.

The terms *plasmon*, *surface plasmon*, *localized surface plasmon*, and *surface plasmon polariton* are generally used interchangeably to refer to the propagation of light along a metallic surface, although strictly speaking, they all refer to a different phenomenon. In this thesis, the discussion will be restricted to surface and leaky waves except when discussing other papers.

2.2 Enhanced transmission

The theory of diffraction of small holes by Bethe [10] is generally accepted to describe the transmission of electromagnetic radiation though sub-wavelength holes in a thin metal film. The experimental result published Ebbesen *et. al.* in 1998 [7] was surprising, since it showed that more radiation was transmitted though an array of holes than expected from Bethe's theory. Ebbesen noted that the experimental results suggested that the observed enhanced transmission was "due to the coupling of

light with plasmons ... on the surface of the periodically patterned metal film [7]." This result was expanded upon by H. F. Ghaemi *et. al.* [11], where an analysis of the zero-order transmission spectrum of a periodic structure was described.

A one-dimensional numerical result for transmission through an array of slits was reported by U. Schroter and D. Heitmann [12]. Numerical simulation of an array of holes was done by Salomon *et. al.* [13], and qualitative agreement was obtained. Salomon *et. al.* concluded that the observed enhanced transmission phenomenon was the result of "... electromagnetic coupling between holes in an array via surface plasmon polaritons propagating on the periodically structured surface [13]."

An interpretation of enhanced transmission in terms of Bloch waves and dynamical diffraction was developed by Treacy [14]. Bloch waves describe the motion of an electron in a periodic potential well, a quantum mechanical result commonly used in the study of x-ray crystallography. Under certain circumstances, dynamical diffraction theory leads to a result where the diffracted light exchanges energy back and forth (dynamically) with the forward scattered light.

Optical resonance in a narrow slit in a thick metallic film was examined by Takakura [15]. The metal was treated as a perfect conductor, and the paper focused on the fields within a single slit. Fabry-Pérot resonances are shown to exist in a slit when the metal is thick enough, where the hole though the film acts as a Fabry-Pérot resonator for light propagating through. The wavelengths at which peak transmission occurs are shown to shift to longer wavelengths than would be expected from a simple Fabry-Pérot analysis.

An experimental test of Takakura's theory was reported by Yang and Sambles [16]. The experiment used radiation in the microwave region, and the results confirmed Takakura's prediction of Fabry-Pérot behavior. At resonance, the transmitted

radiation was found to be more than two orders of magnitude larger than the radiation which was directly incident on the slit.

An array of square apertures was studied by Martín-Moreno *et. al.* [5]. Experimental results showed two interesting effects: enhanced transmission at $\lambda \approx 800$ nm and the appearance of Wood's anomalies at ≈ 550 nm and ≈ 750 nm. Using mode matching, taking into account both evanescent and propagating modes within the hole, a theory describing the interaction of the electromagnetic fields with the array was developed. The mode matching method successfully predicted the enhanced transmission peaks at $\lambda \approx 800$ nm. A simplified model based on a single evanescent mode within the hole was developed to help explain the physical nature of the phenomenon in terms of surface plasmons. The simplified model was used to examine the detail of the enhanced transmission peaks which appear at $\lambda \approx 780$ nm and $\lambda \approx 790$ nm. It was noted that the loss in the metal affects the transmission intensity of each peak differently. In the context of this thesis, it is important to note that the accuracy of the material model in FDTD may significantly effect the ability of FDTD to accurately reproduce experimental results.

A similar analysis of an array of square apertures was performed by Enoch *et. al.* [17]. The mode primarily responsible for the transmission of light through the array was identified by first finding a set of possible modes inside the hole, and then artificially doubling the imaginary part of the propagation constant of each mode in turn. The mode with the propagation constant which is closest to the imaginary axis with the smallest real part was the only mode found to significantly contribute to transmission through the hole.

Enhanced optical transmission through a single aperture in a thin silver film was described by Degiron *et. al.* [4]. It was noted that "the thickness and the finite conductivity of the metal has significant consequences which are far from being well

understood [4]." One of the experiments described by the paper examines the transmission of light through a rectangular aperture where its dimension parallel to the incident electric field varies. As the aperture becomes smaller along this dimension, the peak intensity normalized to the hole area increases. Additionally, the wavelength at which peak transmission occurs is red-shifted as the aperture size decreases.

The reason for this redshift with declining aperture area was theorized and verified numerically by Gordon and Brolo [18]. Through the use of the effective index method, the penetration of the field into the metal and coupling between the surface plasmons propagating on the interior surfaces of the aperture were shown to cause the observed redshift.

The impact of aperture shape on the transmission through an array was investigated by Koerkamp *et. al.* [19]. The width of the rectangular apertures was found to have similar effects on the transmission intensity and redshift as those observed by Degiron *et. al.* [4]. The transmission curve was also affected by the presence of the array, which caused Wood's anomalies to appear. In addition to regular arrays, the transmission through a set of randomly positioned apertures of the same size was also tested. As the transmission through a single hole is very small, it is difficult and expensive to make an accurate experimental measurement.

The use of multiple holes makes it possible to use less accurate equipment by measuring the transmission of a large number of holes, where the transmission through a single hole can be estimated by dividing the total transmission by the number of holes. Some effects due to interactions between surface waves and diffraction should still be expected, but may be difficult to quantify experimentally. The FDTD method is ideal for simulating a large set of randomly placed apertures which would be difficult to analyze by other numerical methods.

The transmission of light through an isolated circular aperture and an array of circular apertures in gold film and a perfect electric conductor has been simulated using the FDTD method by Chang *et. al.* [2]. In the simulations, the gold film was supported on a glass substrate, but no binding material was included. A thin layer of chrome or nickel is generally required to help the gold adhere to the surface of the glass, so the results of this paper may not be representative of experimental situations. An interesting feature of some of their simulations was that they included the probe of a near-field scanning optical microscope (NSOM), which is a device that can be used to gather data experimentally. This revealed that the probe tip had very little effect on the fields near the surface and that any experimental measurements using NSOM would accurately represent the field intensity near the aperture. Surface plasmon polariton Bloch waves and Wood's anomalies were shown to be present for an array of holes.

In addition to arrays of slits and holes, more complex structures have been studied experimentally. H. J. Lezec *et. al.* examined the transmission of 400-900 nm light through a single sub-wavelength hole surrounded by concentric grooves etched into the surface of the film [20]. A similar experiment was done for 4-6 mm microwave radiation, with similar results [21]. Patterning on the illuminated side of the metal plate was found to increase the transmission through the hole, while patterning on the transmission side was found to focus the emitted radiation into a narrow beam.

2.3 Coupling on periodic surfaces

Oliner and Jackson explained the enhanced transmission and collimated beam effects seen in the transmission experiments with periodic structures in terms of leaky waves [22]. They also developed a leaky wave antenna model for such structures [23]. An

example structure was examined, where the metal was modeled as a lossless over-dense plasma with negative real permittivity. The dispersion behavior of surface plasmons was shown and plotted on a band-structure diagram. Of particular importance is the fact that as the frequency increases, the propagation constant of the wave along the surface of the interface must eventually become complex, which corresponds to a leaky wave.

It is noted that periodic structure on the transmission side of a plate can be tuned to focus radiation at the broadside, and that the same process in reverse is responsible for converting incident light into surface plasmon modes. These two papers concisely explain the observed enhanced transmission phenomenon for periodic structures through the use of well known concepts from the microwave world.

2.4 Models of dispersive materials

The interaction of electromagnetic waves with physical materials can be described on a macroscopic scale in terms of the constitutive parameters of the materials, namely complex permittivity and permeability. Many materials of interest are non-magnetic, having constant permeability equal to that of free space. The Lorentz model characterizes dielectrics in which electrons are bound to atomic nuclei. The electron's mass results in an inertia term, collisions with nearby electrons and atoms are treated as a damping term, and the electric field binding the electron to an atom acts like a spring. In the Drude model, electrons are not bound to atoms, and are treated as a free electron gas residing in the material.

Derivations of the Lorentz and Drude models are given in Ishimaru [24] and Bohren [25]. Ishimaru assumes that the electric field applied to the electrons has the

form $e^{j\omega t}$, while Bohren assumes $e^{-j\omega t}$. Care must be taken when interpreting the resulting equations, since the results are slightly different. A abbreviated derivation of the Drude model in which no assumption about the form of the forcing function is made is given in Appendix A.

The Drude model does not completely characterize the permittivity of a metal. The plasma frequency and collision frequency calculated from experimental data are only valid over a fairly narrow frequency band. The Drude model also does not account for permittivity due to other effects. Thus, it is necessary to add a constant bulk permittivity term.

A metal at optical frequencies can be modeled using a set of three parameters, $(\epsilon_\infty, \omega_p, v)$, which are the bulk permittivity of the material, the plasma frequency, and the electron collision frequency. These parameters were found by fitting Equation 2.1 to the experimental data tabulated by Johnson and Christy [1].

$$\epsilon(\omega) = \epsilon_\infty + \frac{\omega_p^2}{-\omega^2 + jv\omega} \tag{2.1}$$

The Drude model parameters for gold were determined by Chang *et. al.* [2]. For wavelengths between 500 nm and 1000 nm, $(\epsilon_\infty, \omega_p, v) = (11.4577, 9.4027 \text{ eV}, 0.08314 \text{ eV})$. For wavelengths near 532 nm, they determined the parameters to be $(\epsilon_\infty, \omega_p, v) = (12.9965, 9.8528 \text{ eV}, 0.2401 \text{ eV})$. The validity of these parameters is shown in Figure 2.1.

For silver, the Drude model parameters were calculated using data from Johnson and Christy [1]. They were determined to be $(\epsilon_\infty, \omega_p, v) = (4.15, 8.9744 \text{ eV}, 0.13408 \text{ eV})$. The permittivity of silver obtained using the Drude model is graphed with the experimental data from Johnson and Christy in Figure 2.2.

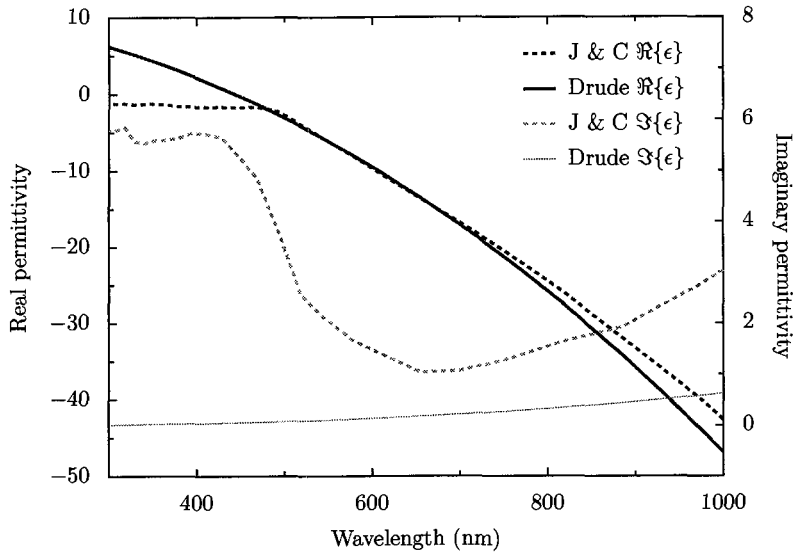


Figure 2.1: Comparison between experimental permittivity data for gold from Johnson and Christy [1] and the Drude model parameters determined by Chang *et. al.* [2] for $500 < \lambda < 1000$ nm.

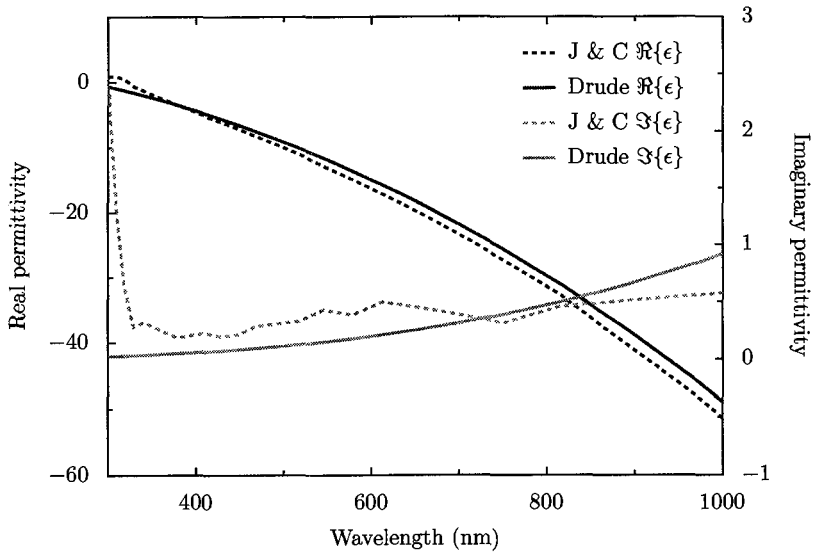


Figure 2.2: Comparison between experimental permittivity data for silver from Johnson and Christy [1] and the Drude model parameters.

2.5 Summary

Three different mechanisms related to enhanced optical transmission have been identified and studied. The interaction of propagating modes with the electromagnetic fields near the surface of a metal is one important aspect of the phenomenon. This has been described in terms of leaky waves and dynamical diffraction, and occurs when periodic structures are present, even for a perfect electric conductor. The existence of surface waves due to the negative real permittivity of some metals at optical frequencies has also been found to play a role in enhanced transmission. The final mechanism is how energy is transmitted from one side of the film to the other, and the effect of the shape of the aperture on the transmission. This effect has been explained through the use of mode matching and the effective index method.

The review of the literature, as briefly outlined, indicates that the individual mechanisms involved in the enhanced transmission phenomena are largely understood and models exist for simple geometries. Further work regarding the interactions between these mechanisms and the behavior of electromagnetic fields near nanostructures could have applications ranging from display technology to system-on-a-chip sensor packages. While some simple computational studies have been done, further development will require increasingly large simulations.

Fully three dimensional simulations of the interaction of light with large, aperiodic structures with fine geometrical features and realistic dielectric properties will required enormous amounts of memory and processor time to compute. The FDTD method's ability to handle large, arbitrarily shaped structures, and its ability to reveal information about the fields interacting with these structures makes it an ideal method for studying enhanced transmission phenomenon.

Chapter 3

Literature Review: Large-scale computation

> "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable." - Leslie Lamport

Optics problems such as those reviewed in the previous chapter can be analyzed using FDTD, but because of the size of the structures, the computational domain required can be very large. The computation of such simulations requires a great deal of memory to store the field components at each grid point in the domain, and fast processors to quickly compute field component updates.

FDTD has been implemented on many different supercomputers that are capable of satisfying these requirements to a certain degree. Supercomputer technology changes quite rapidly, so some very capable codes have rapidly become obsolete as the machines they run on are phased out. New codes must be written to target new machines as technology evolves.

This chapter presents an overview of scientific and high performance computing. Different types of supercomputers and current technologies for high performance computing are described. Recent parallel FDTD implementations are outlined.

3.1 High performance computers

The most common categorization of modern supercomputers was initially outlined by Flynn [26]. Under Flynn's taxonomy, computers are classified according to the relationship between the number of instruction streams and the number of data streams. The simplest form is single instruction, single data (SISD), where one stream of instructions operates on one stream of data. Until recently, most desktop computers with one processor fit in this category.

Single instruction multiple data (SIMD) describes machines where one instruction stream operates on multiple data streams. Usually SIMD machines have a multiple simple processors which all read from a shared memory and execute instructions in lockstep. This type of machine is not very common.

Vector processors, where a single processor is able to operate on multiple pieces of data in one cycle, can also be considered a SIMD machine. Vector processors, once strictly the domain of highly specialized supercomputers such as those built by Cray, are actually quite common today. Common x86 processors from Intel and AMD, along with the G4 and G5 processors used in Apple computers, have vector processing units built it.

Most supercomputers now fall under the multiple instruction, multiple data (MIMD) category of Flynn's taxonomy. MIMD computers consist of a set of processors, each of which executes its own stream of instructions and operates on a

separate stream of data. Such machines can be organized in several different ways.

In shared-memory MIMD machines, all processors access the same memory through a common bus. Typically, this type of computer is called a symmetrical multiprocessor (SMP), where *symmetrical* refers to the fact that all processors are considered equal and can execute any task. Asymmetrical multiprocessors are also possible. In an asymmetrical multiprocessor, processors are assigned certain tasks. For example, one processor would be dedicated to running the operating system kernel, while the other would run user tasks such as simulations. Shared-memory MIMD machines can only scale to a certain number of processors before the bus bandwidth is exhausted. Once this point is reached, adding processors only leads to processors waiting for the bus to become free so that they can communicate with the memory system.

A common type of supercomputer is the distributed-memory MIMD machine. Such computers are built by networking a number of independent processors together. Each processor can access only it's own local memory through a bus; communication between processors must usually be handled explicitly by the application programmer.

Many machines combine shared-memory and distributed-memory, as shown in Figure 3.1. A node in such a machine consists of a limited number of processors sharing memory through a bus. Multiple nodes are then networked through a high speed network.

Distributed shared-memory (DSM) machines are essentially networks of shared-memory nodes connected by specialized high speed interconnects. The memory in all of the nodes is available to all processors in the machine through the interconnect. No explicit message passing is required, which greatly simplifies things for application programmers. Each processor can access memory in another node the same way it would access memory in its own node. Since accessing memory in a different node

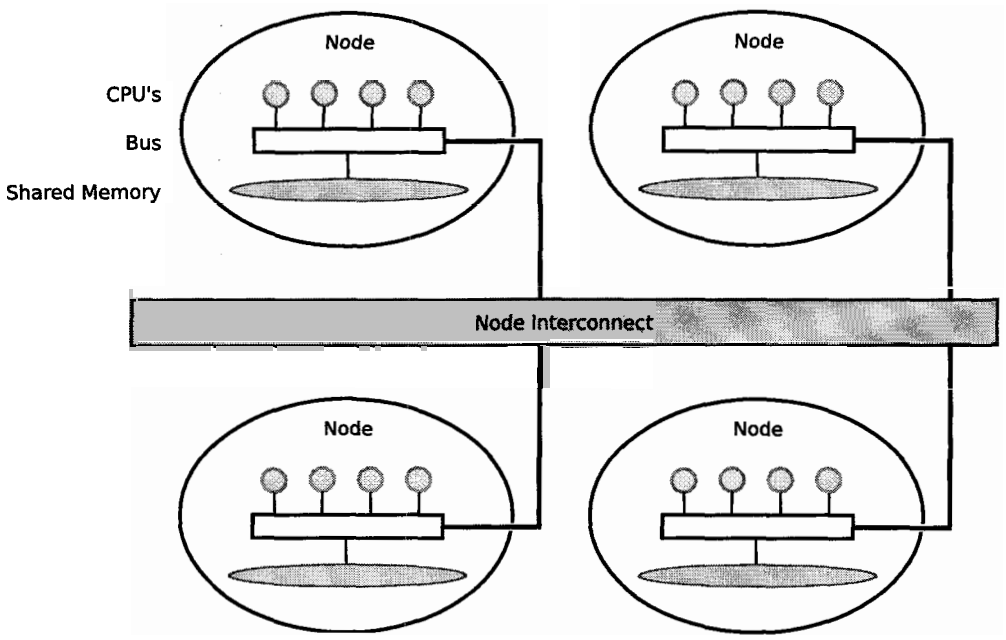


Figure 3.1: Distributed shared memory architecture

requires the use of the network, it is slower than accessing memory in the same node as the processor. Cache coherent non-uniform memory access (ccNUMA) machines are a special kind of DSM machine which ensure that the cache of each processor is synchronized with the cache of every other processor.

The TOP500 list has been tracking the fastest computers in the world since June 1993 [3]. Figure 3.2 shows that the supercomputers have been moving away from single processor, SIMD processor arrays, and SMP architectures. Clusters, constellations, and massively parallel processor (MPP) are currently the only types of supercomputer now on the list.

The difference between clusters, constellations, and MPP machines mostly relates to the speed of the interconnects between nodes, and the construction of the machine. From a programmer's perspective, they are basically all distributed-shared-memory

MIMD machines where both message passing and shared memory parallelism can be used.

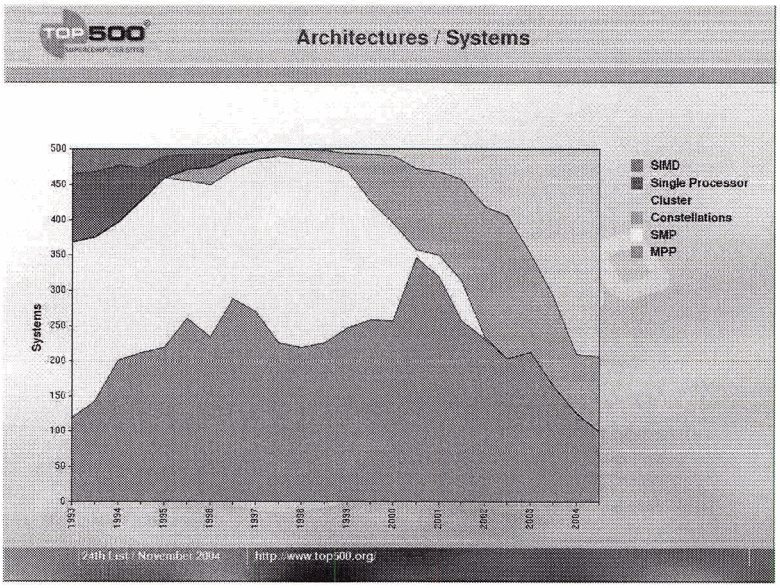


Figure 3.2: Architecture popularity on the top500 supercomputer list [3]

3.2 MPI and OpenMP

The shift away from SIMD processor arrays and vector computers to shared-memory and distributed-memory MIMD systems led to a number of different systems for shared-memory parallelism and message passing. This made it difficult to write portable software. Standardization efforts have centered around the Message Passing Interface (MPI) for message passing and around OpenMP for shared memory SMP computing. Both are now considered *de facto* standards.

Vendors such as IBM and SGI have MPI implementations available for their systems, and their compilers support OpenMP. There are a number of MPI imple-

mentations available for other environments, notably LAM-MPI [27] and MPICH [28].

Support for OpenMP is available in the Intel compiler for IA32 and IA64 platforms, making it possible to get the most out of symmetric multiprocessor machines which are becoming more common on the desktop. IBM and Absoft have a new version of the XL C/C++ compiler for Mac OS X which has a "technical preview" release of OpenMP, making it possible to utilize newer dual G5 machines using the same code as on larger machines such as IBM's RS/6000 SP.

This broad support for MPI and OpenMP means that it is straightforward to write code that scales from desktop to supercomputer, and which runs on a variety of architectures.

3.2.1 Message Passing Interface

The MPI standard is the result of the work of over 40 organizations. It began in 1993 "to discuss and define a set of library interface standards for message passing" [29]. The last release of the MPI standard, MPI-2, was release in 1997. MPI has not been adopted by a standards organization such as ANSI or ISO, but is the dominate message passing standard nonetheless.

The standard defines methods for point-to-point communication, collective communication, process groups, and topologies. Point-to-point communication is used to send data from one process to another. Both blocking and non-blocking methods are available. Blocking communication waits until the data has been transmitted before allowing the program to continue, while non-blocking communication functions return immediately. This makes it possible to overlap communication and computa-

tion, potentially hiding the latency involved in data transmission.

3.2.2 OpenMP

OpenMP is a standard for shared-memory parallelism. It consists of a set of `#pragma` directives which are implemented by conforming compilers, and a library of functions. It's primary strength is in coarse grained loop-level parallelism. In a SMP machine with two processors, a task which sums two arrays of numbers can be made to utilize both processors in a simple manner. OpenMP simply assigns half the work to one processor, and half to the other.

3.2.3 Hybrid Parallel Programming

It is possible to make use of both MPI and OpenMP in a single code. Such code is *hybrid*, able to take advantage both of symmetric multiprocessors and distributed computing.

Hybrid codes come in many forms. Figure 3.3 shows the different ways in which MPI and OpenMP can be combined in a hybrid parallel code. Tactics range from the simple master-only method, in which all MPI communication occurs on a single thread outside of parallel regions, to the very complex, where multiple threads can be computing while multiple other threads are message passing.

The performance of hybrid parallel codes on clusters of shared memory nodes has been examined by Rabenseifner [30]. The effect of inter-node bandwidth on the potential speed of the program is one of the key results. For hybrid parallel programs using the master-only model, a single processor may not be able to fully

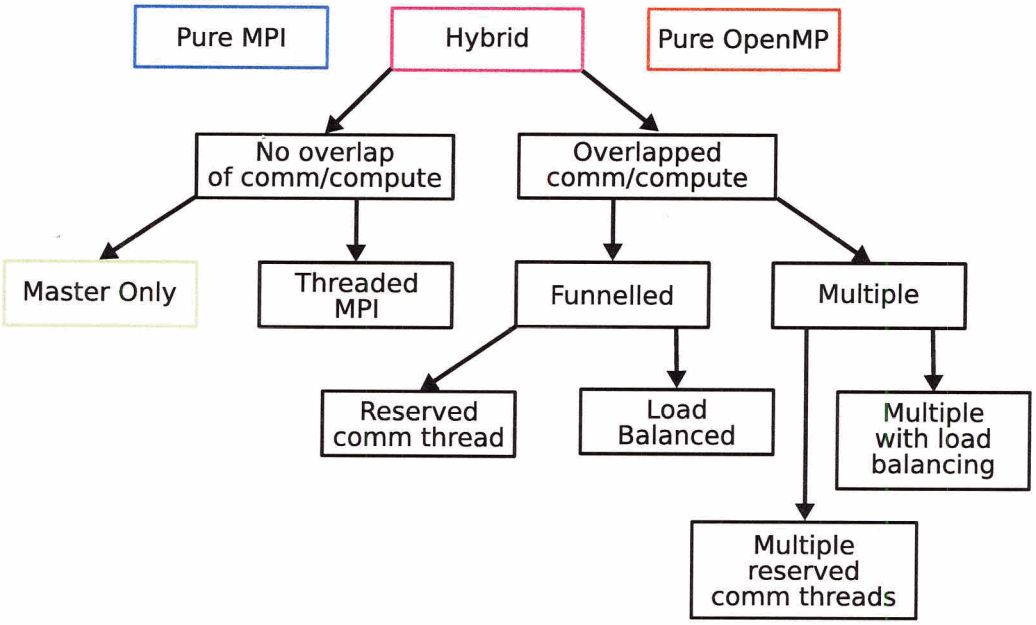


Figure 3.3: Methods of combining OpenMP and MPI in a hybrid parallel program

utilize the available inter-node bandwidth. The use of a threaded MPI implementation or overlapping communication and computation is shown to more effectively utilize the available bandwidth.

3.3 Parallel FDTD implementations

Parallel FDTD using MPI was described by Guiffaut and Mahdjoubi [31] in 2001. This paper describes domain decomposition, which involves splitting the computational domain into sub-domains, each of which is then assigned to a MPI process. The use of Cartesian topology is also elucidated. For MPI implementations that support it, specifying a topology makes it possible to map the processes to the physical processors in the machine. The use of MPI derived data types to transfer non-contiguous blocks of memory was shown to significantly increase performance compared to transferring

each contiguous part of the block individually. The paper does not cover the use of non-blocking communication, which may be useful for hiding the latency involved in message passing.

A hybrid parallel FDTD code for the simulation of photonic band-gap crystals was described by Su *et. al.* [32]. The code described by Su was targeted specifically to run on SGI Origin machines, and much of the paper dealt with SGI specific settings. The paper discussed several optimization techniques that had been used to improve the performance of the code. However, the description of the program operation suggests that message passing was used unnecessarily often. This indicates that there may be room for improvement.

There are now commercial codes capable of simulating nano-scale optical phenomenon, some using parallel computers. Licences for such codes are generally very expensive, especially for parallel processing capabilities. They also suffer from a lack of portability, since it is necessary to rely on the vendor to port the code. This can severely limit the use of such codes on large supercomputers which may be available to some groups, but not necessarily available to vendors.

The code described in this thesis is freely available, making it possible to port it to any machine that may be available. Due to its extensible design, it is straightforward to extend the code to simulate other types of electromagnetics problems. The advanced scripting language makes the automation of repetitive tasks simple and makes it possible to rapidly prototype new components. The hybrid parallel approach makes it possible to take advantage of many different machines and use whichever parallel method, MPI only, OpenMP only, or a hybrid of the two is most effective for each machine.

The trend toward distributed-shared-memory MIMD supercomputers and the

existence of MPI and OpenMP make hybrid parallel codes an attractive solution to demanding computational problems. While different machines require different tuning strategies, a base implementation using MPI and OpenMP is portable across a wide range of platforms, from clusters of networked desktop machines to the fastest super computers.

Chapter 4

Code implementation

The data structures and algorithms are the heart of any program. In an FDTD code, there are two separate but equally important sets of data structures. The first and most obvious set contain the structures that have the electric and magnetic field values, along with any auxiliary data that may be required. These data structures must be simple and fast to operate on, as they are used for the bulk of the running time of any FDTD code.

The second set of structures contains supplementary information about the problem: where objects belong within the domain, what kind of excitation to use, and the results that need to be recorded. This is the set of data structures initially populated by the user. They should be easy to set up and use, but they do not have to be particularly fast, since they are used infrequently during the execution of the code.

The code described in this thesis is designed to fulfill multiple requirements. The code must:

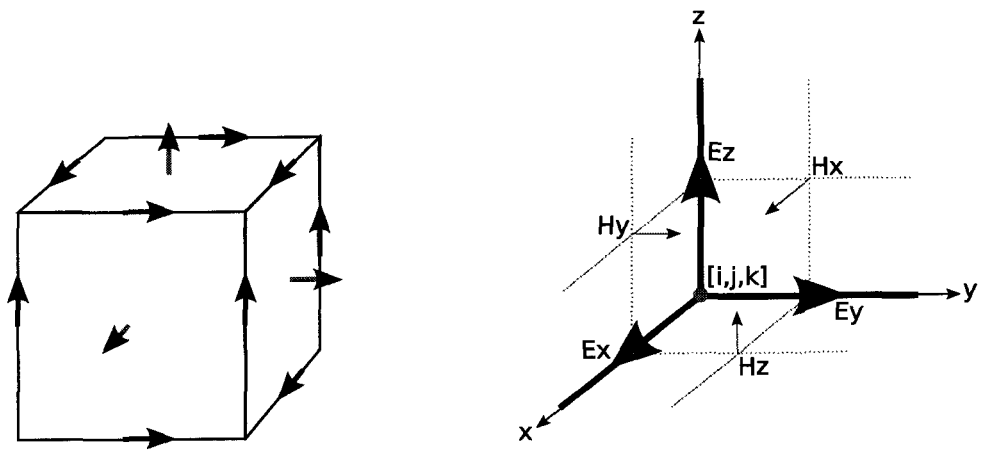
1. model frequency dispersive material,

2. execute the FDTD algorithm as quickly as possible,
3. scale well on distributed-shared-memory MIMD computers,
4. be easy to maintain and read,
5. be modular and extensible, and
6. be portable across multiple computer architectures.

To ensure that the performance goals are achievable, several key ideas are adopted for the development of the code:

- minimize message passing: Message passing is complex to program and requires the use of I/O systems, which are slow compared to the speed of the processor and main memory.
- avoid branches in inner loops: Branching instructions introduce the possibility of stalling the pipeline on modern processors and make vectorization very difficult.
- efficiency should be chosen over generality when appropriate: For example, excitations and results can only be calculated for objects which have surface normal vectors parallel to the basis vectors of the grid.

This chapter briefly describes the FDTD algorithm, and how it can be implemented on a MIMD parallel computer. The choice of implementation language is explained, the algorithms and data structures used in the implementation of the code are described, and some of the novel features of the code are discussed.



a) b)

Figure 4.1: a) The Yee cell. The electric field components lie along the edges of the cell, and the magnetic field components are normal to the faces of the cell. b) The Yee cell field components for cell $[i, j, k]$ as stored in computer memory.

4.1 The FDTD algorithm

The finite difference time domain algorithm, first proposed by Yee [8], is a simple discretization of Maxwell's equations in space and time. The Yee cell shown in Figure 4.1 a) is a unit of volume, where the electric field components are defined along the edges of the cell and the magnetic field components are normal to the faces.

The computational domain, or grid, for a FDTD simulation consists of Yee cells stacked together like bricks. Within the computational domain, any arbitrary point can be located by specifying its x, y, and z coordinates. Coordinates in real space are written as (x, y, z). Assume that a particular Yee cell within the grid can be specified using the integer values i, j, and k. Let the size of the Yee cell be Δx by Δy by Δz. As Figure 4.1 b) shows, the point in space where a Yee cell is said to start is then $x = i\Delta x$, $y = j\Delta y$, $z = k\Delta z$. The location of a cell within the grid is written as $[i, j, k]$.

The face of the Yee cell at $x = i\Delta x$ is the *back* of the cell, and the face at $x = (i+1)\Delta x$ is the *front* of the cell. The *left* face of the cell is at $y = j\Delta y$, and the right face is at $y = (j+1)\Delta y$. The *top* and *bottom* of the cell follow in the same manner. These designations for the faces of the cell are used primarily in the discussion of boundary conditions.

Time must be discretized in the same manner. For the FDTD simulation to be stable, the time step size Δt is calculated from the Yee cell size using the Courant stability criterion [33]. The electric field is calculated using a derivative of the magnetic field with respect to time and vice versa. For this reason, the electric and magnetic fields are considered to be offset from each other in time by $\frac{1}{2}\Delta t$. The magnetic field is assumed to be at $t = n\Delta t$, and the electric field is assumed to be at $t = (n+\frac{1}{2})\Delta t$.

In Figure 4.1, the curl equations relating the electric and magnetic fields are apparent. For example, the H_x component is surrounded by E_y and E_z fields, which is a graphical representation of Equation 4.1.

$$\left(\frac{E_z|^{n+\frac{1}{2}}_{i+1,j+1,k+\frac{1}{2}} - E_z|^{n+\frac{1}{2}}_{i+1,j,k+\frac{1}{2}}}{\Delta y} - \frac{E_y|^{n+\frac{1}{2}}_{i+1,j+\frac{1}{2},k+1} - E_y|^{n+\frac{1}{2}}_{i+1,j+\frac{1}{2},k}}{\Delta z} \right) = -\mu \frac{H_x|^{n+1}_{i+1,j+\frac{1}{2},k+\frac{1}{2}} - H_x|^{n}_{i+1,j+\frac{1}{2},k+\frac{1}{2}}}{\Delta t} \tag{4.1}$$

This equation can be rearranged to calculate the next value of H_x:

$$H_x|^{n+1}_{i+1,j+\frac{1}{2},k+\frac{1}{2}} = H_x|^{n}_{i+1,j+\frac{1}{2},k+\frac{1}{2}} - \frac{\Delta t}{\mu} \left(\frac{E_z|^{n+\frac{1}{2}}_{i+1,j+1,k+\frac{1}{2}} - E_z|^{n+\frac{1}{2}}_{i+1,j,k+\frac{1}{2}}}{\Delta y} - \frac{E_y|^{n+\frac{1}{2}}_{i+1,j+\frac{1}{2},k+1} - E_y|^{n+\frac{1}{2}}_{i+1,j+\frac{1}{2},k}}{\Delta z} \right) \tag{4.2}$$

The other electric and magnetic field components are calculated in a similar manner.

The new value of each field component can be calculated from the previous value and nearby field components. This locality means that for each half time step it is theoretically possible to update all magnetic or electric field components at the same time.

In general, a single processor can only update one field component in one cell at a time. A serial FDTD algorithm proceeds as follows:

1. Load problem definition, initialize data structures.
2. Starting at time step zero, iterate for N time steps:
   (a) Apply electric field excitations and boundary conditions.
   (b) For each cell in the domain, update the magnetic field components.
   (c) Apply magnetic field excitations and boundary conditions
   (d) For each cell in the domain, update the electric field components.
   (e) Process results.
3. Do any final result processing required.
4. Clean up data structures, exit.

4.2 A Parallel FDTD algorithm

There are two fundamental approaches to parallizing an algorithm. The first is functional decomposition, which identifies different tasks which can be executed in par-

allel. The second is domain decomposition, which divides the data to be processed into small pieces that can processed independently. For the FDTD algorithm, there is a clear order in which calculations must be executed, ruling out functional decomposition.

The idea that it is possible to update all field components at the same time for each half-time step clearly shows that domain decomposition can be used to parallelize the FDTD algorithm. In general, there will be many more cells than processors available to compute the update equations. Therefore, the computational domain must be divided into groups of Yee cells called sub-domains.

In order to update the cells on the edges of each sub-domain, it is necessary to have access to the field component values for one plane of cells in the adjacent sub-domain. The best solution to this problem is to introduce *ghost cells*. Ghost cells in a particular sub-domain are shadows of "real" cells that exist in another sub-domain. They exist only to enable update calculations for the field components they are next to. Ghost cells are never updated by the node responsible for the sub-domain in which they reside. Instead, ghost cells are updated by message passing, after the real cells they shadow have been updated.

As discussed in Chapter 3, the majority of supercomputers today are distributed shared memory machines. Such machines consist of multiple nodes connected by a high performance network. Each node consists of multiple CPU's connected on a bus to a shared memory.

Each sub-domain is assigned to a MPI process. In MPI only mode, there will be one MPI process per CPU. In hybrid mode, there will usually be one MPI process per node, and in OpenMP only mode, there will only be one MPI process. Message passing is used to update ghost cells between processes. In this way, the FDTD

algorithm can be parallized to take maximum advantage of the available processor power.

A parallelized version of the FDTD algorithm is shown in Figure 4.2. The parts of the algorithm that deal with OpenMP are shown in the red boxes, and the parts that deal with MPI are shown in blue boxes.

It is important to note that except where writing data to disk is concerned, no process is said to be "master" or "slave." Another approach is to designate one process as the "master," and the remaining processes as "slaves." The master would be responsible for loading the script defining the problem, calculating the domain decomposition, tasking slave processes, and so on. In Phred, all processes are as equal as possible. They all load the script file, they all execute the domain decomposition algorithm, and they all execute roughly the same amount of code.

The reason Phred is designed in this way is to make the code simple to debug and to minimize message passing. There are several places where the code does different things depending on what node it is running on, but this is the exception rather than the rule.

4.3 Domain decomposition

The domain decomposition algorithm implemented in Phred divides a computational domain into N parts, where N is the number of MPI processes that have been tasked to compute the problem. There are three parts to the domain decomposition algorithm:

1. The process assignment algorithm divides N processes up into n processes along

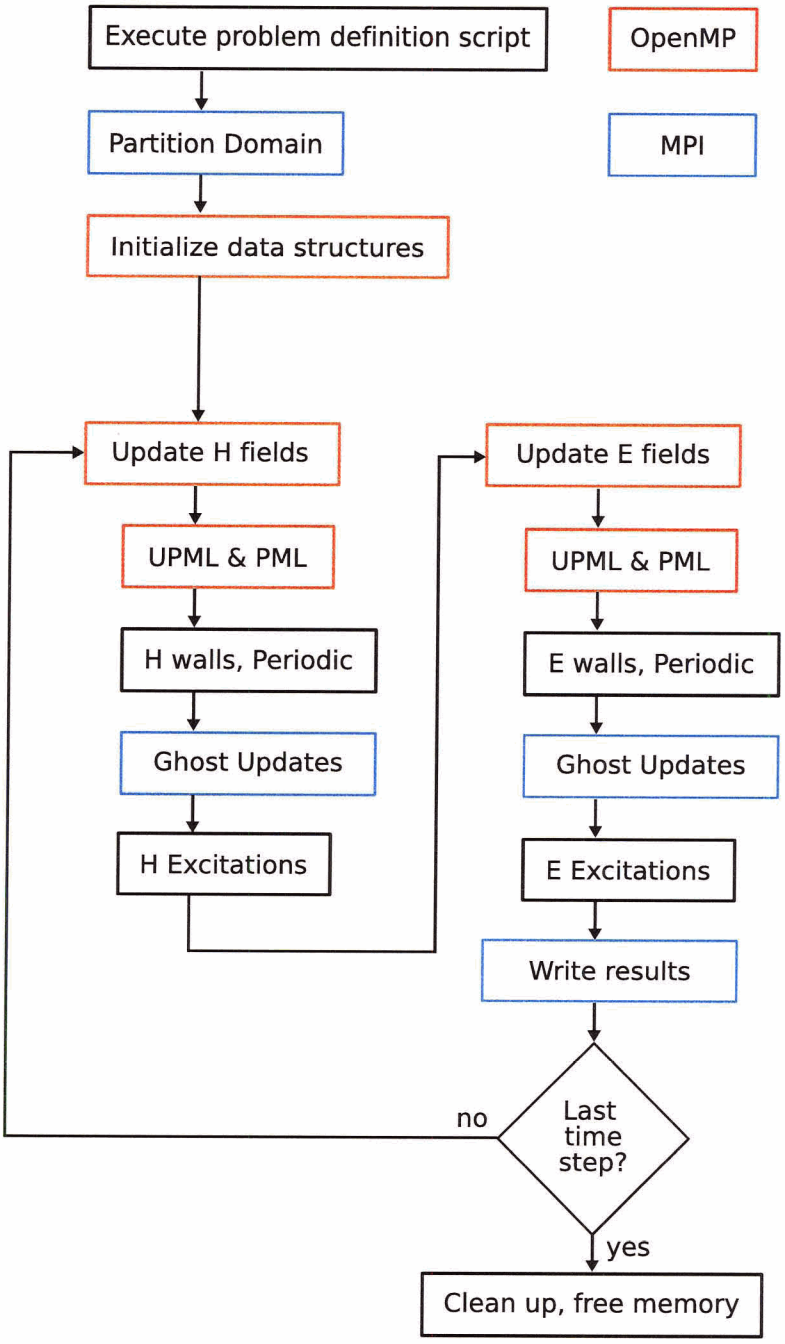


Figure 4.2: Parallel FDTD algorithm

the x axis, m processes along the y axis, and p processes along the z axis, such that $N = nmp$.

2. The N processes are mapped into a Cartesian topology which attempts to match the underlying hardware topology, such that processes which must exchange ghost cells are assigned to nodes which are physically close to each other in the machine.

3. Each process uses the topology information to calculate which sub-domain it will compute. Each sub-domain is uniquely specified by its size and starting point within the entire domain.

4.3.1 Process assignment

The simplest domain decomposition algorithm simply selects the longest axis of the domain and divides it into N pieces, where N is the number of available processors. This basic idea can be extended to three dimensions by dividing the domain along the second and third axis' as well. Let $(n, m, p) = (1, 1, 1)$ be the initial number of processors along the x, y, and z axis respectively. Let (x_g, y_g, z_g) be the number of Yee cells in the entire computational domain, and let (x_s, y_s, z_s) be the number of cells in each sub-domain.

Repeat N times:

1. If $x_s \geq y_s$ and $x_s \geq z_s$ and $(n+1)mp \geq N$, then let $n = n+1$, $x_s = x_g/n$.

2. Else if $y_s \geq x_s$ and $y_s \geq z_s$ and $n(m+1)p \geq N$, then let $m = m+1$, $y_s = y_g/m$.

3. Else if $z_s \geq x_s$ and $z_s \geq y_s$ and $nm(p+1) \geq N$, then let $p = p+1$, $z_s = z_g/p$.

While very simple, this algorithm can generate decompositions which do not work well in practice. In addition, it may not be able to use all available processors, especially when N is odd and greater than three.

Simple process assignment can also be done using the MPI function `MPI_Dims_create`. For a three dimensional grid, `MPI_Dims_create` returns the number of processors along each axis, (n, m, p), such that n, m, and p are as close to equal as possible. This method is used when the previous algorithm fails to utilize all available processors, but it cannot take the size of the computational domain into account and may produce less than optimal domain decompositions.

4.3.2 Mapping sub-domains to processes

Once the computational domain has been divided into a set of sub-domains, each sub-domain must be assigned to an MPI process for computation. This task is delegated to MPI through *process topologies*, which was first discussed in the context of FDTD by Guiffaut and Mahdjoubi [31].

After the domain decomposition algorithm has determined how many processes extend along each axis, this information is used to construct a three dimensional Cartesian topology. The MPI implementation is expected to have information about how the processors within the machine are connected to one another. A good MPI implementation should use this information to arrange processes which are close together on the physical machine such that they are close together in the Cartesian topology.

The function `MPI_Dims_create` takes the output of the process assignment algorithm, number of processes in each dimension, and creates an data structure which

represents how the processes are assigned to processors in the machine. Each process calls the `MPI_Cart_coords` function to find where in the Cartesian topology it is. `MPI_Cart_coords` returns the coordinates of the process, (i_r, j_r, k_r), where $0 \leq i_r \leq n$, $0 \leq j_r \leq m$, and $0 \leq k_r \leq p$. The r subscript is the *rank* of the process, a unique integer identifier greater than or equal to zero.

4.3.3 Calculating sub-domain size and position

Each process, knowing its coordinates within the Cartesian topology, can now calculate the size and position of the sub-domain it will be applying the FDTD algorithm to. A very simple method is used to calculate the size and starting point of each sub-domain.

The size of the sub-domain along each axis is the calculated by dividing the number of cells along each axis by the number of processes along that axis.

$$\begin{aligned} x_s^r &= \frac{x_g}{n} \\ y_s^r &= \frac{y_g}{m} \\ z_s^r &= \frac{z_g}{p} \end{aligned}$$

In general, x_g, y_g, and z_g may be odd numbers, so it is necessary to account for the remainder of the division operation.

$$\begin{aligned} rem_x^r &= x_g - nx_s^r \\ rem_y^r &= y_g - my_s^r \end{aligned}$$

$$rem_z^r = z_g - pz_s^r$$

For each sub-domain where $i_p \leq rem_x^r$, $x_s^r = x_s^r + 1$, and likewise for y_s^r and z_s^r. The starting point of each sub-domain is then calculated.

$$start_x^r = \begin{cases} i_p x_s^r + i_p & : \quad i_p \leq rem_x^r \\ i_p x_s^r & : \quad i_p > rem_x^r \end{cases}$$

$$start_y^r = \begin{cases} j_p y_s^r + j_p & : \quad j_p \leq rem_y^r \\ j_p y_s^r & : \quad j_p > rem_y^r \end{cases}$$

$$start_z^r = \begin{cases} k_p z_s^r + k_p & : \quad k_p \leq rem_z^r \\ k_p z_s^r & : \quad k_p > rem_z^r \end{cases}$$

The size and position of the sub-domain *including* any ghost cells that may be required must now be calculated. For each face of the sub-domain which touches the face of another sub-domain, an extra cell must be added along that dimension. Define $x_{s,ghost}^r = x_s^r$ and $start_{x,ghost}^r = start_x^r$ as the length and starting point of the sub-domain including ghost cells. Similar variables are defined for the y and z axis.

If $i_p \neq 0$, the back face of the sub-domain touches another sub-domain, a plane of ghost cells is required at the back face. Therefore $x_{s,ghost}^r = x_{s,ghost}^r + 1$ and $start_{x,ghost}^r = start_{x,ghost}^r - 1$. If $i_p \neq (n-1)$, the front face touches another sub-domain. $x_{s,ghost}^r = x_s^r + 1$ and $start_{x,ghost}^r$ does not change. Similar operations are performed for the y and z axis.

The sub-domain is now completely specified. For the process with rank r, the FDTD algorithm is applied to a set of cells starting at $(start_x^r, start_y^r, start_z^r)$, with

size (x_s^r, y_s^r, z_s^r). Including any ghost cells that may be present, the sub-domain starts at $(start_{x,ghost}^r, start_{y,ghost}^r, start_{z,ghost}^r)$, with size $(x_{s,ghost}^r, y_{s,ghost}^r, z_{s,ghost}^r)$.

4.3.4 Improving domain decomposition

For FDTD, goals other than minimizing the amount of data to be transferred are usually more appropriate. It will be shown in Chapter 5 that data transfer speeds for ghost cells lying in different planes are unequal. Minimizing the amount of time required for communication is therefore a better goal than minimizing the amount of data to be transferred.

Boundary condition computations are often more computationally complex than the computations applied to the interior of the domain, especially in the case of perfectly matched layer boundary conditions. Some boundary conditions may also require significantly more memory per cell than interior regions. Equalizing the computational workload and the memory requirements between sub-domains that must compute complex boundary conditions and those that do not is also an important goal to consider.

The exceedingly simplistic processor assignment algorithm currently used is not expected to perform well for large numbers of processors. Implementing better processor assignment algorithms that take data transfer rates, computational load, and memory requirements into account have the potential to significantly increase the performance of the code when large numbers of processors are involved.

4.4 Implementation Language

Computer programming for high performance computing has traditionally been done in C/C++ or FORTRAN. Which is better is the subject of a great deal of debate [34], much of which revolves around the performance of codes written in each language. Compiler technology is such that the performance difference is usually quite small, and the small remaining difference can often be overcome by understanding some key points about how each language deals with memory. This will be covered in more detail in Chapter 5.

As part of the extensibility goal that was set for the code, a scripting language was embedded. This allows users of the code to write scripts which define FDTD simulation problems in a full fledged interpreted programming language. Since there are several C/C++ libraries available to accomplish this embedding, but none for FORTRAN, C/C++ was the more attractive option. Scripting is discussed in detail in Section 4.7.

Another reason for the selection of C/C++ over FORTRAN was the availability of an older serial FDTD code which was implemented in C. The code described in this thesis is partly derived from that code. In particular, the original update equations and the Berenger PML implementation are derived from that code. The rest was discarded due to the complexity and maintenance difficulty inherent in its supplementary data structures, and due to the slow memory scheme it used.

While there are a number of C++ libraries are available for abstracting away the details of dealing with arrays [35] [36], none were used. Most such libraries are focused on high performance linear algebra operations, and as such are not suitable for the implementation of FDTD. It is also unclear how to use hybrid parallel techniques with such libraries.

For these reasons, C/C++ was selected as the language of implementation. Object oriented programming is used extensively for the supplemental data structures, while simple C arrays of contiguous memory are used to represent the electric and magnetic fields in the computational domain. This allows for flexible, modular development, while making it possible to implement high speed hybrid parallel update equations.

4.5 FDTD implementation

The most essential part of any FDTD code is the implementation of the FDTD update equations. The first step in implementing FDTD is selecting a scheme for storing information about the electromagnetic fields.

An object-oriented way to approach this problem would be to define a class containing all the information for a given Yee cell. Such an approach is inappropriate for high performance computing because it doesn't make good use of cache.

The approach inherited from the original C code on which Phred is based used pointers to pointers so that native C array syntax could be used to access the field components. For example, the E_x field component for the Yee cell at (i, j, k) could be accessed using `ex[i][j][k]`. While this made for simple and clear update equations, it introduced large costs for allocation and deallocation, and meant that accessing a field component required de-referencing three memory addressed, two more than required.

The scheme selected for Phred was a single contiguous array of memory for each field component. Accessing each field component now requires only one memory access, and it allows the cache to take advantage of spatial locality. This means that

accessing a piece of data automatically loads nearby data into the cache as well, under the assumption that it will be used soon. This memory model was found to improve performance by approximately 33% over the previous method.

Data for the Yee cell at (i, j, k) is accessed by calculating the offset of the data in the contiguous array using Equation 4.3. It should be noted that the contiguous array varies fastest along the z axis, as a C multidimensional array would.

$$offset = k + (j + iy_{s,ghost})z_{s,ghost} \tag{4.3}$$

The field component arrays must include enough space for any ghost cells which may be present at the faces of the sub-domain.

4.5.1 MPI derived datatypes

The use of a contiguous array of memory to store each field component enabled the use of MPI derived datatypes. Derived datatypes are used to simplify and facilitate the transfer of chunks of data. The parallel FDTD algorithm that has been developed relies on the transfer of of ghost cells between processes.

The ghost cells may be present in the xy, yz, or zx planes. Since data within the arrays varies fastest along the z axis and second fastest along the y axis, data in the yz plane is contiguous and simple to transfer.

In the case of the zx plane, the data is made up of $x_{s,ghost}$ pieces of contiguous data, each of which is $z_{s,ghost}$ elements long. The stride, or the number of elements between the starting offsets of each contiguous piece of data, is $y_{s,ghost}z_{s,ghost}$ elements. One method to transfer this data would be to call MPI functions to send and receive

each contiguous piece of data individually. A simpler method is to define an MPI derived datatype to encapsulate the length and stride information about data in the zx plane, and use only one set of send and receive calls instead.

The use of MPI derived datatypes considerably simplifies many parts of the code that deal with data transfers.

4.5.2 Field update equations

The field update equations are the heart of an FDTD implementation. Phred implements several sets of field update equations. The simplest set of equations supports only perfect conductors and dielectrics. A specialized set of electric field update equations supports Drude model materials, and can also support perfect conductors and dielectrics. The perfectly matched layer boundary conditions are each implemented in their own set of electric field update equations.

Each set of update equations is separate from one another to eliminate branches inside the loops which iterate over the cells in each sub-domain. On modern processors, instructions are pipelined into the processor. When a branch is present, processors "guess" which path the branch will take, and load the instructions for that path into the processor. If the processor guessed wrong, then the instructions and data that have been loaded into the pipeline must be flushed out, and the instructions for the new path must be loaded. When this happens, the processor is "stalled," and significant delays can be incurred.

The most effective way to eliminate stalling is to not have branches in the first place. For loops which work on long arrays of contiguous data, this also allows the processor to pre-fetch data into the cache, potentially increasing speed further.

Problems which use more than one type of dispersion are probably not that common, so maintaining separate sets of update equations is not that onerous. If features such as sub-cell modeling were to be added, maintaining that code in each set of update equations would quickly become an maintenance nightmare. In that case, a better solution would be to have a set of update equations which implemented all dispersions and features, and keep the other sets of equations as simple as possible.

4.5.3 Looping

The update equation for each field component is implemented inside three nested loops which iterate over the x, y, and z axis of the domain. Each field component cannot be updated at certain faces of the domain, as doing so would require data which is outside the domain and thus does not exist. For this reason, each field component is updated separately with its own set of loops.

Preliminary testing and the paper by Su *et. al.* [32] suggest that updating all three electric or magnetic field components within a single loop performs better due to increased cache utilization. In this case the set of loops which update all three components iterate over the part of the domain where all three components can be updated. Each field component must then be updated separately on the outer faces of the domain which were missed by the main loop. This would greatly increase the amount of code that must be maintained.

4.5.4 OpenMP

It is in the implementation of the update equation loops that OpenMP is used. OpenMP directives have been placed on the outermost loop, as shown below.

```
#pragma omp parallel
  {
#pragma omp for
    for (int i = xmin; i < xmax; i++) {
      for (int j = ymin; j < ymax; j++) {
        for (int k = zmin; k < zmax; k++) {
          // Update field component
        }
      }
    }
  }
```

The `#pragma omp parallel` command tells the compiler that the following block of code is to be parallized. The `#pragma omp for` command tells the compiler that the following for loop is to be executed using N OpenMP threads. The range of data the loop traverses, from `xmin` to `xmax`, is divided into N pieces. The body of the loop, starting with `for (int i...)`, is transparently placed in a separate function by the compiler. Individual threads then call that function with different parameters, so that they each process a different part of the sub-domain.

4.6 Geometry

The representation of solids is a well studied problem in computer science. It is an important part of computer-aided design (CAD) and computer-aided manufacturing (CAM) systems.

Most geometric modeling systems are designed to support rendering a model to a display device. This is not necessary for FDTD, since the model is only used once to fill the material index array, and any visualization can be done with a third-party application. For this reason, a simple constructive solid geometry (CSG) system has been implemented which provides a great deal of geometric expressiveness, but nothing in the way of rendering capability.

Constructive solid geometry describes a geometric model in terms of *primitive* objects which are transformed and combined through the use of *operators*. The result is a tree structure that can be queried to discover information about the model.

The CSG system is based on the specification for VRML and it's successor, X3D [37]. Because it is based on these specifications, it leaves open the possibility of using X3D or VRML modeling programs to construct structures which can be used in Phred.

The primitives supported by Phred are:

- a 1m square box,
- a 1m tall cylinder with a radius of 0.5m oriented along the z axis,
- and a sphere with a radius of 0.5m.

All of these primitives are centered at $(0, 0, 0)$ by default.

Phred supports three Boolean operators, union, difference, and intersection, and a transformation operator which can scale, rotate, and translate any CSG object, either primitive or constructed.

In addition, Phred also supports an array operator, which duplicates a CSG object into an array. This operator was added to simplify the definition of arrays of holes in a metallic film.

4.6.1 The inside/outside function

Each CSG object, primitive and operators both, must implement an inside/outside function. This function returns whether a particular point is inside the object, outside it, or on its boundary. For primitive objects, the process is simple; geometric formulas are used to test if a given point is inside or not.

The transformation operator applies a transformation to the point that is the inverse of the transformation applied to the child object. It then calls the child object's inside/outside function on the transformed point are returns the result.

4.6.2 Voxelization

Voxelization is the process which converts the geometry described in the CSG tree into a material identifier that can be used to look up material information during the calculation of the FDTD update equations.

In the problem set up, the Grid object is instructed about the CSG box within which the Yee cells are contained. From this information, the Grid is able to calculate

where the centre of each Yee cell is in the same coordinate system that the geometry is defined in. The Grid then asks if each point is inside a given object, and if so, retrieves the material identifier associated with that object. Since every point within the grid is explicitly set to a certain material, there is no chance of unwanted gaps or artifacts appearing at the interface between two objects.

It may be necessary to evaluate a large number of inside/outside functions for each Yee cell in the grid, especially if the arrays are involved. This process can be somewhat slow when a large number of primitives are involved. The simplicity and effectiveness of this method trumps the lack of performance, since the voxelization process only happens once and only accounts for a fraction of a percent of the run time, even for a large simulation.

4.7 Python Scripting

The Python scripting language [38] is embedded in Phred as a more powerful alternative to a simple script loader. This gives Phred the power of the entire Python language, which has a large number of modules enabling it to be used for visualization, optimization, file I/O, and a number of other features. Scripts describing FDTD problems call methods which have human readable names. This makes it possible to make script files largely self documenting and easy to read. The ability to use looping constructs, define functions and classes, and make use of Python's built in data structures makes Phred a power tool for implementing complex FDTD simulations.

The Boost.Python [39] library was used to translate the C++ objects representing Phred's internal state into Python objects that could be accessed from a script file. Boost.Python was chosen over several alternatives due to its maturity and ease of

use.

4.7.1 Prototyping in Python

It is possible to implement source functions, excitation windows, and other features in Python. This capability can be used to prototype ideas. The Python implementation can later be re-written in C++ to increase performance.

Prototyping has proven to be a powerful and useful tool during the later stages of Phred's development. It eliminated many compilation cycles during the testing of several excitation windowing functions.

Chapter 5

Performance and Optimization

"Premature optimization is the root of all evil" - Donald Knuth

This chapter discusses the performance measurement and optimization techniques used to find and correct deficiencies in the code. The effects of problem size, MPI and OpenMP load-balancing, memory access patterns, and memory and network bandwidth are discussed. The performance of an SGI Origin machine and a IBM BladeCenter are examined, and performance measurements of the code are presented.

5.1 Performance Measurement

It is impossible to know what parts of a program need optimization, and whether the optimization has been effective, unless the performance of the program is measured. The simplest method of examining the performance of a code is to measure the amount of CPU and wall clock time it takes to run. This can be done using the `time` command. This is useful for comparing two different versions of a program and verifying that one performs better than another, but it cannot identify what parts of

the program may benefit from optimization.

On some platforms, profiling the code can be done by compiling it with the `-p` or `-pg` switch. When the program is run, it will write profiling information to disk which can then be analyzed with the using the `prof` or `gprof` program. Compiling Phred with `-pg` and executing one of the built-in test problems results in the following `gprof` output:

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative  self              self     total
 time   seconds  seconds  calls  s/call   s/call  name
 16.72  22.72    22.72     1000   0.02     0.02  Grid::update_ey()
 15.97  44.42    21.70     1000   0.02     0.02  Grid::update_ex()
 15.97  66.12    21.70     1000   0.02     0.02  Grid::update_hz()
 15.78  87.56    21.44     1000   0.02     0.02  Grid::update_ez()
 15.05 108.01    20.45     1000   0.02     0.02  Grid::update_hx()
 14.64 127.90    19.89     1000   0.02     0.02  Grid::update_hy()
  5.75 135.71     7.81    12000   0.00     0.00  Ewall::apply()
  0.06 135.79     0.08        1   0.08     0.08  Grid::alloc_grid()
  0.02 135.82     0.03        1   0.03   135.85  FDTD::run()
```

The test problem uses basic field updates with no dispersive materials. It doesn't have any excitations or results, and uses electric wall boundary conditions. As a result, it exercises only the update loops and boundary conditions. The remaining parts of the program are negligible, so it is clear that any optimization effort should be focused on the update loops and boundary conditions.

On SGI systems, more sophisticated methods for quantifying program performance are available. SpeedShop (`ssrun`) is a suite of test programs which can be used to obtain a wide variety of performance metrics. The more commonly used SpeedShop experiments measure `usertime`, which is the time spent by CPUs actually executing the program, and `totaltime`, the wall clock time spent executing the program. The output of these experiments is similar to what is obtained using `prof` or `gprof`. More interesting SpeedShop experiments measure the input/output performance of the program, MPI performance, memory usage, and performance counter data such as the number of floating point exceptions, cache misses, cache hit rate, and NUMA memory access patterns.

SpeedShop was used to measure the NUMA memory access characteristic for a large problem using different numbers of MPI processes and OpenMP threads. The experiment was performed on a machine with four processors per node, and a total of sixty-four processors were used in each case. The results, shown in Table 5.1, demonstrate that choosing the appropriate ratio of MPI processes to OpenMP threads is critical for good performance. In this case, performance is measured in millions of nodes per second (MNPS), which is defined as the number of times a 100 x 100 x 100 set of cells can be updated in one second.

Using sixteen MPI processes, all four OpenMP threads were confined to a single node, where memory is accessed through a bus. With four MPI processes, the sixteen OpenMP threads are spread across four nodes. As threads may not have been located on the same node as the memory they needed to operate on, memory latency was much higher and performance was much lower.

The `perfex` program is another tool for measuring program performance on SGI systems. The MIPS processors used by SGI in its Origin line of machines incorporate a number of event counters for keeping track of things such as the number of instructions

# MPI	# OpenMP	% Remote accesses	Avg routing distance	MNPS
16	4	0.658	0.016	141
4	16	52.414	2.152	40

Table 5.1: Performance impact of non-uniform memory access

decoded, mispredicted branches, cache misses, and TLB misses. The cache data in particular can be used to find problems such as "cache thrashing," which occurs when memory is accessed in such a way that cache misses frequently occur.

5.2 Message Passing Performance

In any message passing application, the overhead costs and transmission times for each message are very important factors. The memory access patterns used to construct messages are also a very important considerations. In order to test the message passing performance of various machines, a test program was written. The program allocated a block of contiguous memory and used MPI derived datatypes to access it in the same way Phred accesses the field component arrays to transfer ghost cells.

This program was run several times on an IBM BladeCenter HS20. The HS20 is a cluster of dual processor SMP machines, known as *blades*. Fourteen blades are arranged in a chassis, and the entire machine is made up of sixty chassis'. Within each chassis, blades are connected to one another through a 1GB full duplex switch. The switch has eighteen 1GB ports, one for each blade, and four ports for communicating with other chassis'. The data transfer rate between blades within the same chassis is therefore higher than the transfer rate between blades in different chassis. This was confirmed, as shown in Figure 5.1. It shows that it is much faster to transfer large blocks of contiguous memory, such as the yz plane, than it is to transfer many small

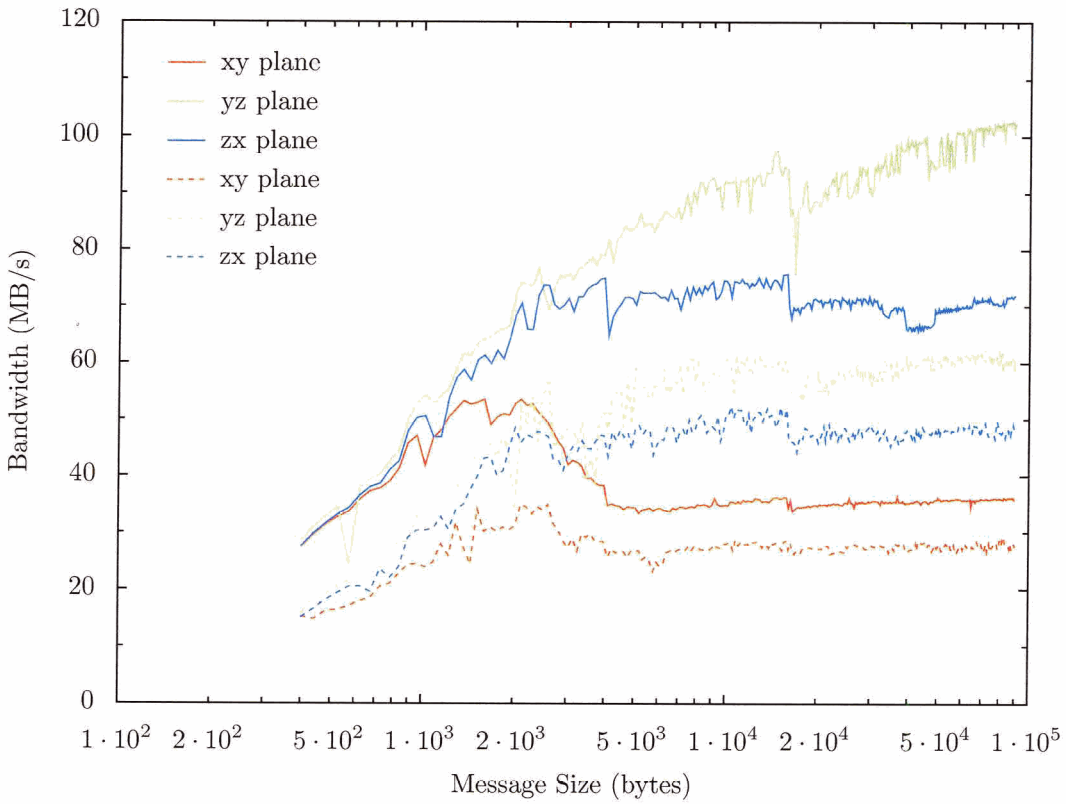


Figure 5.1: Data transfer rates as a function of message size on an IBM BladeCenter HS20 for a) blades in the same chassis (solid lines) and b) blades in different chassis' (dashed lines).

blocks, such as the xy plane.

In terms of scheduling jobs on an HS20, it is important to make sure all processes run on blades in the same chassis if possible. Since it is much faster to transfer data lying in the yz plane, the domain decomposition algorithm should favor divisions along the x axis.

5.3 Parallel Speedup

How well a program scales as the number of processors increases is an important measure of performance for a parallel code. Scalability is ultimately limited by Amdahl's law, which states that the parallel performance of a code is limited by the portions of code which must run in serial. Equation 5.1 expresses this idea mathematically [40].

$$\text{Execution time after improvement} = \left(\frac{\text{execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time unaffected}\right) \tag{5.1}$$

Increasing the number of processors has increasingly little effect when the majority of the execution time is spent on tasks that cannot be executed in parallel. For the FDTD algorithm, the majority of time is spent evaluating the update equations and boundary conditions, both of which can be parallized. For this reason, good scalability can be expected until the size of the sub-domains becomes so small that the processors can't fully utilize pipelining and cache.

For any single parallel computer, *speedup* is a useful way of determining how well a code scales as additional processors are added. Equation 5.2 expresses the speedup for N processors with respect to the amount of time the code takes to run on a single processor, T_1.

$$S_N = \frac{T_1}{T_N} \tag{5.2}$$

The scalability of the code was tested using a 128 x 128 x 128 cell computational domain. Speedup curves graph for pure MPI and pure OpenMP on an IBM RS/6000 SP are shown in Figure 5.2. The pure MPI method performed considerably better

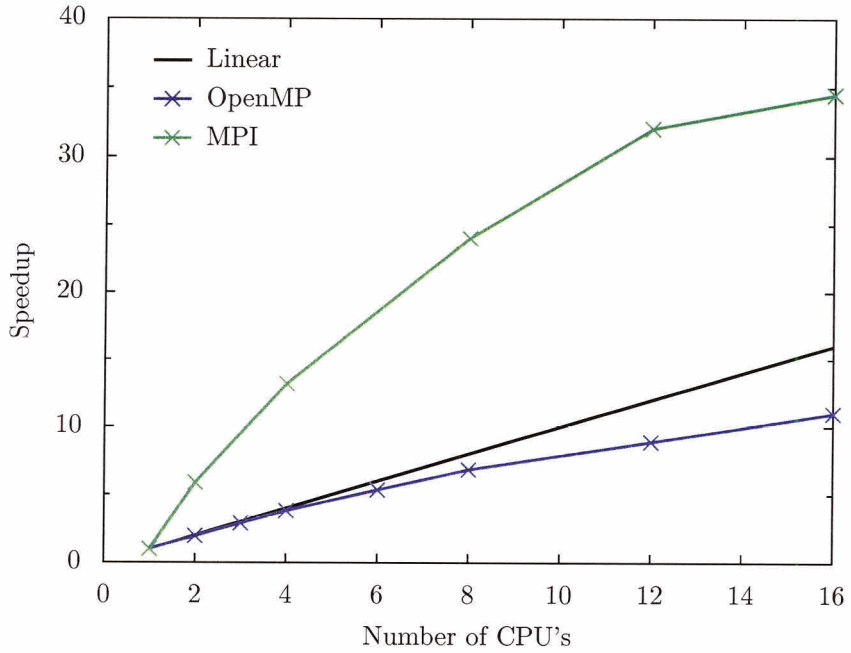


Figure 5.2: Speedup for a 128 x 128 x 128 cell computational domain on an IBM RS/6000 SP using OpenMP or MPI.

than the pure OpenMP method. The super-linear speedup increase for pure MPI is most likely due to smaller sub-domains which allow the processor to make better use of the memory hierarchy. It is also possible that significant improvements could be made to how OpenMP is utilized within the code.

5.4 Performance Enhancement

The sub-domain boundary condition for exchanging ghost cell data between processes was first implemented using blocking communication. For a problem being computed by two MPI processes, where the domain has been decomposed along the x axis, ghost cell exchanges occurred as follows:

1. Process 0, which processes the sub-domain at $(0,0,0)$, calls `MPI_Send`. The `MPI_Send` may return before a matching `MPI_Recv` call is made, but it should be assumed that it blocks.

2. Process 1, which processes the sub-domain at $(1,0,0)$, calls `MPI_Recv`. This matches the `MPI_Send` call made by process 0, and data is sent from process 0 to process 1.

3. Process 1 now calls `MPI_Send`.

4. Process 0 now calls `MPI_Recv`, and the ghost cells on process 0 are updated.

This process repeats for each field component. The communication calls are careful ordered so that no dead-locks can occur, which is important for more than two processes. Due to the fact that the `MPI_Send` and `MPI_Recv` calls block, only two processes can exchange data at a time. For a large number of processes, this blocking wastes large amounts of time.

Non-blocking calls, which are slightly more difficult to use than blocking calls, can eliminate the majority of time wasted while waiting for other processes to exchange data. A non-blocking version of the sub-domain boundary condition was implemented using `MPI_Isend` and `MPI_Irecv`. These calls both return as soon as they have queued the data transfer operation, allowing the transfer to happen in the background while the remaining transfers are set up. Before computation of the domain can continue, the `MPI_Waitall` function is called. This function waits until all of the ghost cell transfers have completed.

In order to test the performance improvement due to the use of non-blocking communication, SpeedShop was used to measure the time spent executing MPI functions for a large problem simulated using 64 processors. The results, given in Table 5.2,

	Blocking		Non-blocking	
Function	Calls	Seconds	Calls	Seconds
MPI_Send	148176	28.201		
MPI_Recv	156015	331.123		
MPI_Isend			148176	19.170
MPI_Irecv			148176	17.037
MPI_Waitall			12348	34.014

Table 5.2: Time spent executing MPI function calls for blocking and non-blocking implementations of the sub-domain boundary condition, as measured by SpeedShop.

show that the use of non-blocking communication resulted in a 69.8 % reduction in time spent executing MPI functions. The blocking version achieved 14.4 MNPS, and the non-blocking version achieved 31.1 MNPS, a 216 % improvement.

Instead of waiting until all communication had completed to resume computation, overlapping the computation of the cells, which do not make use of ghost cell data, with communication has the potential to further increase performance. Such an enhancement would be a good candidate for future version of the program.

5.5 Conclusion

While several optimizations have been examined, a great deal of work can still be done. A careful analysis of floating point requirements may reveal that some potentially unsafe compiler optimizations could be enabled. Modeling cache behavior and considering data alignment more carefully could lead to more efficient stenciling algorithms to more effectively take advantage of data locality in cache.

Measuring the performance as a function of problem size, the number of processors used, memory and network bandwidth utilization, and other factors could lead to a performance model which could be use to predict optimal configurations for a given machine. Properly predicting the number of processors that can be efficiently utilized and the domain decomposition to do so is an important aspect of responsible use of high performance computing facilities.

Chapter 6

Validation Problems

The usefulness of any simulation code is determined by its ability to accurately simulate problems. The accuracy of the code is usually determined by comparing the results of a simulation with analytical results or simulation results produced by other codes. In this case, it is also useful to phenomenologically compare simulation results with experimental results.

The success of a simulation, i.e. the production of meaningful results, depends on many factors. The cell size must be chosen so that it accurately represents both the fields and the geometrical features of the problem. The computational domain must be terminated to prevent spurious reflections from the boundaries. The time step size must be chosen to ensure the stability of the FDTD update equations and to make sure that no aliasing occurs in the Drude model. The number of time steps must be sufficient for the fields within the computational domain to dissipate. This is particularly important when simulating resonant structures. The selection of these parameters is constrained by the availability of computer resources and time.

This chapter explores the selection of simulation parameters and the ability of

the code to produce meaningful results.

6.1 Single aperture

An experimental investigation of the transmission through single isolated apertures in a metal film was reported by Degiron *et. al.* [4]. Of particular interest is their result showing increased transmission through rectangular holes of progressively smaller area. This problem is ideal for testing the code, as it exercises many important components. The problem size is relatively small, so meaningful simulations do not take a lot of time.

The set-up for this problem is shown in Figure 6.1. A 300 nm thick silver film is placed in the center of the domain in the x-y plane. The aperture in the centre of the plate is 270 nm by 105 nm. The excitation plane and the power measurement plane are located at -180 nm and 180 nm along the z axis respectively. This places them two cells away from the surface of the metal at when the cell size is 15 nm. The wavelengths of interest range from 400 nm to 900 nm, and the excitation is a Gauss modulated sine wave with a polarization vector parallel with the y axis. The edges of the excitation plane are tapered with a Gaussian window to prevent the abrupt field discontinuity that would otherwise appear.

A 2-d excitation plane is placed at $z = -180$ nm, and the power passing through a plane is measured at $z = 180$ nm. The simulation is run twice, once with the metal film, and once without. Let the transmission through the plane with no metal film present be $I(\lambda)$, and the power through the plane with the metal film in place be $T(\lambda)$. The normalized power transmitted through the plane is given by Equation 6.1. Normalizing the power in this manner removes any features that would otherwise

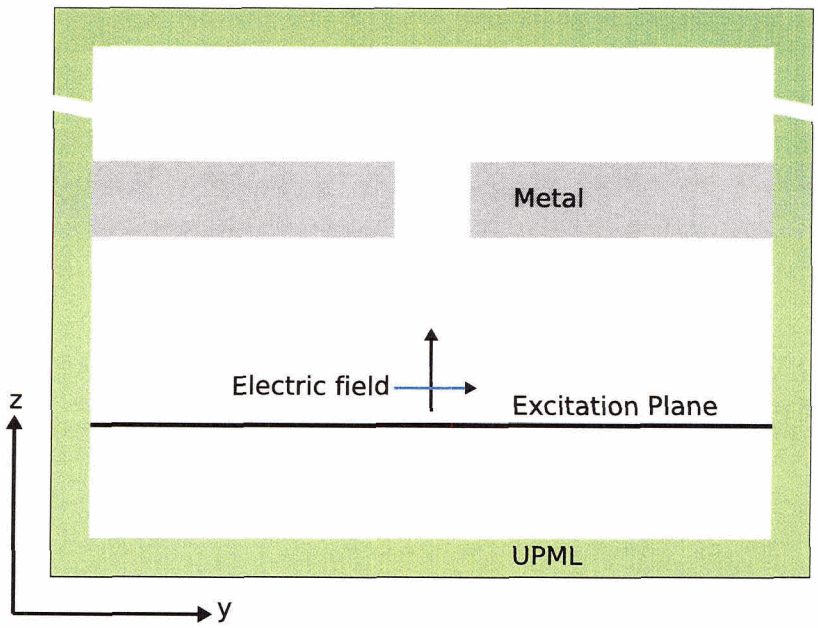


Figure 6.1: Computational domain for the simulation of light transmission through a single rectangular aperture in a silver film

appear in the output due to the fact the the power in the incident wave is not constant over the frequency range of interest.

$$T_{\mathrm{n}}(\lambda) = \frac{T(\lambda)}{I(\lambda)} \tag{6.1}$$

In some cases it is useful to normalize the transmission to the area of the aperture. In this case, Equation 6.2 is used, where A is the area of the aperture.

$$T_{\mathrm{na}}(\lambda) = \frac{T(\lambda)}{I(\lambda)A} \tag{6.2}$$

6.1.1 Discretization

In order to accurately simulate a structure, it is necessary to choose a cell size which is small enough to accurately model the electric and magnetic field components. In this case the metal has a large imaginary permittivity, so the field penetrating the metal rapidly decays. The simulation of surface waves at the metal/dielectric interface depends on the cell size being small enough to accurately model the field within the metal. Since the geometrical features of the problem are smaller than the wavelength at the frequencies of interest, it is also necessary to choose a cell size that allows for accurate modeling of the geometry. The need for accuracy must be balanced against the availability of computer resources and time.

The initial cell size was selected to be 15 nm^3, which is approximately 13 cells per wavelength at 400 nm. The simulation was run for cell sizes of 15 nm^3, 7.5 nm^3, 5 nm^3, and 3.75 nm^3. The transmission through the isolated aperture for different cell sizes are plotted in Figure 6.2. The percent change in peak normalized transmission between cell sizes of 5 nm^3 and 3.75 nm^3 was 4.3 %, while the storage requirements increased by 48 %. Based on this, a cell size of 5 nm^3 was selected for the remaining simulations.

While all cells used to test the effects of discretization are cubical, this is not a requirement. The cells can be cuboids, but the aspect ratios between the lengths of the sides should be limited. The numerical phase velocity of a wave propagating in a FDTD grid is anisotropic and it depends on the cell size. Large aspect ratios result in much faster propagation in one direction than another. This can drastically affect the results of the simulation.

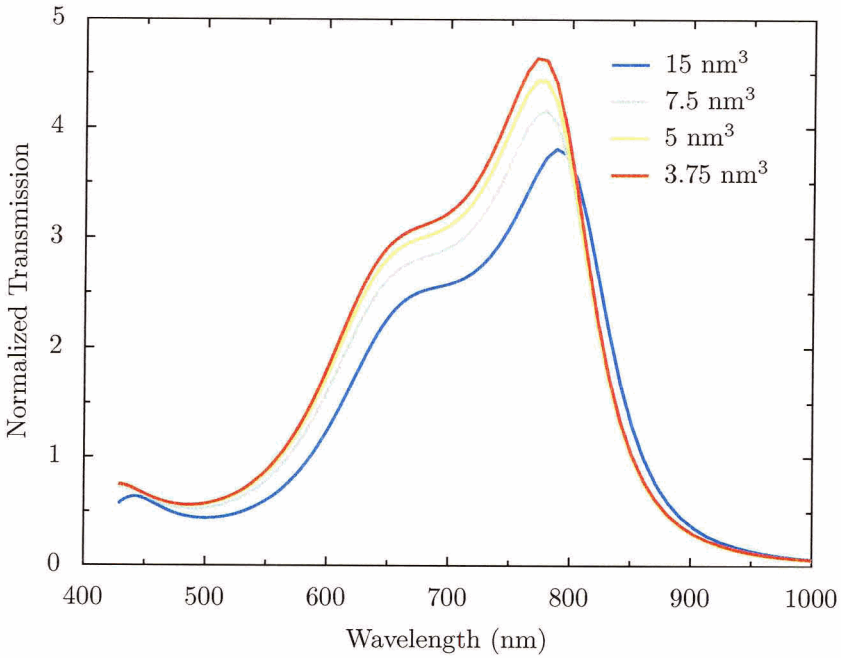
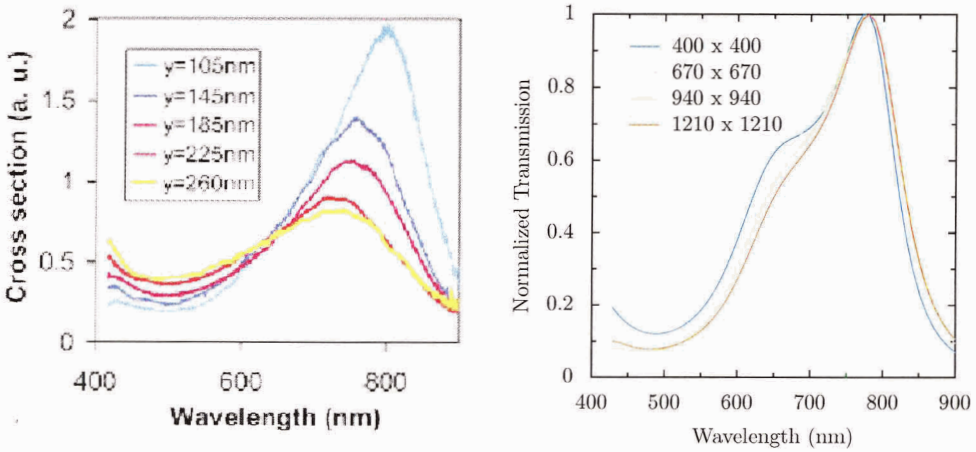


Figure 6.2: Transmission through an isolated rectangular aperture for different cell sizes



a) b)

Figure 6.3: a) Experimental results for the transmission of light through rectangular apertures ($h = 300$ nm, $x = 270$ nm) reported by Degiron *et. al.* [4] b) The normalized transmission through a 270 x 105 nm aperture in 300 nm thick Ag film for various domain sizes.

6.1.2 Domain size

The size of the computational domain is an important consideration for any FDTD simulation. A domain which is too large consumes large amounts of memory and takes a long time to simulate. A domain which is too small may not accurately simulate the fields in the region of interest due to interaction with the boundaries.

The computational domain was initially selected to be 400 nm by 400 nm by 460 nm, plus 16 cells on each face for UPML termination. The domain size was increased along each dimension by 270 nm (the length of the hole) to 670 nm, 940 nm, and 1210 nm. Figure 6.3 shows the normalized transmission curves, where the maximum of each curve has been normalized to 1.

A phenomenological approach is taken in comparing these transmission curves the

experimental curves shown by Degiron *et. al.* [4]. The 400 x 400 nm curve deviates from the experimental results in two ways. It rises as λ approaches 400 nm, and it shows a significant elbow at $\lambda \approx 680$ nm. The elbow is much less pronounced for larger domains, and the curve more closely approximates the behavior of the experimental plot near 400 nm. The source of these errors is most likely due to the imperfect plane wave excitation exciting higher order modes in the hole. Increasing the region size increased the size of the excitation, resulting in a more accurate plane wave approximation at the aperture. Given that the results can only be compared visually, the 670 x 670 nm domain appears to offer sufficient accuracy.

6.1.3 Excitation

The excitation required for this problem is a plane wave which is normally incident on the aperture. In order to excite the frequency range of interest, a sine wave enveloped by a Gaussian function given by Equation 6.3 was used. The parameters of the function were selected to be $f_0 = 500$ THz and $\Delta f = 200$ THz.

$$f(t) = e^{-((t-\frac{4}{\pi\Delta f})\pi\Delta f)^2} \sin\left(2\pi f_0(t - \frac{4}{\pi\Delta f})\right) \tag{6.3}$$

In order to approximate a plane wave, the signal given by Equation 6.3 was added to the E_y field component in the xy plane at $z = -180$ nm. It was windowed by Equation 6.4, a rectangular region where the edges are tapered with a Gaussian function. This prevents sharp discontinuities in the field at the edge of the excitation. Assuming that the window is centred at $(0, 0)$, let $(2L_x, 2L_y)$ be the size of the window, and let R be the fraction of the length along each dimension in which the Gaussian functions will be applied. Then let $r_x = RL_x$ and $r_y = RL_y$.

The Gaussian must drop to almost zero before the edge of the window. This means that at least 4σ must be under the Gaussian between $r_x < x < L_x$. Let $\sigma_x = (L_x - r_x)/\sigma_d$ and $\sigma_y = (L_y - r_y)/\sigma_d$, where $\sigma_d \geq 4$.

$$w(x,y) = \begin{cases} 1 & : \quad -r_x < x < r_x, \quad -r_y < y < r_y \\ e^{\frac{-(|x-r_x|^2)}{2\sigma_x^2}} & : \quad r_x < |x| < L_x, \quad -r_y < y < r_y \\ e^{\frac{-(|y-r_y|^2)}{2\sigma_y^2}} & : \quad -r_x < x < r_x, \quad r_y < |y| < L_y, \\ e^{\frac{-(|x-r_x|^2)}{2\sigma_x^2}} e^{\frac{-(|y-r_y|^2)}{2\sigma_y^2}} & : \quad r_x < |x| < L_x, \quad r_y < |y| < L_y \end{cases} \tag{6.4}$$

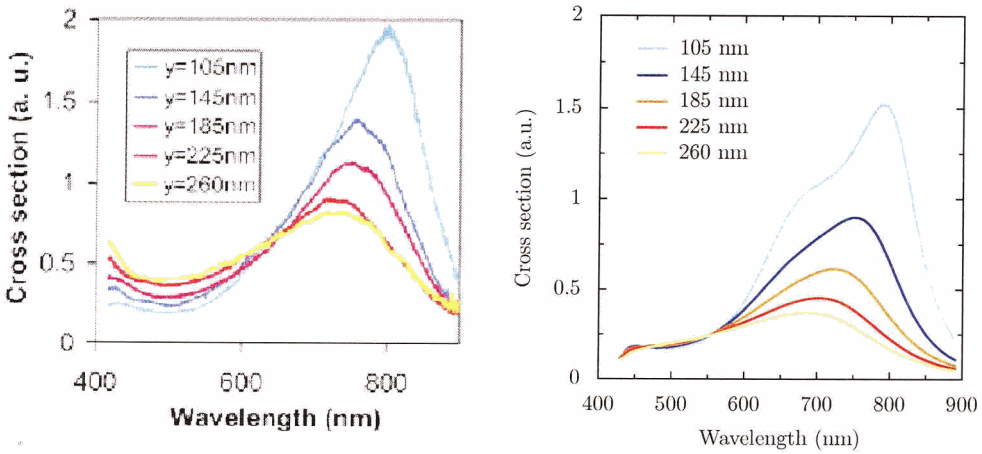
The choice of R and σ_d affect the accuracy of the plane wave approximation produced by the window. This in turn affects the accuracy of the transmission through the aperture, since an incident wave front which is not planar can excite modes in the aperture which would not be present in an experiment, skewing the results.

6.1.4 Transmission

One of the experiments performed by Degiron *et. al.* showed that as the size of a rectangular aperture became smaller along one dimension, the amount of transmitted power normalized to the aperture area increased [4]. The experiment tested holes with dimensions $x = 270$ nm by $y = 105,\ 145,\ 185,\ 225,\ 260$ nm in a 300 nm thick film.

Simulations were run for apertures of the same size. The results, shown in Figure 6.4, show that the simulation has been able to match the experimental quite well. In particular, the increasing transmission with reduced aperture area has been demonstrated.

FDTD simulations can reveal effects that cannot be directly observed in experi-



a) b)

Figure 6.4: a) Experimental results for the transmission of light through rectangular apertures ($h = 300$ nm, $x = 270$ nm) reported by Degiron *et. al.* b) Simulated results

ments. For example, the idea that the hole acts as a Fabry-Pérot resonator has been inferred from transmission measurements and theoretical calculations. Figure 6.5 shows the time domain behavior of the electric field within the hole, which clearly resonates well after the initial excitation has passed.

6.1.5 Symmetry

The single rectangular aperture problem has two obvious planes of symmetry. However, because of the nature of the fields within the hole, symmetry boundary conditions cannot be used. The polarization of the electric field is parallel with the y axis. If the electric field were normal to the $y = 0$ plane in the entire computational domain, then an electric wall could be placed at $y = 0$, halving the size of the domain. Measurements of the electric field in the centre of the hole show that a small but significant E_z component exists at $y = 0$, so it is not possible to make use of

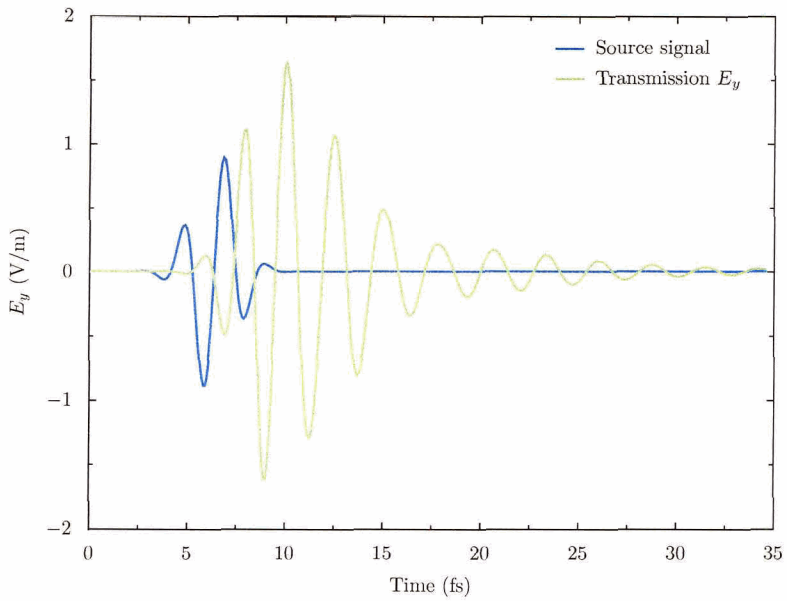


Figure 6.5: The time domain behaviour of the source excitation and the y component of the electric field in the centre of the hole.

symmetry planes.

6.2 Array of holes

A problem very similar to the one analyzed in the previous section is the one examined by Martín-Moreno [5]. A 320 nm thick silver film is perforated by an array of holes, where the hole size is 280 x 280 nm and the lattice constant is $L = 750$ nm. To simulate this problem, a cell size of 5 nm^3 is used. The computational domain is 750 x 750 x 480 nm, plus 16 cells of UPML on the top and bottom faces. The left, right, front, and back faces were set to use periodic boundary conditions. For periodic boundaries, only a plane wave excitation for which the direction of propagation is perpendicular to the normals of the faces may be used. As the excitation is constant

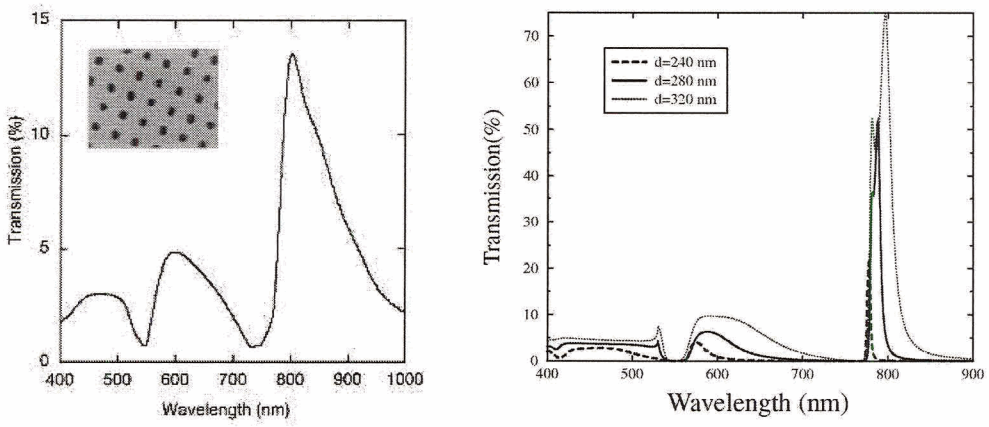
across the x-y plane, a better plane wave is possible than with the windowed excitation used in the previous section. The same time domain signal is used.

The transmission through the array calculated using a theoretical model and measured experimentally by Martín-Moreno *et. al.* is shown in Figure 6.6 [5]. The results of an FDTD simulation are shown in Figure 6.7. The shape of the FDTD curve matches very well with the theoretical model of Martín-Moreno *et. al.* . The double peaks near 800 nm are apparent in both, although the peak at the longer wavelength is greater that the one at the shorter wavelength in the theoretical model, opposite from what was found in the FDTD result. This could be due to the fact that the peak is very sharp and the DFT is not computed at enough frequency points to observe the peak.

The experimental results compare reasonably well with the FDTD simulation and theoretical results. The Wood's anomalies are present at $\approx$ 500 nm and $\approx$ 730 nm, but the enhanced transmission peak is significantly different. The experimental result shows only one peak, while the simulation and theoretical models show two. This is most likely due to the difference in the shape of the hole used for simulation and the experiment. For the simulation, as for the theoretical work done by Martín-Moreno *et. al.* , the apertures were square [5]. The experimental apertures were roughly circular.

6.3 Conclusion

The ability of the code to replicate experimental results for periodic and aperiodic structures has been demonstrated, and a set of simulation parameters which lead to reasonably meaningful and accurate results has been developed. The code can now be



a) b)

Figure 6.6: a) Experimental transmission though an array of holes and b) theoretical transmission through an array of holes from Martín-Moreno *et. al.* [5].

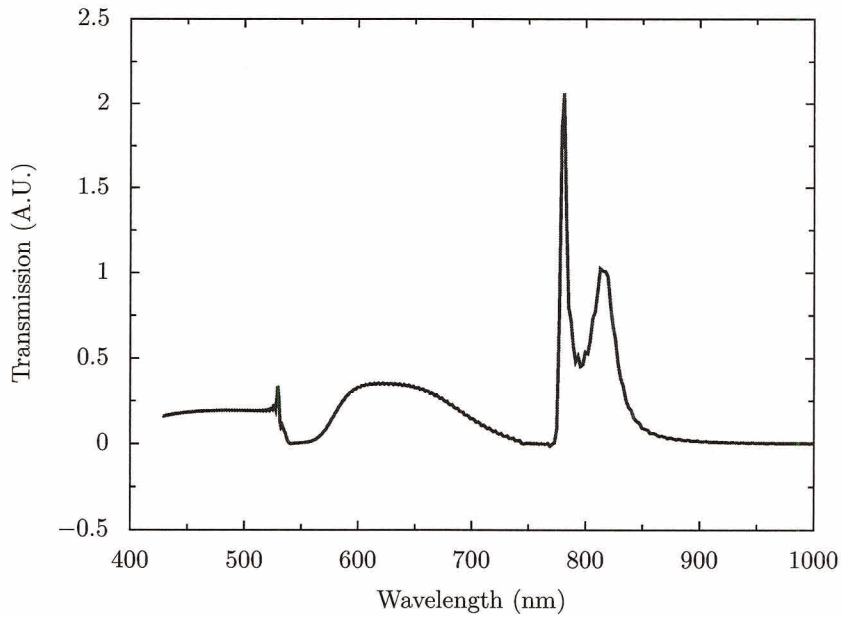


Figure 6.7: Transmission through an array of square holes in a 320 nm thick silver film simulated using FDTD. The lattice constant of the array is 750 nm.

used to predict the characteristics of potential experimental structures and confirm experimental results.

Chapter 7

Simulations

7.1 Basis and lattice polarization effects

The effect of aperture shape on the transmission of light through an array of apertures in a metal film has been well documented. Previous publications have focused on situations where the elongated basis of the aperture is co-aligned with the lattice vector of the array. Due to this alignment, it has not been possible to distinguish the effects due to the lattice from the effects due the basis of the aperture.

Gordon *et. al.* tested the effect of aperture basis and polarization rotation on transmission, with FDTD simulations performed by the code described in this thesis, thus verifying the experimental results [6]. A 100 nm thick gold film was deposited on a glass substrate, with a 5 nm layer of chromium present between the glass and the gold to aid adhesion. Arrays of ellipses and double holes were milled into the gold using a dual-beam focused ion beam. The arrays were imaged with a scanning electron microscope, as shown in Figure 7.1.

Four arrays of ellipses were fabricated, for basis angles of 0°, 15°, 30°, and 45°

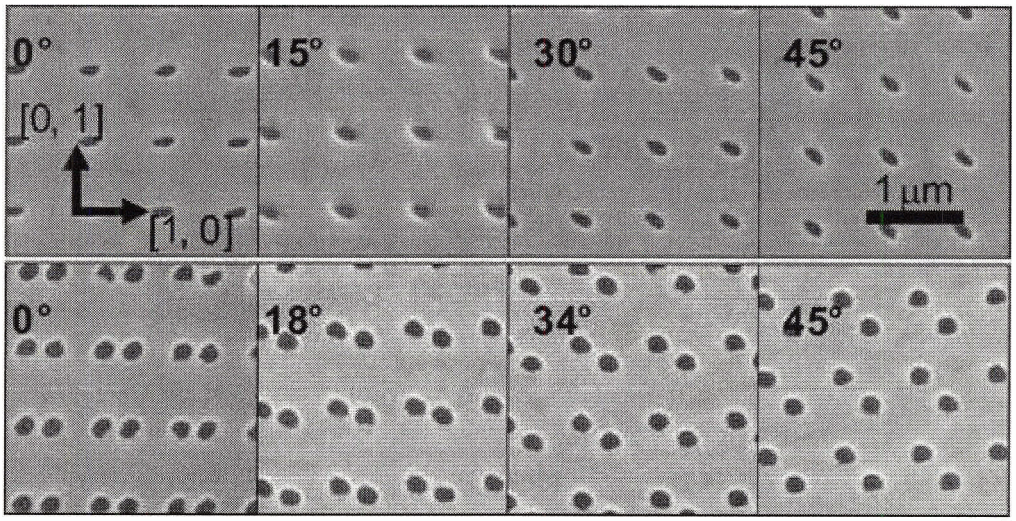


Figure 7.1: A scanning electron microscope images of the arrays of ellipses and double holes for different basis angles. All arrays have a lattice constant of 710 nm [6].

with respect to the lattice vector. The ellipses had a major axis length of 215 nm and a minor axis length of 95 nm. Each array had a lattice constant of 710 nm.

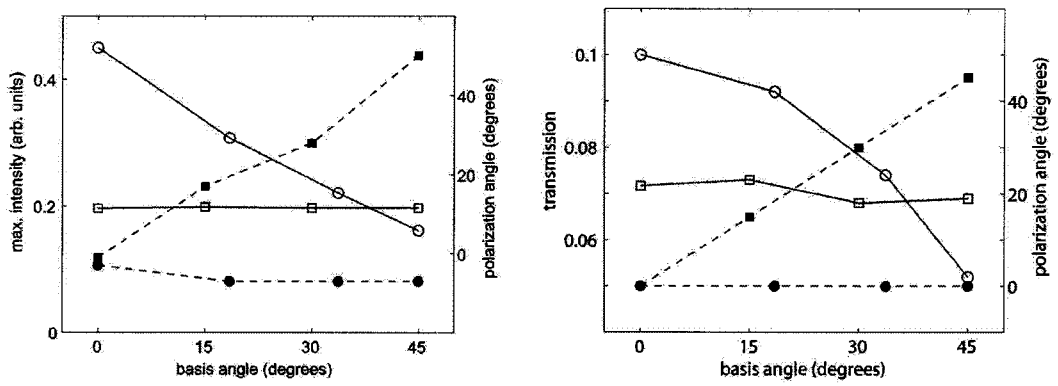
Arrays which had a unit cell consisting of a pair of circular holes were also fabricated. The diameter of each hole was 180 nm, and the centres of the holes were offset by 225 nm along the [0, 1] lattice vector. Each array had a different centre-to-centre offset along the [1, 0] direction. Steps of 0 nm, 85 nm, 170 nm, and 255 nm were used, resulting in basis vector rotations of 0°, 18.4°, 33.7°, and 45° respectively. Only the offset along the [1, 0] vector was varied to facilitate fabrication.

The films were illuminated with collimated and polarized white light from a halogen lamp. The light was focused onto the films at normal incidence through a 20x microscope objective. The light transmitted through the film was collected with a 400 μm core fiber located 1 cm away from the sample.

The FDTD simulations of the arrays were done using a 710 x 710 x 500 nm^3

computational domain with a cell size of 5 nm^3. The simulations ran for 13,308 time steps, with a time step of 8.67 attoseconds. A simulation was performed for each of the eight arrays with incident wave polarizations of 0°, 15°, 30°, and 45°.

For each array, the maximum transmission and the polarization angle at which it occurred was noted. Figure 7.2 shows the measured and simulated results. The maximum transmission through the array of ellipses occurs when the basis angle of the unit cell is the same as the polarization. The maximum transmission through the double holes occurs when the polarization is aligned with the lattice vector. This shows that maximum transmission occurs either when the incident wave polarization is aligned with the lattice or the aperture basis, depending on the shape of the aperture.



a) b)

Figure 7.2: Transmission and the polarization angle of peak transmission as a function of basis angle for a) experimental results and b) FDTD simulated results from Gordon *et. al.* [6]. The solid lines show the maximum intensity (left axis) for the ellipses (open squares) and the double holes (open circles). The dashed lines show the polarization angle of maximum transmission (right axis) for the ellipses (filled squares) and double holes (filled circles).

The results of the FDTD simulations were found to be in good agreement with the experimental results. In both cases, the maximum transmission through the ellipses occurs when the polarization angle of the incident wave is parallel with the basis angle of the unit cell, while the maximum transmission through the double holes occurs when the polarization of the incident wave is parallel to a lattice vector. The simulation successfully captures the effect present in the experiment. Additionally they provide accurate absolute values of the transmission coefficient.

The shape of the aperture and the orientation of the aperture with respect to the polarization of the incident wave clearly has a significant effect on the transmission through the aperture. The FDTD method can be applied to further explore the effect of aperture shape on transmission and field intensity within the aperture.

7.2 Randomly positioned holes

Koerkamp *et. al.* discussed the transmission through a set of randomly positioned apertures in a film as method of determining the transmission through a single aperture, since the signal for a single aperture is very weak [19]. They hypothesized that the experimentally measurable transmission from a single aperture was not significantly greater than the intrinsic transmission of the film. For this reason, a randomly positioned set of apertures is suggested as a way to obtain a transmission signal, which is significantly larger than the intrinsic transmission of the film.

Arranging the holes in a periodic fashion has a significant impact on the transmission, due to the excitation of surface waves on the film, and their interaction with leaky waves. The holes are arranged randomly in an effort to minimize the effect of periodicity on the transmission, so that the experimental result is more representative

of the transmission through an isolated aperture.

It is possible that for any given arrangement of randomly positioned apertures, significant enhanced transmission may be observed due to coincidental periodicity of the apertures. In addition, it was not apparent how close one aperture can be positioned to its neighbour without significant interactions between the fields. For this reason, it was necessary to measure the transmission through a single aperture, two apertures placed various distances apart from one another, and finally, a set of randomly positioned apertures.

7.2.1 Single aperture

In order to determine how well a set of randomly positioned apertures approximates the transmission due to a single isolated aperture, the transmission through a single aperture in a 200 nm thick gold film mounted on a glass substrate is simulated. A 5 nm layer of chromium is present between the glass and the gold, as it would be necessary experimentally to aid adhesion. The relative permittivity of the glass used in the simulation was 2.13, and the Drude model parameters used for gold are $(\epsilon_\infty, \omega_p, v) = (11.4577, 9.4027 \text{ eV}, 0.08314 \text{ eV})$. The conductivity of the chromium is 7741000 Ω m^{-1}.

Three different apertures have been simulated; a circular aperture with a diameter of 190 nm, a 150 x 225 nm^2 rectangular aperture, and a 75 x 225 nm^2 rectangular aperture. The computational domain for this problem is 600 x 600 x 400 nm^3, with a cell size of 5 nm^3. UPMLs of 16 cells were added to each face to terminate the domain.

A rectangular window with Gaussian tapered edges is used to excite the domain

30 nm below the film. The incident electric field is polarized in the x direction, parallel with the short axis of the rectangular apertures. The transmission through the film is evaluated at a plane 30 nm above the film. The time step size is 8.67 attoseconds, and the simulations with apertures in the film runs for 20,220 time steps. In order to determine the intrinsic transmission of the film, T_{film}, a simulation was run with no aperture present. The intrinsic transmission of the film was found to be negligible, and is neglected. To measure the spectral content of the incident wave, T_{inc}, a simulation was run where no structure at all was present.

The results of the simulations are shown in Figure 7.3. The normalized transmission through the aperture was calculated using Equation 7.1. The areas of the apertures have been normalized as follows: the area of the 75 x 225 nm^2 rectangular aperture is 1, the area of the 150 x 225 nm^2 aperture is 2, and the area of the circular aperture is 1.68.

$$T = \frac{T_{\text{ap}}}{T_{\text{inc}} A} \tag{7.1}$$

The transmission through the rectangular apertures is consistent with the results of Section 6.1.4, where reducing the aperture size in the direction of polarization increases transmission and produces a redshift in the transmission peak.

7.2.2 Double apertures

In order to determine the impact of multiple apertures in close proximity to one another on the transmission through the film, simulations were run for pairs of identical apertures various distances apart. Each aperture has a cross-section of 75 x 255 nm^2. This aperture size was chosen, since it transmitted the largest amount of energy

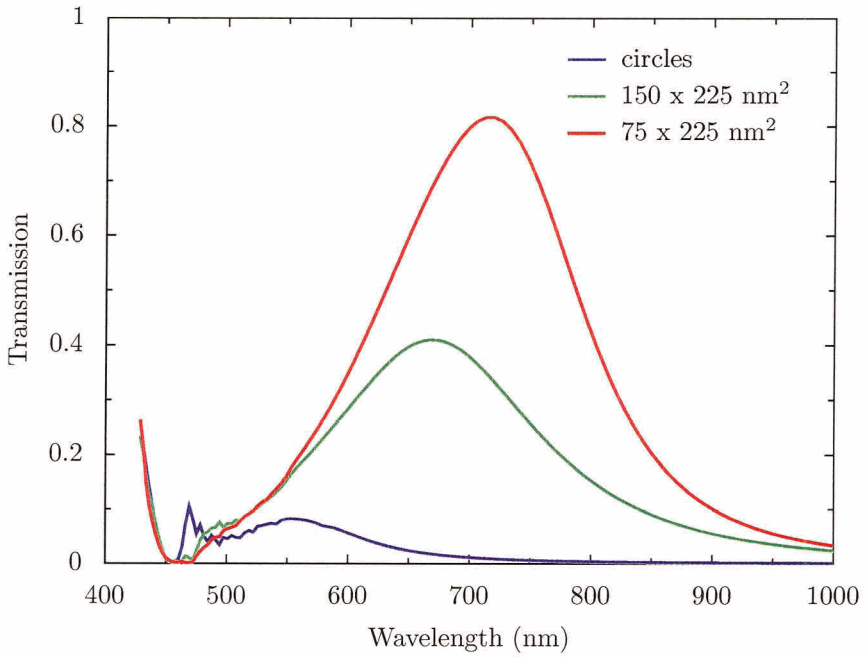


Figure 7.3: The transmission through single isolated apertures in a 200 nm thick gold film mounted on a glass substrate, where the transmission is normalized to the area of the aperture.

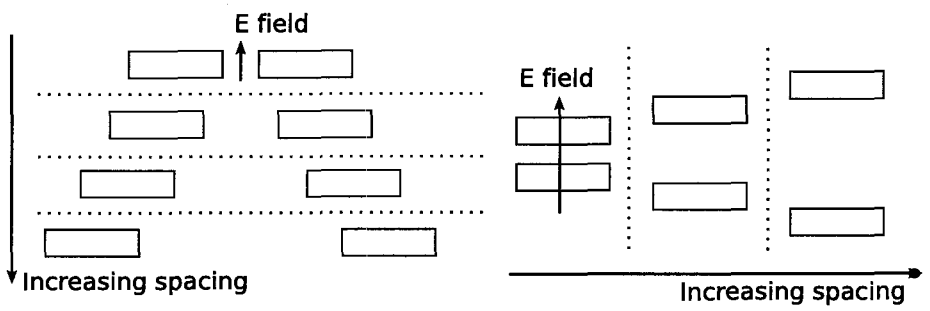


Figure 7.4: Problem set up for a pair of apertures.

through the film out of the three sizes tested experimentally [19].

For aperture pairs separated along the axis of incident wave polarization, simulations were run for centre-to-centre aperture spacings of between 150 nm and 1150 nm, in 50 nm increments. Perpendicular to the axis of polarization, the apertures had centre-to-centre spacings of 250 nm to 1150 nm in 50 nm increments.

The computational domain for these simulations, shown in Figure 7.4, started at 1000 x 1000 x 400 nm^3 for a single aperture, and increased by the same amount as the spacing between the apertures. The cell size for these simulations is 5 nm^3. The excitation is a Gauss modulated sinusoid, applied over a Gauss tapered window at $z = -160$ nm, with the electric field polarized along the x axis. The power transmitted through the film is measured as it passed through a plane positioned at $z = 160$ nm.

The normalized peak transmission through the pairs of apertures as a function of centre-to-centre aperture spacing is plotted in Figure 7.5. As the separation between the apertures increases along the axis of incident wave polarization, the transmission rises from an initial value near the transmission through a single aperture to well over twice the transmission through the single aperture. The transmission then roughly follows the pattern of a decaying sinusoid. As the spacing between the apertures increases along the direction perpendicular to the incident wave polarization, a similar,

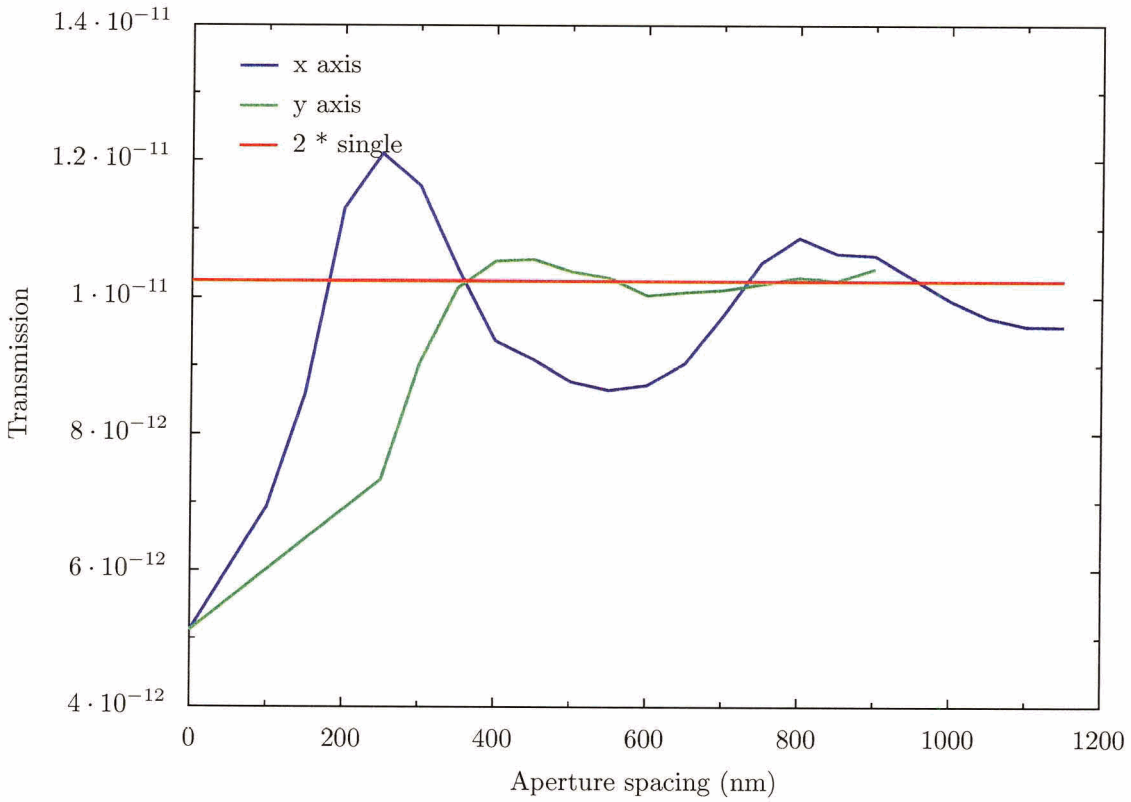


Figure 7.5: Normalized transmission through pairs of apertures as a function of centre-to-centre separation between the apertures.

but much weaker effect occurs.

Apertures in very close proximity to one another clearly do not transmit twice the power of a single aperture. More detailed analysis of the fields near the apertures is necessary to understand what is happening in this case. The oscillations observed in the peak transmission with increasing aperture spacing are most likely due to interactions between the surface waves emitted from each hole and traveling along the film between the two holes. The much smaller oscillations in the direction perpendicular to incident wave polarization are due to the fact that surface wave excitation is much weaker in this direction.

While it is not possible to draw definite conclusions about the causes of this behaviour without more in-depth analysis, some general guidelines for the design of structures involving apertures in close proximity to each other can be inferred. Along the direction of incident wave polarization, the apertures should be separated by at least 2.5 times the length of the apertures along that dimension. In the example analyzed here the period of the oscillation is approximately 575 nm, so apertures should be spaced accordingly. Perpendicular to the polarization, the apertures should be separated by at least 1.75 times the length of the aperture along that dimension. These guidelines are inferred from a very small set of data, and may not be valid for holes of different shapes or films of different thicknesses. The dielectric material bounding the film is also expected to affect the peak transmission as a function of aperture spacing.

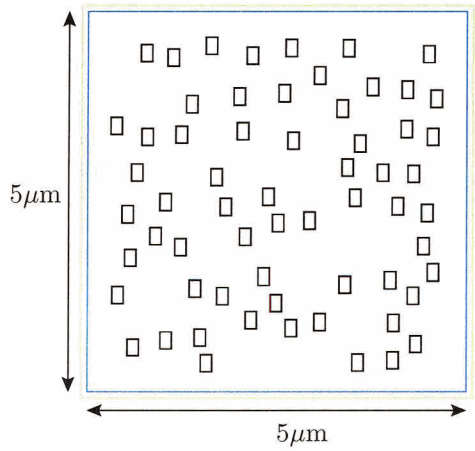


Figure 7.6: 62 randomly positioned apertures in a 200 nm thick gold film

7.2.3 Randomly positioned holes

In the experiments performed by Koerkamp *et. al.* , a 20 x 20 μm^2 film perforated by 500 randomly positioned holes was used [19]. Using a cell size of 5 nm^3, a computational domain of that size would be 4032 x 4032 x 112 cells, and require approximately 164 GB of memory. Somewhere on the order of 10,000 time steps would be required. On a SGI Origin 3900 with 256 processors, the simulation would take approximately twenty-four hours to run.

The increased transmission due to the presence of additional holes should be apparent with fewer than 500 holes. As a first test, simulations were run using a computational domain of 5 x 5 μm^2 film with 62 holes. The arrangement of holes in the film, shown in Figure 7.6, was approximately 1/16th the size of the experimental set up. The positions of the holes were randomly selected using a uniform distribution, where the centre of the hole is in the range $-2.15\ \mu m < x < 2.15\ \mu m$, $-2.15\ \mu m < x < 2.15\ \mu m$. The distance between the centres of any two holes was at least than 350 nm.

With a 5 nm^3 cell size, this problem was 1032 x 1032 x 112 cells, including UPML. Simulations were run for three different aperture shapes: circular apertures 190 nm in diameter, 150 x 225 nm^2 rectangular apertures, and 75 x 225 nm^2 rectangular apertures. The positions of the apertures were the same for all simulations. Each simulation was run for 4038 time steps, with a time step size of 8.67 attoseconds. About 11 GB of memory was required for the problem. Using 128 processors of a SGI Origin 3900, each simulation took about three hours.

The results of the simulations are shown in Figure 7.7, where the transmission through the 190 nm diameter circular apertures is shown in blue, through the 150 x 225 nm^2 rectangular apertures in green, and through the 75 x 225 nm^2 rectangular apertures in red. The transmission through the randomly placed apertures is largely undistorted, and matches fairly well with the transmission through a single hole. Due to the small number of apertures used in this case, it is likely that any latent periodicity was very weak. As the number of randomly positioned apertures increases, so to should the effects of latent periodicity.

A second set of simulations were run for a 10 x 10 μm film perforated by 125 apertures. The positions of the apertures were randomly generated as above. On an SGI Origin 3900, using 256 processors, a simulation running for 5192 time steps took approximately 3.5 hours. The performance of the code in this case was approximately 218 MNPS, up from the 50 MNPS achieved in the previous set of simulations, due to ongoing performance improvements.

The transmission through 125 apertures is compared to the transmission through a single aperture in Figure 7.8. Increasing the number of holes is seen to increase the roughness of the plots. This occurs because the transmission at a given wavelength depends on the distance between apertures. Increasing the number of apertures increases the number of aperture spacings which can either enhance or degrade

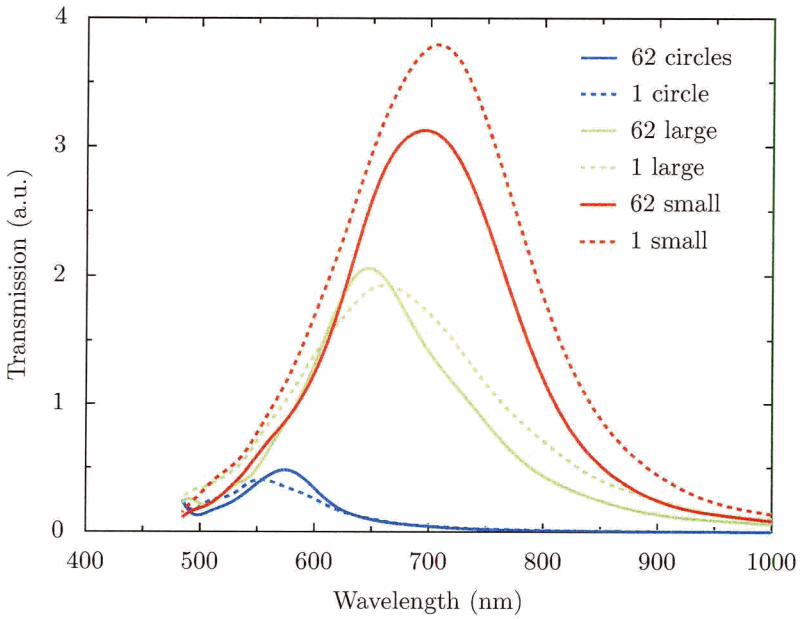


Figure 7.7: The transmission, normalized to aperture area, through a random arrangement of sixty-two apertures through a 5 x 5 μm^2 film. The transmission through a single hole multiplied by 62 is shown for comparison (dashed lines).

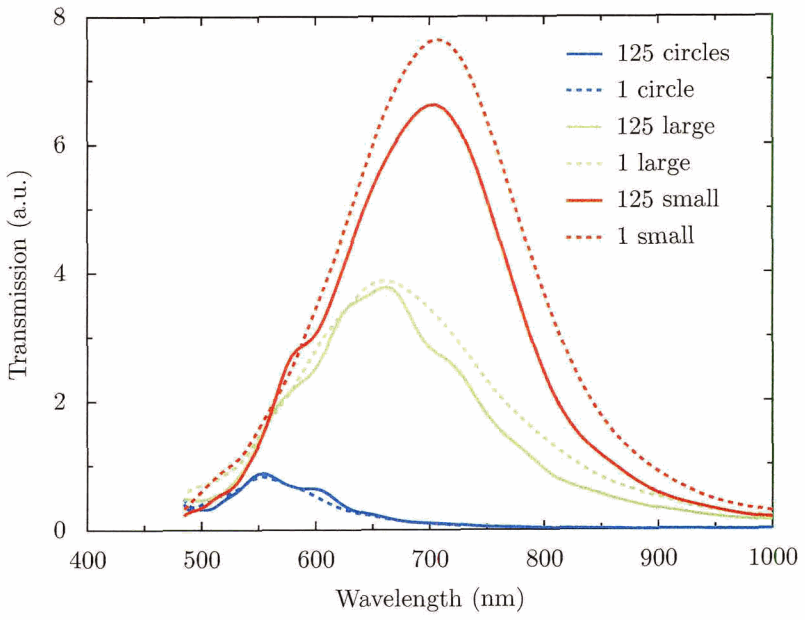


Figure 7.8: The transmission, normalized to aperture area, through a random arrangement of 125 apertures in a 10 x 10 μm^2 film. The transmission through a single aperture multiplied by 125 is shown for comparison (dashed lines).

transmission at certain wavelengths. An increased number of aperture interactions leads to transmission enhancement or degradation at a larger number of wavelengths, resulting the observed roughness.

The peak transmission, wavelength of peak transmission, and effective number of apertures for different numbers of apertures is summarized in Table 7.1. A randomly positioned set of circular apertures transmits more power than expected for both of the studied problems.

Small rectangles transmit significantly less than the expected amount of power in both cases. The reason for this is that the minimum distance between apertures is 350 nm, so the average distance between apertures is probably between 400 nm and 700 nm. As shown in Figure 7.5, this range of aperture spacing along the direction of polarization corresponds to significantly less that twice the single aperture power transmission for a pair of holes. Decreasing the minimum separation along this axis to approximately 180 nm, or increasing it to approximately 750 nm, is expected to result in more than expected power being transmitted.

Large rectangles show less variation in transmitted power than the small rectangles do. Large rectangles have been shown to transmit less power than small rectangles do. The surface waves excited from such apertures are therefore expected to be weaker, leading to less variation in transmission as the spacing between apertures varies. Additional simulations similar to those discussed in Section 7.2.2 can confirm this.

The transmission of light through a randomly positioned set of N apertures has been shown to be comparable to N times the transmission of light through a single aperture to within 18%. However, a great deal more work is required to fully understand the interactions between apertures in close proximity and the effects of random

Aperture type	# of apertures	Peak transmission (a.u.)	λ of peak (nm)	Effective # of apertures
Circular	1	0.007	553	1
	62	0.485	574	73.4
	125	0.884	553	133.6
Small rect	1	0.061	705	1
	62	3.12	705	51
	125	6.61	705	108
Large rect	1	0.031	661	1
	62	2.05	644	66.2
	125	3.78	661	121.8

Table 7.1: The peak transmission, wavelength of peak transmission, and effective number of apertures for different sizes and numbers of randomly positioned apertures

aperture placement on transmission through a thin metallic film.

Chapter 8

Conclusion

Contributions in this thesis have been made in two areas: namely new hybrid parallel FDTD code development and evaluation, and its use for solution of selected problems in nano-optics.

This thesis has introduced a new hybrid parallel finite-difference time-domain simulation code. Using MPI and OpenMP, the code has been found to scale to large numbers of processors. Good performance has been obtained on a variety of platforms, from desktops to supercomputers. The code embeds the Python scripting language, which is used to define problems to be simulated, and can be used to add new signals, excitations, and domain decomposition algorithms to the code.

Three major enhancements could be made to improve the robustness and utility of the code for very large problems. Many computing installations limit the wall clock time any one job is allowed to use to 24 hours. In order to process jobs that take longer than 24 hours, a facility for handling check pointing, writing the program state to disk and later reloading it, is required for systems that are not able to do check pointing automatically.

While the code can be used to simulate very large problems, the implementation of the Drude model uses a method which uses more memory than other methods. Switching to a different implementation of the Drude model would reduce the amount of memory required for a problem, making it possible to simulate even larger problems.

The domain decomposition algorithm can be improved to minimize the time spent communicating between processes and more evenly balance the memory and computational loads over the processes. The development of a performance model for the code would be useful to help predict the number of processors that could be effectively utilized, and to aid the domain decomposition algorithm.

In addition to the major items listed above, numerous smaller improvements can be made to the code. Detailed examination of the code performance on various processors could lead to insight into cache performance and pipelining, and potentially faster code. Vectorizing compilers could be used to take advantage of the vector processing units that are now built into some commodity processors. The performance of the result gathering and writing code has not yet been examined in detail and could be improved, particularly through the use of parallel I/O, which was introduced in MPI-2.

The code has been applied to study the transmission of light through sub-wavelength apertures at optical frequencies. Using the Drude model to capture the frequency dispersive properties of gold and silver, agreement with experimental data and theoretical models was obtained for the transmission of light through a single isolated aperture and an array of apertures. Important features of the transmission spectrum, such as resonance peaks and Wood's anomalies, were clearly and quantitatively elucidated.

The effects of polarization angle and basis rotation on the transmission through an array of ellipses and double holes have been explored. The polarization which

exhibits maximum transmission through an array of ellipses is found to coincide with the basis vector of the unit cell. In contrast, the polarization vector must be co-aligned with the lattice vector for maximum transmission through the array of double holes.

The transmission through a thin metallic film perforated by a set of randomly positioned apertures has been simulated. The transmission though N apertures was found to be within 18% of N times the transmission though a single aperture. This suggests that experimental measurements of the transmission through such films are approximately indicative of the transmission through a single aperture.

The transmission through a pair of rectangular apertures in close proximity to one another has also been simulated. If the apertures are very close together, the transmission through the pair is less than twice the transmission through a single aperture. As the separation between the apertures increases, the transmission was found to rise above twice the transmission of a single aperture, and then oscillate about and decay towards twice the transmission of a single aperture.

Future work should explore two apertures in close proximity in greater detail, and attempt to model the initial low transmission and oscillatory behavior of the transmission as the spacing between the apertures increases. Simulations of larger numbers of randomly positioned apertures with different minimum spacing should lead to a better understanding of the latent periodicity which may exist. The code itself performs well and while numerous optimizations and improvements can be made, it is capable of simulating large problems.

Bibliography

[1] P.B. Johnson and R.W. Christy. Optical constants of the noble metals. *Physical Review B*, 6(12):4370–4379, December 1972.

[2] Shih-Hui Chang, Stephen K. Gray, and George C. Schatz. Surface plasmon generation and light transmission by isolated nanoholes and arrays of nanoholes in thin films. *Optics Express*, 13(8):3150–3165, April 2005.

[3] top500. http://top500.org/.

[4] A. Degiron, H. J. Lezec, N. Yamamoto, and T. W. Ebbesen. Optical transmission properties of a signle subwavelength aperture in a real metal. *Optics Communications*, 239:61–66, 2004.

[5] L. Martín-Moreno, F. J.García-Vidal, H. J. Lezec, K. M. Pellerin, T. Thio, J. B. Pendry, and T. W. Ebbesen. Theory of extranordinary optical transmission through subwavelength hole arrays. *Physical Review Letters*, 86(6):1114–1117, February 2001.

[6] R. Gordon, M. Hughes, B. Leathem, K. L. Kavanagh, and A. G. Brolo. Basis and lattice polarization mechanisms for light transmission through nanohole arrays in a metal film. *Nanoletters*, 2005. submitted.

[7] T. W. Ebbesen, H. J. Lezec, H. F. Ghaemi, T. Thio, and P. A. Wolff. Extraordinary optical transmission through sub-wavelength hole arrays. *Nature*, 391:667–669, February 1998.

[8] K. S. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14:302–307, 1966.

[9] E. Burstein. Introductory remarks on polaritons. In Elias Burstein and Francesco De Martini, editors, *Polaritons: proceedings of the first Taormina Reasearch conference on the structure of matter*, pages 1–4, Taormina, Italu, October 1972. Pergamon Press, Inc.

[10] H. A. Bethe. Theory of diffraction by small holes. *The Physical Review*, 66(7):163–182, October 1944.

[11] H. F. Ghaemi, T. Thio, D. E. Grupp, T. W. Ebbesen, and J. H. Lezec. Surface plasmons enhance optical transmission through subwavelength holes. *Physical Review B*, 58:6779–6782, 1998.

[12] U. Schröter and D. Heitmann. Surface-plasmon-enhanced transmission through metallic gratings. *Physical Review B*, 58(23):419–421, December 1998.

[13] Laurent Salomon, Frédéric Grillot, Anatoly V. Zayats, and Frédérique de Fornel. Near-field distribution of optical transmission of periodic subwavelength holes in a metal film. *Physical Review Letters*, 86(6):1110–1113, February 2001.

[14] M. M. J. Treacy. Dynamical diffraction in metallic optical gratings. *Applied Physics Letters*, 75(5):606–608, August 1999.

[15] Y. Takakura. Optical resonance in a narrow slit in a thick metallic screen. *Physical review letters*, 86(24):5601–5603, June 2001.

[16] Fuzi Yang and J. R. Sambles. Resonant transmission of microwaves through a narrow metallic slit. *Physical Review Letters*, 89(6), August 2002. 063901.

[17] Stefan Enoch, Evgueni Popov, Michel Neviere, and Raymond Reinisch. Enhanced light transmission by hole arrays. *J. Opt. A: Pure Appl. Opt.*, 4:S83–S87, 2002.

[18] Reuven Gordon and Alexandre G. Brolo. Increased cut-off wavelength for a subwavelength hole in real metal. *Optics Express*, 13(6), March 2005.

[19] K. J. Koerkamp, S. Enoch, F. B. Segerink, N. F. van Hulst, and L. Kuipers. Strong influence of hole shape on extraordinary transmission through periodic arrays of subwavelength holes. *Physical Review Letters*, 92(18):183901, May 2004.

[20] H. J. Lezec, A. Degiron, E. Devaux, R. A. Linke, L. Martin-Moreno, F. J. Garcia-Widal, and T. W. Ebbessen. Beaming light from a subwavelength aperture. *Science*, 297:820–822, August 2002.

[21] Matthew J. Lockyear, Alastair P. Hibbins, and J. Roy Sambles. Surface-topography-induced enhanced transmission and directivity of microwave radiation through a subwavelength circular metal aperture. *Applied Physics Letters*, 84(12):2040–2042, 2004.

[22] A. A. Oliner and D. R. Jackson. Leaky surface-plasmon theory for dramatically enhanced transmission throught a subwavelength aperture, part i: Basic features. In *Antennas and Propagation, 2003 IEEE Society International Conference on*, volume 2, pages 1091–1094, July 2003.

[23] A. A. Oliner and D. R. Jackson. Leaky surface-plasmon theory for dramatically enhanced transmission throught a subwavelength aperture, part i: Leaky-wave antenna model. In *Antennas and Propagation, 2003 IEEE Society International Conference on*, volume 2, pages 1095–1098, July 2003.

[24] Akira Ishimaru. *Electromagnetic Wave Propagation, Radiation, and Scattering.* Printice Hall, 1991.

[25] Craig F. Bohren and Donald R. Huffman. *Absorption and scattering of light by small particles.* John Wiley & Sons, Inc., 1983.

[26] M.J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computing*, C-21:948–960, 1972.

[27] LAM-MPI. http. http://www.lam-mpi.org/.

[28] MPICH. http://www-unix.mcs.anl.gov/mpi/mpich/.

[29] Message passing interface forum. *mpi-2: extensions to he message-passing interface*, July 1997. http://www.mpi-forum.org/.

[30] Rofl Rabenseifner. Hybrid parallel programming on HPC platforms. In *Fifth European workshop on OpenMP, EWOMP '03*, Aachen, Germany, September 2003.

[31] C. Guiffaut and K. Mahdjoubi. A parallel FDTD algorithm using the MPI library. *IEEE Antennas and Propagation Magazine*, 43(2):94–103, April 2001.

[32] Mehmet F. Su, Ihab El-Kady, David A. Bader, and Shawn-Yu Lin. A novel FDTD application featuring OpenMP-MPI hybrid parallelization. In *33rd Intern Conf Parallel Proc (ICPP)*, pages 373–379, August 2004.

[33] Allen Taflove and Susan C. Hagness. *Computational Electrodynamics.* Artech House, Boston, second edition, 2000.

[34] T. L. Veldhuizen and M. E. Jernigan. Will C++ be faster than Fortran? In *Proceedings of the 1st International Scientific Computing in Object-Oriented Parallel Environments (ISCOPE'97)*, Lecture Notes in Computer Science. Springer-Verlag, 1997.

[35] Todd Veldhuizen. Blitz++. http://www.oonumerics.org/blitz/.

[36] Andrew Lumsdaine, Jeremy Siek, and Lie-Quan Lee. The matrix template library. http://www.osl.iu.edu/research/mtl/.

[37] Web3D consortium. http://www.web3d.org/.

[38] Python Software Foundation. Python. http://python.org/.

[39] Dave Abrahams. Boost.python. http://boost.org/libs/python/doc/index.html.

[40] David A. Patterson and John L. Hennessy. *Computer Organization & Design.* Morgan Kaufmann Publishers, Inc., San Francisco, California, second edition, 1998.

[41] Dennis M. Sullivan. Z-transform theory and the FDTD method. *IEEE Transactions on antennas and propagation*, 44(1):28–34, January 1996.

Appendix A

The Drude Model

The Drude model is a special case of the Lorentz model, which treats electrons as bodies bound elastically to heavy nuclei. This derivation is similar to that found in Ishimaru [24] and Bohren [25], but no assumptions about the form of the forcing function are made.

$$m\frac{d^2}{dt^2}\mathbf{x}(t) + b\frac{d}{dt}\mathbf{x}(t) = e\mathbf{E}(t) \tag{A.1}$$

This second order ordinary differential equation describes the position of an electron due to an applied electric field, where the local Mossotti field has been neglected. The Fourier transform of this equation is:

$$-\omega^2 m\mathbf{X}(j\omega) + j\omega b\mathbf{X}(j\omega) = e\mathbf{E}(j\omega) \tag{A.2}$$

The polarization $\mathbf{P}$ due to electron motion is $\mathbf{P} = Ne\mathbf{E}$, where N is the number of electrons per unit volume. Electric field density is $\mathbf{D} = \epsilon_0\mathbf{E} + \mathbf{P} = (1 + \mathbf{P}/\epsilon_0)\epsilon_0\mathbf{E}$.

Let $\omega_p = \frac{Ne^2}{m\epsilon_0}$ and $\gamma = b/m$.

$$\epsilon_r(j\omega) = 1 + \frac{\omega_p}{-\omega^2 + j\omega\gamma} \tag{A.3}$$

Equation A.3 represents a transfer function relating $\mathbf{D}$ and $\mathbf{E}$. Partial fraction expansion and inverse Fourier transform gives the impulse response.

$$\epsilon_r(j\omega) = 1 + \frac{\frac{\omega_p^2}{\gamma}}{j\omega} - \frac{\frac{\omega_p^2}{\gamma}}{j\omega + \gamma} \tag{A.4}$$

$$\epsilon_r(t) = \delta(t) + \frac{\omega_p^2}{\gamma}u(t) - \frac{\omega_p^2}{\gamma}e^{-\gamma t}u(t) \tag{A.5}$$

The impulse response can now be discretized and transformed to the z domain. The sampling rate for discritization is the time step size for the FDTD simulation. If the sampling rate is to low, aliasing in the z domain can result. This introduces another constraint on the time step size; it must be small enough that aliasing is negligible.

$$\epsilon_r(\Delta tn) = \delta(\Delta tn) + \frac{\omega_p^2}{\gamma}u(\Delta tn) - \frac{\omega_p^2}{\gamma}e^{-\gamma\Delta tn}u(\Delta tn) \tag{A.6}$$

$$\epsilon_r(z) = 1 + \frac{\omega_p^2}{\gamma}\frac{1}{1 - z^{-1}} - \frac{\omega_p^2}{\gamma}\frac{1}{1 - e^{-\gamma\delta tn}z^{-1}} \tag{A.7}$$

This can now be substituted into $\mathbf{D}(z) = \epsilon_0\epsilon_r(z)\mathbf{E}(z)$. The rest of the derivation is the same as that given by Sullivan [41].