University of Alberta

Modelling and Fabrication of Near-Field Optical Systems

by

Anthony Dechant 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

Department of Electrical and Computer Engineering

Edmonton, Alberta

Fall 2004

Your file Votre référence
ISBN: 0-612-95733-0
Our file Notre référence
ISBN: 0-612-95733-0

Acknowledgements

I would first like to thank all those who helped me along the way, who provided me with useful insights and advice, and who assisted me when I ran into the roadblocks that are so common in experimental science. I would also like to express my appreciation to my family for their continued support in all my scholastic endeavours. Finally, I would like to thank Karmon Helmle for putting up with me during the past two years, for being my personal secretary when needed and for always being there to encourage me when everything didn't happen to go as planned.

Table of Contents

List of Figures

List of Acronyms

AFM	Atomic Force Microscope
BOE	Buffered Oxide Etch
CW	Continuous Wave
DRIE	Deep Reactive Ion Etching
EM	Electromagnetic
FIB	Focussed Ion Beam
FDTD	Finite-Difference Time-Domain
FWHM	Full-Width Half-Maximum
ICP	Inductively Coupled Plasma
RIE	Reactive Ion Etching
NSOM	Near-field Scanning Optical Microscope
SP	Surface Plasmon
TM	Transverse Magnetic
TMAH	Tetra-methyl Ammonium Hydroxide

Chapter 1: Introduction

1.1 Thesis Motivation

High resolution imaging has long been an extremely important tool in understanding the fundamental nature of the world around us. For centuries, disciplines including biology, chemistry, and engineering have relied on traditional optical techniques for their imaging needs. Given their extensive historical development, simplicity, and cost effectiveness, optical imaging methods have, until recently, been the predominant standard in research and development. Unfortunately, there are inherent limits to the resolution obtainable with conventional optics. Specifically, the spatial resolution of traditional optical techniques is limited to half of the wavelength of the light used [1]. When using radiation in the visible gamut of the electromagnetic spectrum, the fundamental limit to resolution thus lies between 200 and 300 nm. In order to attain higher resolutions, researchers must either use shorter and shorter wavelengths, or else they must turn to alternative imaging technologies.

The natural constraints imposed on conventional optics have, therefore, motivated the development of higher resolution systems including the scanning electron microscope, and transmission electron microscope. More recently, the evolution of scanning probe microscope systems such as the scanning tunneling microscope and the atomic force microscope has had a direct impact on image resolution. Using these modern methods, investigative imaging at the atomic scale is not uncommon; however, the gains in resolution have not come without a price. Much of the information inherent to conventional optical techniques is no longer available. Optical contrast mechanisms, spectroscopic capabilities, and the high temporal resolution once accessible via all optical methods have been discarded in the pursuit of higher resolution. Moreover, many of these

new techniques suffer from other problems. Scanning tunneling microscopy, for example, necessitates a conductive sample thereby precluding most biological applications. Atomic force microscopy can be used in ambient conditions and yields atomic-scale topographical information, yet chemical and compositional information are not retrievable. Scanning electron microscopy requires not only a conductive, but also a vacuum compatible sample due to the relatively short mean free path of the electron in air. Thus, researchers are once again turning to novel all-optical imaging techniques. Near-field nano-optics is one such technology. Still in its infancy, near-field imaging relies on collecting from a sample, light that has been confined at the nanometer scale. All of the contrast mechanisms inherent to traditional optical imaging are maintained while still acquiring images at the molecular level. Because of the multitude of advantages all optical imaging offers, near-field nano-optics should prove to be a viable alternative to modern imaging techniques.

1.2 Thesis Objectives

The goal of this thesis is to investigate the interaction of light in the near-field with nanometer scale structures. To achieve this, a prototype near-field optical probe is constructed. Along with a numerical model used to predict the operation of the near-field probe and to determine the propagation characteristics of ultrashort pulsed light in the near-field, the fabrication of the near-field probe is described.

1.3 Thesis Organization

This thesis is composed of five chapters and four appendices that outline the completed experimental and numerical work. Chapter 2 describes the fundamental principles of near-field optics and provides a review of previous experiments on the subject. Chapter 3 entails the development of a theoretical model for the time-dependant propagation of light through metal and dielectric nano-structures. A finite-differencing scheme in time-domain is employed in conjunction with a formalism that accurately describes the interaction of light with metallic structures. This model is employed to predict the operation of the near-field optical probes as well as to determine the propagation characteristics of ultrashort pulsed light in metallic nano-structures. Chapter 4 illustrates the fabrication process for the near-field optical probes. Chapter 5 provides a conclusion of the completed work as well as future avenues to explore in using the newly developed near-field probes in experimental work. All relevant C++ code written during this thesis is presented in Appendix A, as well as the mask designs in Appendix B.

1.4 References

1. Abbe, E., *Beiträge zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung.* Arch. Mikrosk. Anatomie, 1873. **9**: p. 413.

Chapter 2: Background

2.1 Diffraction Limit

The limits to optical resolution have been well known for over a century. This diffraction limit stems from the spot size to which light can be focused by conventional lens-based optical systems. At the focal point of a lens system, light tends to form a pattern of concentric rings instead of a well resolved circle. Known as the Airy disk, its characteristics were first described by Ernst Abbe in 1873[1]. He stated that the distance between the point of highest intensity and the first bright ring is given by

$$\Delta x = 0.61\lambda / NA. \qquad (2.1)$$

Here, λ is the wavelength of the light used, and NA is the numerical aperture of the objective. Although Abbe was the first to describe the intensity profile of focused light, it was Lord Rayleigh who actually interpreted his results as a limit to attainable resolution. Rayleigh's criterion maintains that two separate objects are resolvable only if they are separated by a distance greater than that specified by equation 2.1. Since modern oil immersion lenses can have numerical apertures as high as 1.4, this leads to the familiar simplification of $\Delta x = \lambda/2$. In other words, the maximum theoretical resolution attainable is limited to half the wavelength of the light used. Practically this limit may be much larger. Experimental considerations such as vibrational noise caused my mechanical oscillations in the optical system, and optical aberrations caused by defects in the imaging system often conspire to artificially limit image resolution.

2.2 Subwavelength Diffraction

Although radially polarized light can be focused to a spot size ($0.16\lambda^2$) [2], in order to circumvent the inherent limitations to optical imaging, near-field light is usually

employed. The light field near the surface of an object actually contains more spatial frequencies and hence more spatial information than can be gathered by using a far-field lens system, as only the spatial frequencies that reach the imaging lens are observable [3]. These are the propagating, low-frequencies. Higher spatial frequencies exist at the sample surface, but they are non-propagating due to their spatial localization. These evanescent modes, known as the near-field, decay exponentially from the surface, so they are undetectable by conventional optics. Light propagating away from the surface of a sample effectively limits the resolution of the image obtained since all the fine details are filtered out.

The idea of using the near-field in achieving higher resolution optical measurements is not new. Synge [4], in 1928, published a detailed overview of an optical imaging system that would not be subject to the limitations imposed by diffraction. He proposed an imaging system in which a subwavelength aperture is created in an optically opaque screen. Light incident upon the backside of the screen is transmitted through the aperture. As depicted in Figure 2.1, the transmitted light would be confined by the dimensions of the aperture, and, if a sample were placed within one wavelength of the aperture, a correspondingly small area of the sample could be imaged without a reduction in resolution.

The feasibility of Synge's proposal was first theoretically demonstrated by Bethe in 1944. By studying the propagation of light through a subwavelength aperture in an infinitely thin, perfectly conducting plane, Bethe [5] managed to convey an accurate representation of the electromagnetic (EM) near-field. Prior to this, diffraction studies, both experimental and theoretical, concentrated mainly on the interaction of light with

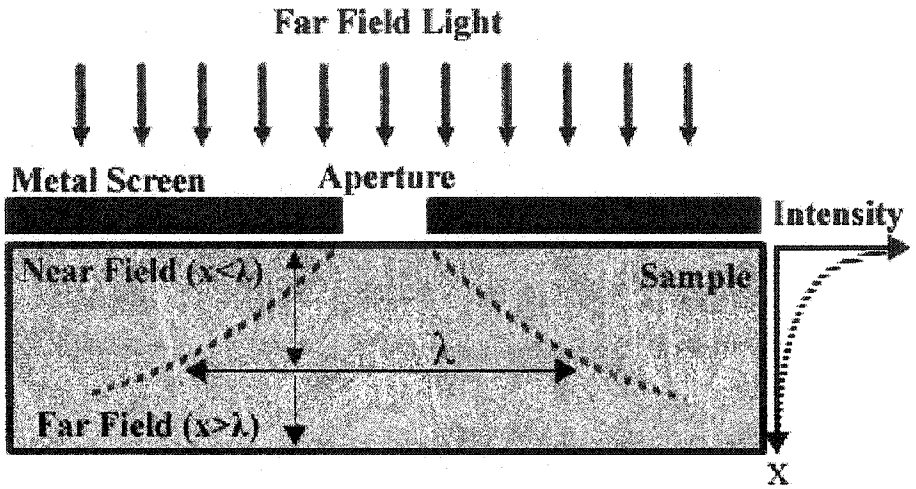


Figure 2.1: Schematic representation of Synge's proposed near-field imaging device in which light is confined and transmitted through a subwavelength aperture and subsequently collected to form a high resolution image of the sample.

objects larger than the wavelength employed [6]. However, these methods failed to match the boundary conditions for Maxwell's equations at the aperture and they were thus unsuitable for subwavelength studies. Specifically, they were unable to meet the criterion that the tangential component of the electric field at the screen must be zero. This boundary condition is necessary as electric field lines must meet a perfect conductor perpendicular to its surface. Although not significant when dealing with large apertures, this criterion has a considerable impact when the aperture diameter, a, is reduced to subwavelength dimensions as in Figure 2.1. Bethe was able to overcome these restrictions and match the boundary conditions by employing a fictitious magnetic charge, ρ_m, and current, $\vec{J}_m$, at the surface of the screen as seen in Figure 2.2. Further refinement and correction of Bethe's theory by Bouwkamp [7] resulted in consistent expressions for both the far and near-field light emanating from a subwavelength aperture. The electric, $\vec{E}_{ap}$, and the magnetic, $\vec{B}_{ap}$, field vectors as well as the vector, $\vec{A}_m$, and scalar, Φ_m, magnetic potentials are

$$\vec{E}_{ap} = \nabla \times \vec{A}_m \tag{2.2}$$

$$\vec{B}_{ap} = -ik_o \vec{A}_m - \nabla \Phi_m \tag{2.3}$$

$$\vec{A}_m = -\int \vec{J}_m \frac{e^{ik_o r}}{r} dA \tag{2.4}$$

$$\Phi_m = -\int \rho_m \frac{e^{ik_o r}}{r} dA \tag{2.5}$$

where the integrals are calculated over the area of the aperture, r is the distance from the source point in the aperture to the field point, and k_o is the wave vector. If a plane EM

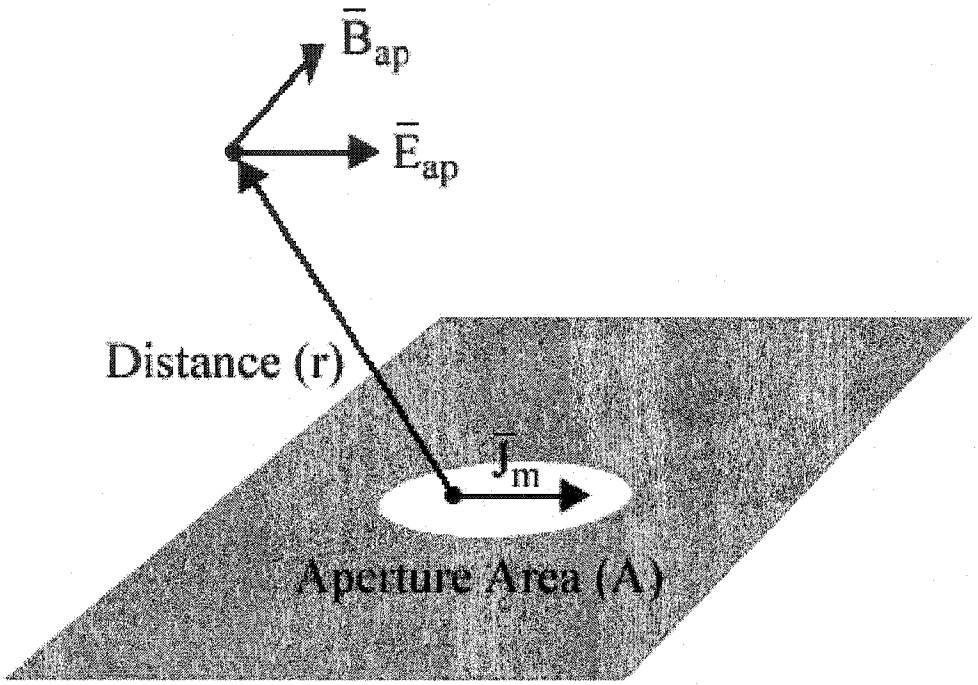


Figure 2.2: Bethe-Bouwkamp model displaying an infinitely thin conductive plane with a subwavelength aperture.

field linearly polarized in the x-direction is incident upon the screen then the magnetic charge density and current density at a point (x,y,z) in the aperture are found to be

$$\rho_m = \frac{2yE_o}{\pi^2\sqrt{a^2 - x^2 - y^2}} \tag{2.6}$$

$$J_{mx} = -\frac{2ik_o xyE_o}{3\pi^2\sqrt{a^2 - x^2 - y^2}} \tag{2.7}$$

$$J_{my} = \frac{2ik_o E_o(2a^2 - x^2 - 2y^2)}{3\pi^2\sqrt{a^2 - x^2 - y^2}} \tag{2.8}$$

$$J_{mz} = 0 \tag{2.9}$$

This simple, but unphysical, analytical model has been widely used to demonstrate that near-field light emanating from an aperture remains collimated for a distance approximately equal to the aperture radius [8]. By corroborating the theory of super-resolution image formation, a subwavelength image can be formed if this arrangement can be implemented and a probe can be positioned within one aperture radius of the sample.

2.3 Surface Plasmons and the Diffraction Limit

Surface plasmons (SPs) also provide a mechanism to better the diffraction limit. SPs coupled to metallic nano-structures inherently overcome the diffraction limit since the coupled optical mode is transformed into a non-radiating mode confined to the dimensions of the nano-structure. As the size of these nano-structures is typically far less than half the wavelength of the coupled light, the diffraction limit is no longer significant. However, in order to appreciate this confinement process, the mechanisms of SP creation,

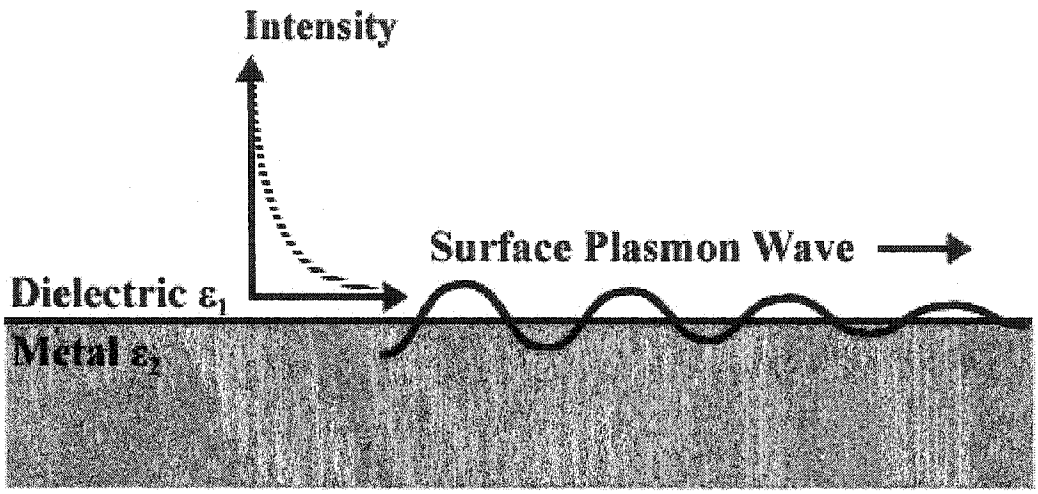


Figure 2.3: Depiction of a surface plasmon wave propagating along a metal-dielectric interface. The intensity of the electric field into the dielectric follows an exponential decay and the wave suffers significant damping as it propagates. The propagation length is roughly 13 μm in silver at 800 nm.

propagation and absorption must be understood. To that end, the following section provides a brief overview of surface plasmon theory.

2.3.1 Surface Plasmon Theory

Surface plasmons are collective electron oscillations that propagate along the surface of a metal-dielectric interface [9]. They have exponentially decaying electric field components tangential to the interface, as depicted in Figure 2.3, and they are strictly composed of transverse magnetic (TM) waves. Along with the dispersion curve of a single photon, the SP dispersion relation, given by

$$k_{sp}^2 = \frac{\omega^2}{c^2}\frac{\varepsilon_1(\omega)\varepsilon_2(\omega)}{\varepsilon_1(\omega)+\varepsilon_2(\omega)}, \tag{2.10}$$

is plotted in Figure 2.4. Here k_{sp} is the SP wave vector, ω is the frequency of light, and $\varepsilon_1(\omega)$ and $\varepsilon_2(\omega)$ are the frequency dependent dielectric functions of the metal and dielectric materials respectively. The SP frequency, ω_{sp}, corresponds to the asymptotic line when $\varepsilon_1(\omega) = \varepsilon_2(\omega)$ as the wave vector approaches infinity. At this point, a metal with dielectric function

$$\varepsilon_2(\omega) = 1 - \frac{\omega_p^2}{\omega^2}, \tag{2.11}$$

will have an SP frequency equivalent to

$$\omega_{sp} = \frac{\omega_p}{\sqrt{\varepsilon_1 + 1}} \tag{2.12}$$

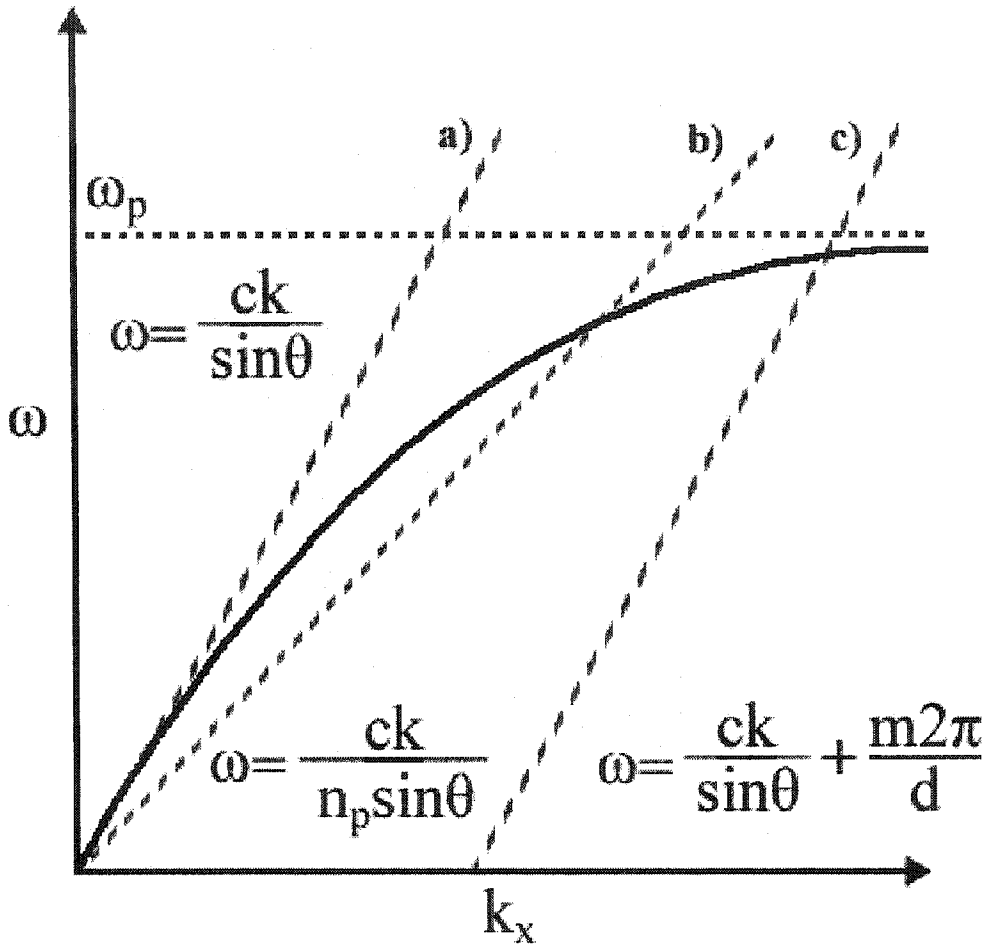


Figure 2.4: The surface plasmon dispersion curve (solid line) is plotted along with the dispersion curves for a photon in free space (a), a photon passing through a dielectric material with refractive index, n_p, at an angle, θ, (b), and a photon impingent at an angle, θ, on a grating with periodicity, d and diffracted order m (c).

where $\omega_p = \sqrt{\frac{ne^2}{\varepsilon m_e}}$ is the plasma frequency of the metal and n is the number density of electrons in the metal. This results in $\omega_{sp} = \omega_p / \sqrt{2}$ when considering SP propagation at a metal/vacuum interface. Also shown in Figure 2.4 is the lack of intersection between the dispersion curves of the photon and the surface plasmon. Thus, direct coupling between photons and SPs is impossible since momentum and energy conservation are not satisfied. Special geometries must, therefore, be employed and are discussed in the following section.

2.3.2 Coupling to Surface Plasmon Waves

Fortunately, this momentum conservation requirement can be satisfied in a variety of ways, and coupling between photons and SPs is indeed possible. In order for coupling to occur, two conditions must be met. First, a portion of the incident electric field must be parallel to the SP's direction of propagation along the metal-dielectric interface in order to drive the SP oscillation. This condition is easily fulfilled by employing TM polarized incident light. In this case, the electric field vector lies in the plane formed by the light's Poynting vector and the normal vector to the interface. Secondly, momentum and energy must be conserved as the photon couples to the SP. There exist two common techniques to create this wave vector matching condition: total internal reflection coupling and grating coupling.

The total internal reflection coupling technique, also known as the Kretschmann geometry [10], entails placing a material with a refractive index, $n_p > 1$, in contact with the metallic surface as shown in Figure 2.5. TM-polarized light passing through the

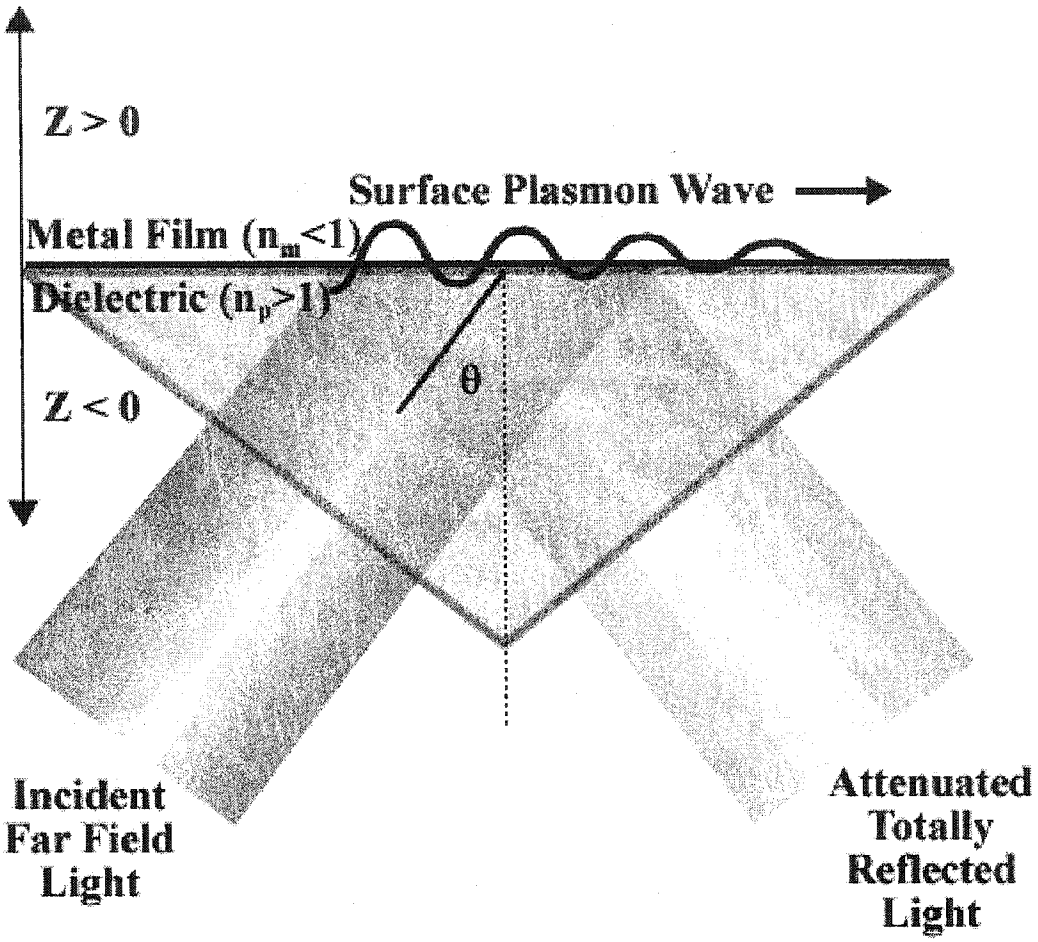


Figure 2.5: Kretschmann geometry for surface plasmon excitation. TM-polarized far-field light is incident at an angle larger than that for total internal reflection. The wave vector of the incident light parallel to the surface is coupled to a surface plasmon wave, and the reflected light is attenuated.

material at an angle larger than that for total internal reflection contains a longitudinal wave vector, k_p, parallel to the interface equal to

$$k_p = \frac{\omega}{c} n_p \sin\theta . \tag{2.13}$$

By tuning the incident angle, θ, the photon dispersion curve is shifted to intersect that of the SP as illustrated in Figure 2.4. At the resonance angle, incident light is coupled to the propagating surface wave resulting in attenuated reflection.

Similarly, the grating coupling technique, displayed in Figure 2.6, employs a ruled grating at the metal-dielectric interface. Incident light impingent upon the grating is subsequently scattered, thereby modifying the wave vector enough to match that of the SP [9]. The resulting wave vector for the diffracted light is given by

$$k_g = \frac{\omega}{c}\sin\theta + \frac{m2\pi}{d} . \tag{2.14}$$

where d is the periodicity of the grating, θ is the angle of incident and m is a positive integer referring to the diffraction order. The shift in the photon dispersion curve, also depicted in Figure 2.4, is due to additional wave vector components created by the periodicity of the grating. As clearly demonstrated, SP coupling is possible in either geometry, although coupling efficiencies vary.

2.3.3 Surface Plasmon Propagation

Once a SP has been launched, the electric field on a smooth metal surface propagating in the x-direction is governed by [9]

$$E_{sp}(x,z) = E_o e^{ik_{sp}x - k_z|z|} \tag{2.15}$$

where the field decay factor in the tangential direction, k_z is

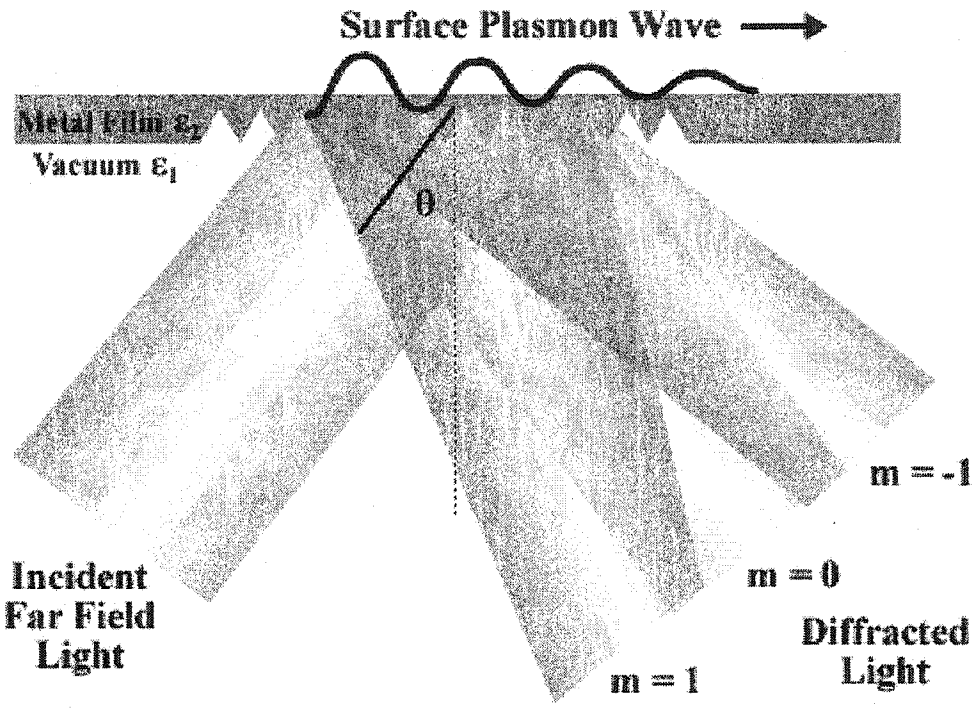


Figure 2.6: Grating geometry for surface plasmon coupling. The momentum of the incident photons is modified by the grating structure facilitating coupling to surface plasmon waves.

$$k_z^2 = k_{sp}^2 - \frac{\omega^2}{c^2} x \begin{cases} \varepsilon_1 \text{ in dielectric} \\ \varepsilon_2 \text{ in metal} \end{cases} . \quad (2.16)$$

Since at visible frequencies the real portion of the dielectric constant of the metal, ε_2, is usually larger than that of the adjacent dielectric, ε_1, the penetration depth of the SP in the metal is comparatively short. However, since the electric field is concentrated at the surface of the metal layer, it undergoes significant enhancement [9]. This enhancement is curbed only by the Ohmic losses of the metal and other losses associated with surface roughness. Moreover, the Ohmic losses in the metal also impose a limit on the propagation length of the SP, L_{sp}, along the surface. This propagation length is directly related to the imaginary portion of the SP wave vector (Eq. 2.10)

$$L_{sp} = [2\,\mathrm{Im}(k_{sp})]^{-1}. \quad (2.17)$$

As will be seen in the next section, however, other processes exist which further limit the SP lifetime.

2.3.4 Localized Surface Plasmons

Although SPs are supported on planar metallic surfaces, localized SPs can also exist as bound electron plasmas in small metallic particles or nanometer scale metallic structures [11]. These localized surface plasmons are typically confined to non-planar geometries much smaller than the wavelength of the light employed and can be excited by light of an appropriate frequency irrespective of the light's wave vector due to scattering off of the metallic particle. Because of this fact, localized SPs also decay due to light emission. Moreover, localized SPs can be excited by propagating SPs on planar geometries due to surface roughness or local subwavelength scale topologies [12, 13]. This also leads to

significantly enhanced electrical fields due to the localization of the SP in an extremely small volume [14] as well as SP re-emission of light due to secondary scattering events [12]. Recent experimental and theoretical efforts have been made demonstrating the feasibility of localized SPs in creating SP waveguides and nano-devices [15, 16]. Since SPs contain the inherent ability to localize, enhance, and transport EM energy below the diffraction limit, they will undoubtedly have a pronounced effect on near-field photonics and near-field optical imaging in particular. Simulations of this effect will be further discussed in Chapter 3.

2.4 Near-field Scanning Optical Microscopy

In 1972, Ash and Nicholls [17] successfully demonstrated the use of near-field light as an imaging tool. Using microwaves with a 3cm wavelength, they were able to discern periodic features in a metal grating with a resolution greater than $\lambda/60$. Visible wavelength near-field microscopy, however, was still not feasible since methods to control aperture size and probe-sample distance did not exist until a decade later. Along with the development of several other scanning probe microscopy techniques, IBM's Zurich laboratory was responsible for the development of the true near-field scanning optical microscope (NSOM) [18, 19]. Near-field scanning optical microscopy characteristically involves the placement of a subwavelength probe into close proximity with the sample to be imaged. As this probe is raster-scanned across the sample surface, a photodetector acquires the optical signal created by the interaction of the near-field light with the sample's nanometer scale features. Although near-field light cannot be used to image the sample directly, the interaction of the near-field with the sample is easily detected.

2.4.1 Imaging Modes

There are various designs employed in near-field scanning optical microscopy. The first method, known as collection mode and shown in Figure 2.7(a), involves illuminating a sample with far-field light [20]. Introducing a subwavelength apertured probe into the near-field region effectively introduces a conduit that guides light towards the detector. Since the detected light originates from an area approximately equivalent to the aperture size, a super-resolution image is formed.

Conversely, the following two methods utilize the subwavelength aperture as a source of near-field light. Depending upon the location of the detector, these two methods are known commonly as the illumination [21] and reflection [22] modes. The illumination mode typically employs a transparent sample, as the scattered near-field light emanating from the aperture is detected directly below the probe as seen in Figure 2.7(b). In reflection mode, scattered light is detected above an opaque sample as portrayed in Figure 2.7(c).

Finally, a subwavelength aperture can also be used as both source and detector as seen in Figure 2.7(d). Known as illumination-collection mode, light passes through the aperture illuminating the sample in the near field. Light reflected by the sample is then collected by the probe, and passed to a detector resulting in an image of extremely high resolution.

2.4.2 Image Resolution

Theoretically, NSOM resolution is limited only by the size of the subwavelength aperture used; however, this is not the case in practice. Several factors serve to degrade NSOM resolution. Foremost among these is the effect of surface force interactions with the

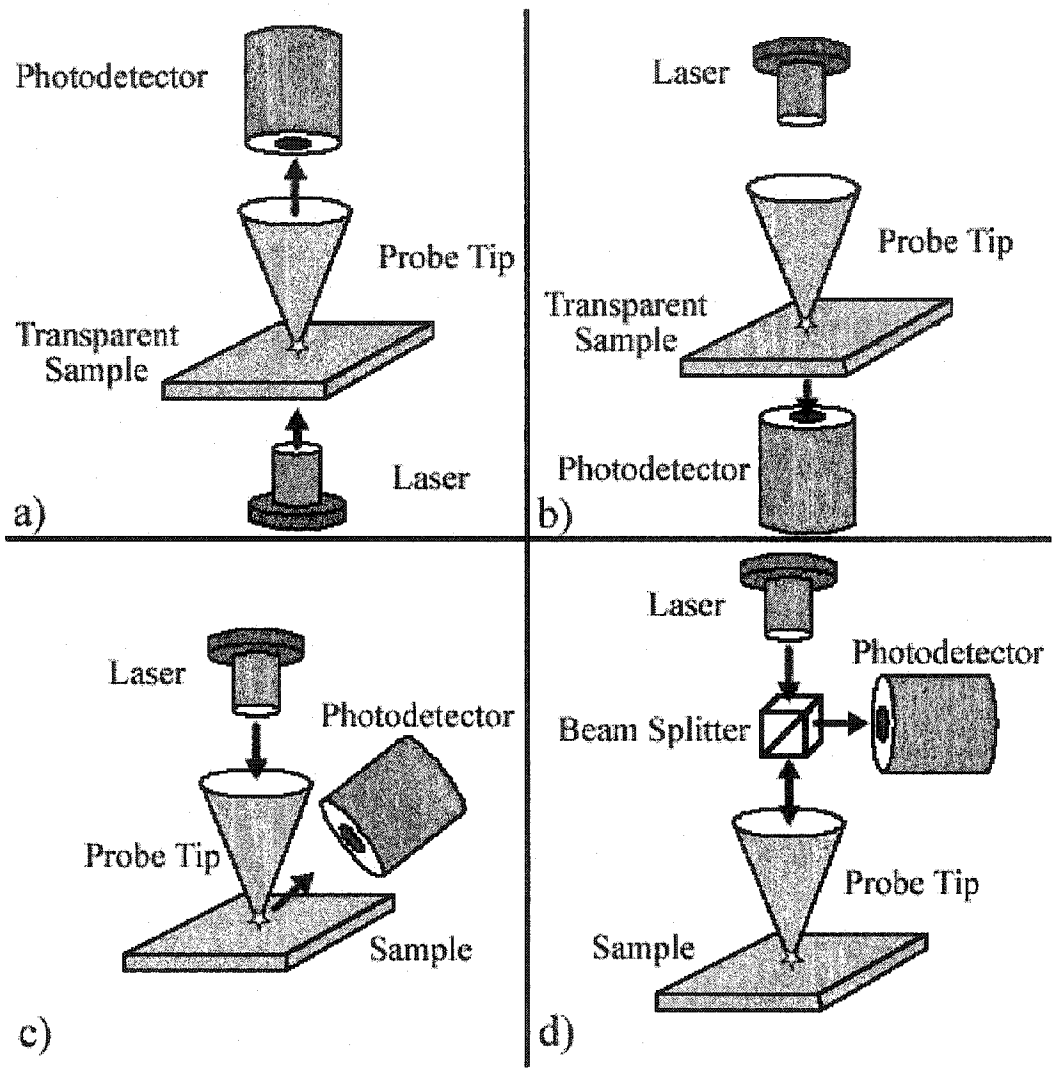


Figure 2.7: In collection mode (a), the probe tip is used to collect light from a transparent sample. Illumination mode (b) consists of passing light through the probe tip and collecting the scattered light on the far side of a transparent sample. Reflection mode (c) entails gathering the scattered light from an opaque sample, whereas illumination-collection (d) mode employs the probe tip as both collector and distributor of near-field light.

optical probe. The evanescent electromagnetic field at the aperture is highly sensitive to fluctuations in the vertical positioning of the probe [23]. Since maintaining a constant distance between probe and sample is rarely feasible due to mechanical vibrations, image artifacts often result. Secondly, image resolution and optical transmission are tightly linked. There exists a significant tradeoff between resolution and light throughput, with the transmitted power decreasing as the inverse fourth power of the tip radius [7]. Given these fundamental restrictions, apertured probes are currently limited to resolutions between 50 and 100 nm.

2.4.3 Control Mechanisms

As mentioned previously, accurately measuring the near-field at the surface of a sample requires maintaining the probe in close and constant proximity to the sample. Several mechanisms and operating modes exist to regulate tip-sample distance. Constant height mode, portrayed in Figure 2.8(a), involves scanning the near-field probe parallel to the sample at a fixed height. No feedback mechanisms are required since the tip-sample distance does not vary along a scan line. Constant distance mode, shown in Figure 2.8(b), ensures that the tip follows the topographical contours of the sample by continually adjusting the tip height. In this case tunneling current, atomic force, or shear force are used as a feedback mechanism to regulate the tip height.

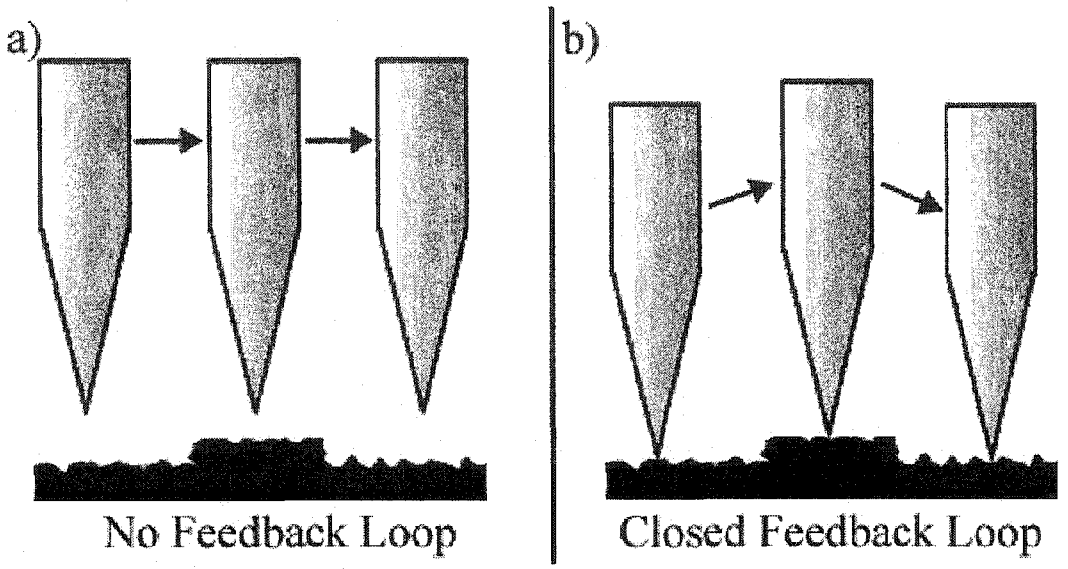


Figure 2.8: In constant height mode (a), the tip is maintained at a fixed height in space whereas in constant distance mode (b), the tip-sample distance remains fixed thereby following the sample's contour.

Tip-sample distance is regulated via three common mechanisms. For a conductive sample, the gap between probe and sample can be adjusted by monitoring the tunnel current as depicted in Figure 2.9(a) [20]. The tunnel current, typically several nA, exponentially depends on the tip-sample distance. A feedback circuit is employed to control the piezo driver as the probe is scanned across the surface. At each scanning point (x,y) the probe is raised or lowered to maintain a constant tunnel current and accordingly a constant distance. An optical signal is also collected at each scanning point providing a nanometer scale image of the surface.

For fiber based near-field probes, a shear-force feedback mechanism is characteristically used to regulate the tip-sample distance [24]. The probe is mounted on a piezoelectric or tuning fork component and vibrated at its mechanical resonance as seen in Figure 2.9(b). This vibration is damped due to shear force interaction as the probe is brought into close proximity (<10 nm) to the sample. Again a feedback circuit is used to monitor the vibrational frequency, and adjust the tip height as needed.

Typically used with a cantilevered probe, the third mechanism, portrayed in Figure 2.9(c), entails measuring the atomic force interactions between the nanometer scale probe and sample. Based on the conventional atomic force microscope [25], the forces monitored can include Van der Waals, electrostatic, capillary or even ion repulsion forces depending on the chemical composition of sample and probe. In a manner similar to the shear force damping of vibrational amplitude, the atomic force interactions cause the cantilever to bend as it approaches the sample. Laser light reflected off the bent cantilever is deflected towards a four quadrant photodetector and translated

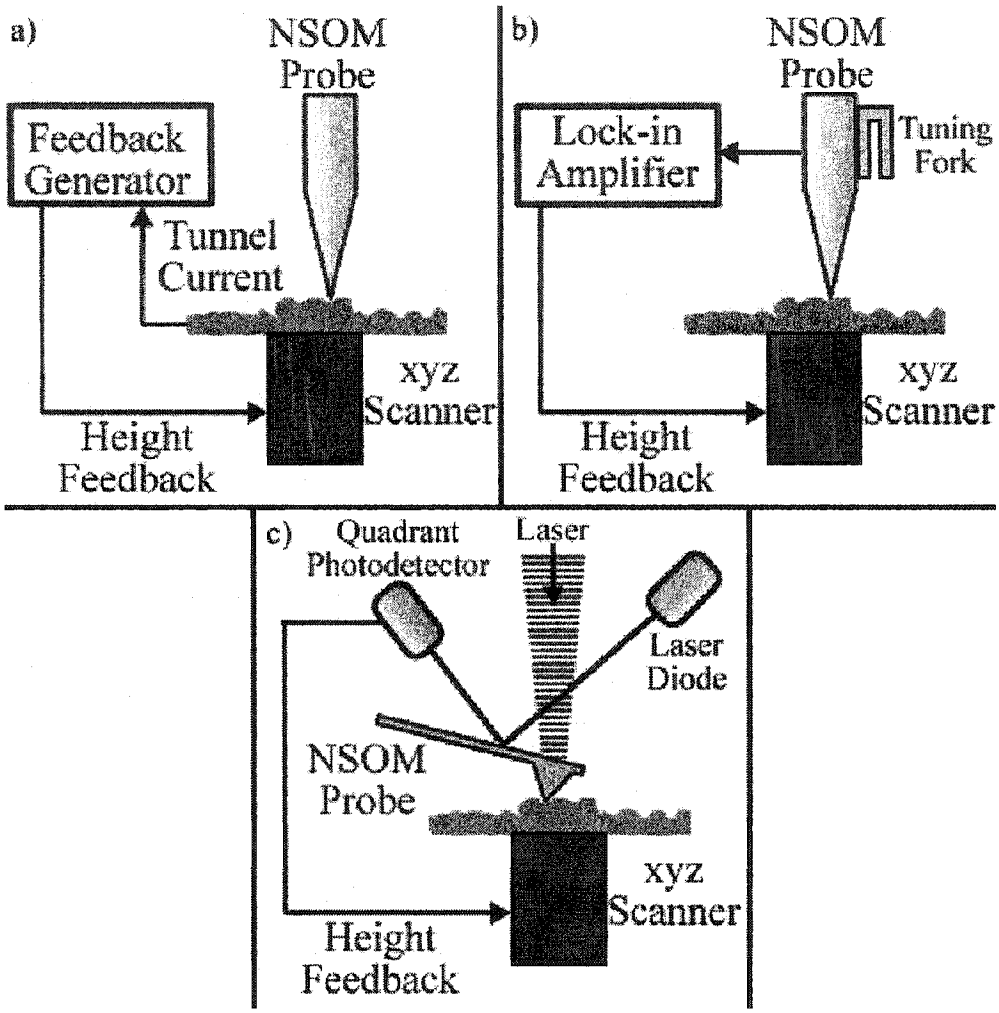


Figure 2.9: Control mechanisms for a near-field optical microscope come in a variety of forms. Monitoring the tunneling current (a) between sample and tip, the vibrational frequency of the tip (b), or the atomic force interaction (c) between sample and tip are the three foremost regulation techniques.

into a relative vertical position. A feedback circuit is again used to adjust the drive voltage on a vertical piezo to ensure that the probe remains a constant distance from the sample. In contact mode, the signal from the quadrant photodiode is continuously compared to a user specified reference point via a differential amplifier. The output from this comparison is a correction voltage sent to the vertical piezo in order to adjust the vertical position of the probe tip.

2.5 Fiber Optic Near-Field Probes

Systems based on fiber optic probes represent the most common type of near-field scanning optical microscope. Although early designs were based on heat-pulled micropipettes, these probes tended to be unreliable, and produced low-grade images [18, 19]. The most successful design, which is in common use today, was introduced by Betzig, et. al. [20] and incorporates a metal-coated tapered optical waveguide. These probes are created by heating and pulling single-mode telecommunications grade optical fibers to a sharp point several tens of nanometers wide by using an automated micropipette puller. This method results in highly reproducible tip diameters and tapers, although the fabrication scheme still suffers from low fabrication throughput as only a single probe can be created at any given time.

Several studies have been performed in order to determine the optimal tip and taper dimensions [26-28]. Maximum optical transmission through the aperture is achieved for tapered fibers having designs similar to a compound parabolic concentrator which is typically used for solar light collection. However, maximum optical efficiency is not always paramount since a delicate balance exists between optical throughput and the damage threshold of the metal coating.

Typically, an aluminum film is utilized due to its small skin depth of ~13nm for λ=500nm; however, the tendency towards grain formation in aluminum thin films results in porous films that allow light to escape; therefore, much thicker films, on the order of hundreds of nanometers, are necessary. These thicker films, although still porous, ensure that no cavities exit that penetrate to the underlying fiber. This opaque metal coating is necessary for light confinement since light tends to leak from the sides of the fiber tip [19, 23, 29]. This light leakage occurs when the diameter of the fiber is reduced beyond the mode cutoff for the waveguide. Typically this cutoff effect results in a typical transmission efficiency of 10^{-6} to 10^{-5} for a 100 nm aperture [30]. When considering aperture diameters below 100 nm, only the TE_{11} mode can approach the tip, whereas all other modes are typically cutoff and absorbed by the metallic layer leading to thermal damage and destruction [31, 32].

The low output efficiency of pulled fiber optic probes leads to several problems when considering low signal applications. This has motivated the development of alternative techniques for fiber probe manufacturing. Recently, fiber probe design is increasingly reliant on chemical etching processes since wet chemical etching provides greater control over the quality and structure of NSOM fiber probes [33-35]. Moreover, chemical etching facilitates the batch fabrication of NSOM fiber tips. The process characteristically involves etching the optical fibers in a solution of buffered hydrofluoric acid (BHF),

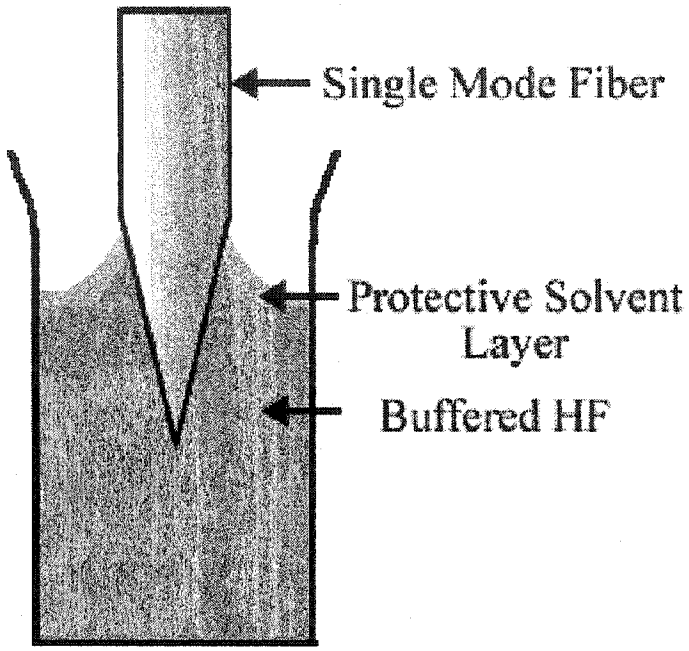


Figure 2.10: A fiber based NSOM probe is fabricated from a single mode optical fiber by etching in a solution of buffered hydrofluoric acid. The cone taper is formed due to the meniscus created at the protective layer as the fiber is pulled from the etchant.

NH_4:HF:H_2O, that is covered by a protective liquid layer. The protective layer is commonly composed of a solvent such as toluene, 1-Octanethiol or hexadecane [36-38]. The sharp fiber tip is etched via the formation of a meniscus at the protective layer as seen in Figure 2.10. Mononobe et al. [34] have a produced fiber probes with a stepped taper. The initial tapering of the probe is achieved via etching for 50 minutes in a 4:1 solution of 50% hydrofluoric acid and 95% sulfuric acid covered with a solvent layer of dimethylsilicone oil. This results in a 20° taper angle. Subsequent etching in a solution of 10:1 buffered hydrofluoric acid for 120 min results in a secondary taper on the probe with an angle of 10°. Therefore, by varying the composition of the protective layer, the etching time, and the etchant concentration, numerous taper angles can be formed resulting in probes tailorable to specific applications.

Besides etching, other microfabrication processes have been used to improve the performance of fiber probes. Notably, focused ion beam (FIB) machining has become a popular tool in the post-processing of fiber probes since it can produce and modify structures at the nanometer scale [39-41]. The metal coating of the fiber probe often extends beyond the boundary of the fiber as seen in Figure 2.11. This effectively limits the probe-sample distance and the intensity that can reach the sample resulting in lower quality images. Focused ion beam milling has been employed to cleave off the tip of fiber optic NSOM probes [39, 41] revealing an extremely flat surface and unstructured aperture region. Focused ion beam processes have also been used to fabricate the aperture itself. By milling into the tip of a coated metal fiber, FIB

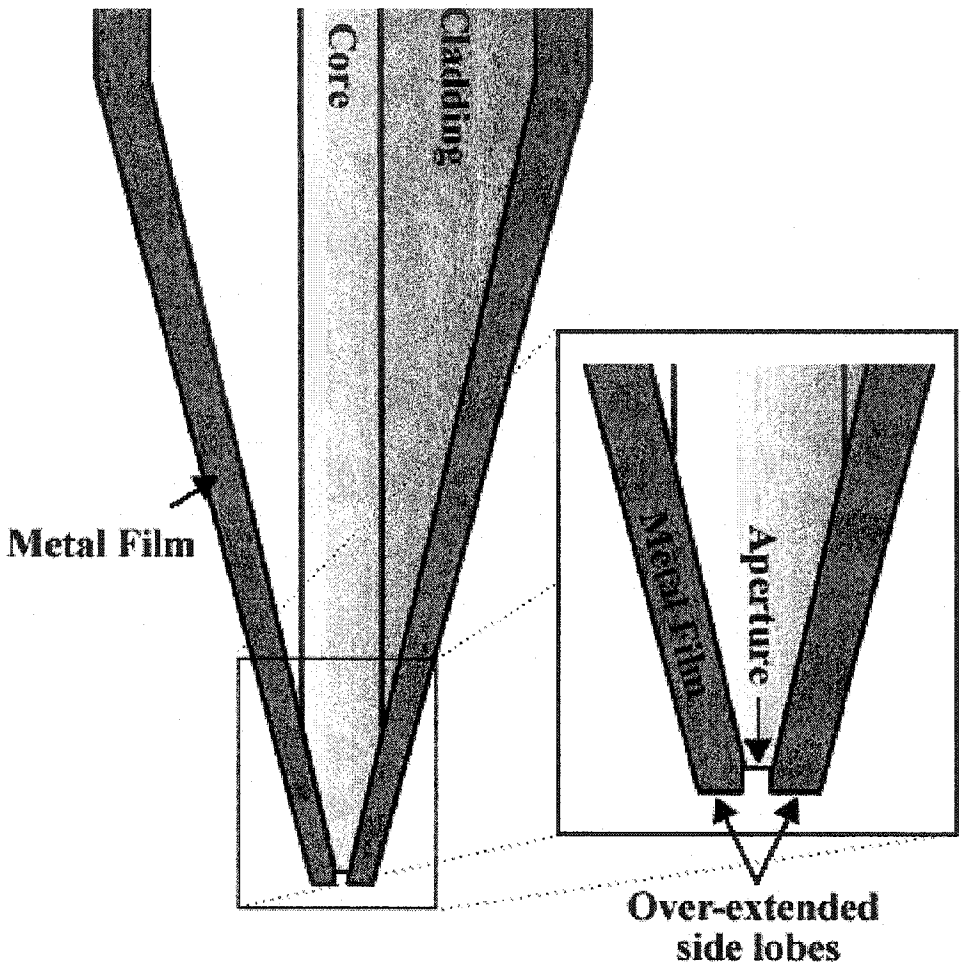


Figure 2.11: Due to the metal deposition process, the metal film often extends beyond the aperture of the fiber, thereby reducing the actual amount of near-field light reaching the sample.

milling enables the creation of subwavelength apertures with dimensions less than 50nm [40]. Despite the inherent gains in image quality, focused ion beam processes have several drawbacks. Namely, the cost and availability of the necessary equipment is often restrictive. Furthermore, unlike other microfabrication processes, FIB milling can not be performed on simultaneous samples and, therefore, is not conducive to batch fabrication. Finally, when using FIB cleaved probes, a compromise must be made in terms of the quality of the simultaneous force images collected. Shear force topographic images are largely a function of the metal grains that exist at the aperture [42], and as the grains are removed during FIB machining, topographic image quality suffers.

2.6 Cantilevered Near-Field Probes

Although fiber based systems seem to be the industry standard, an increasing trend towards micromechanical cantilevered near-field optical systems has been realized [43-53] since fiber based probes suffer from several drawbacks. Fiber optic probes are ill suited for imaging soft biological tissue due to their rigidity and fragility. Moreover, the fiber-tip fabrication process is cumbersome, not conducive to batch fabrication and suffers from low reproducibility.

Atomic force microscopy (AFM), however, has been used for years to image living cells and batch fabrication is easily accomplished using conventional silicon micromachining technologies. The AFM measures angstrom scale sample topography by detecting minute changes in the tip-sample interaction forces at the pN scale as the tip is rastered across the sample [25]. Although AFMs have largely been used for topographical characterization, high-resolution lithography [54], and mass data storage [55] have also been demonstrated using an AFM. Conversely, the NSOM utilizes locally

confined light passing through a subwavelength aperture that is rastered across the sample to create a near-field optical image. Like its AFM counterpart, the NSOM has been used for high-density data storage [56], near-field optical lithography [57], and thin film characterization. Combining an AFM with a cantilevered NSOM, therefore, seems only logical, and various integrated systems have been reported.

The key to coupling these two technologies is the fabrication of a combined AFM/NSOM probe. Two simple and common techniques are employed in the creation of AFM probes. The first method entails chemically etching a silicon substrate with a hard masking layer such as silicon dioxide or silicon nitride. By employing these masking layers, undercutting results in the formation of a very sharp tip [58]. This etching process is either isotropic or anisotropic depending on the etchant used. The former case yields a concave conical tip as depicted in Figure 2.12(a), whereas the latter case results in a pyramidal shaped tip as seen in Figure 2.12(b). Similar to the fiber probes, the resulting transparent tips must be subsequently coated with a metal layer in order to achieve light confinement at the nanometer scale. Several methods have been proposed for aperture formation including etching of the metal coating [49], focused ion beam milling [41], and even electron beam lithography [50]. Rather than etching, the latter fabrication method

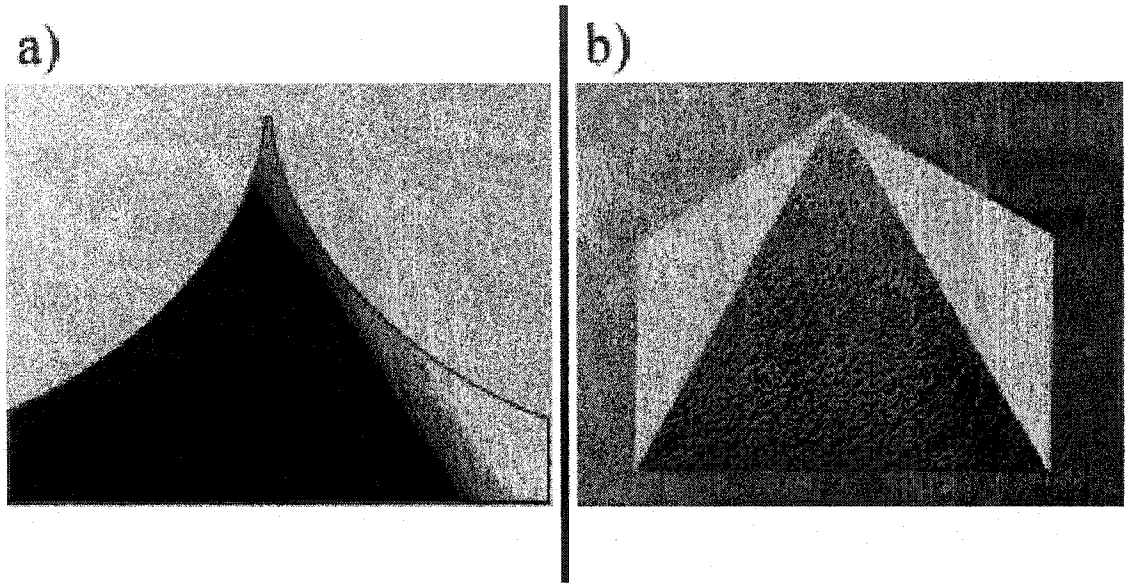


Figure 2.12: Etched silicon NSOM tips are formed either by isotropic (a) or anisotropic (b) etching. In the former case a sharp cone structure is fabricated, whereas in the latter case a pyramidal structure is evident.

involves a deposition process. A pyramidal probe is formed similar to that depicted in 2.10(b) by depositing metal such as chrome, silver or aluminum, into a silicon template [52]. The aperture, in this case, is a direct result of the deposition process.

The use of conventional microfabrication techniques in the development of cantilever based near-field optical probes also facilitates the integration of other "on-board" optical or electronic devices into the cantilever. Researchers have demonstrated integrated far-field detectors such as photodiodes [59], as well as near-field light sources such as light emitting and laser diodes [59, 60]. Micromechanical actuators and piezo-electric bimorphs have also been introduced as on-cantilever tip-sample distance regulation mechanisms [61-63].

In contrast to the fabrication process for fiber based probes, cantilevered probe designs are much more conducive to batch fabrication. This leads to decreased costs and higher rates of production. Moreover, the optical throughput of the conventional fiber probe is very low due to absorption effects of the metallic coating as well as the cutoff effect within the waveguide. Cantilevered probes do not suffer from such limitations. Finally, the shear-force regulation mechanism commonly employed in far-field systems is complicated and tends to limit optical resolution. Due to the dithering motion of the probe caused by the piezo driver, a fiber based system's lateral resolution is limited by the amplitude of the vibration. A general comparison of fiber and cantilevered probes is found in Table 2.1.

Table 2.1: Comparison of fiber based and cantilever based NSOM probes

Comparison Parameter	*Optical Fiber Probe*	*Silicon Micromachined Probe*
Fabrication	No mass production (> cost)	Mass production (< cost)
Reproducibility	Poor	Good
Optical Throughput	Low (10^{-6}-10^{-3} for 100 nm aperture)	Higher (10^{-3}-10^{-2} for 100 nm aperture)
Polarization Control	Complicated	Feasible
Mechanical Properties	Poor	Excellent
Aperture Size	20 – 200 nm	20 – 200 nm
Tip-Sample Regulation	Shear Force	Atomic Force
Array Capability	Complicated	Feasible

2.7 Silicon Micromachining Technology

Silicon (Si) micromachining technologies are currently the most appropriate means to construct sensors, actuators and micro-devices for applications as far reaching as medicine, biology, telecommunications and transportation. This set of technologies is, by consequence, highly conducive to the fabrication of SPM probes. Among the main advantages inherent to silicon micromachining are its high reproducibility, mass production capability, ease of integration with other devices, and its precise manipulation of properties at the micron and sub-micron scale. The following section, therefore, discusses the fundamental technologies necessary in the fabrication of the silicon AFM/NSOM probes described above as well as the probes described further on in this thesis.

2.7.1 Thermal Oxidation

Thermal oxidation is a key, multipurpose component in the life-cycle of any silicon micro-device. Thermally grown silicon dioxide (SiO_2) can serve as an electronic insulator, as an etch mask or etch stop, as a diffusion mask for Boron or Phosphorous doping, as a sacrificial medium, or as a lens or conduit material for optical applications. SiO_2 is typically thermally grown on a silicon surface by either dry or wet oxidation. In a stream of pure dry oxygen (dry oxidation), silicon atoms at the wafer surface are oxidized according to

$$Si + O_2 = SiO_2. \tag{2.18}$$

Wet oxidation, however, involves a more complex reaction between silicon and evaporated water at elevated temperatures (600° – 1250° C). High temperatures are

necessary to aid diffusion of the oxidizing species through the existing SiO_2 layer, thereby increasing the rate of the reaction

$$Si + 2H_2O \rightarrow SiO_2 + 2H_2 . \quad (2.19)$$

When a thin oxide layer is grown on a silicon substrate, the silicon will be consumed at 0.46 times the growth rate of the oxide layer according to the Deal-Grove model [64]. This theory suggests that the oxidation process is accomplished in three stages. First, a species of oxidant is transported from the bulk gas phase to the surface of the substrate. Second, the reactive species is diffusively transported across the oxide film to the silicon layer, where it finally reacts with the fresh silicon forming a new oxide layer according to equations 2.18 and 2.19. The entire procedure occurs within a thermal oxidation furnace.

This Deal-Grove theory adequately describes the growth of SiO_2 on planar substrates; however, an anomaly arises when a non-planar structure is considered, as demonstrated in Figure 2.13. Specifically, the thermally grown SiO2 layer is thinner at convex and concave corners when low temperatures are employed (< 1050°C). When higher temperatures are utilized, the thinning effect is less noticeable due to the increased viscous flow of the oxide layer. The reduction of the oxidation rate at the concave corners is explained by the geometry of the structure. A larger oxide interface exists at a concave corner than on an oxide layer bounded by a flat surface. The concentration of reactive species reaching the silicon, therefore, is reduced, resulting in a reduction of the oxidation rate. The reduction in oxidation at the convex corner, however, is largely attributed to increased compressive stress in these regions. This increased level of stress, it is theorized, has the effect of reducing the diffusion of the reactive species to the silicon

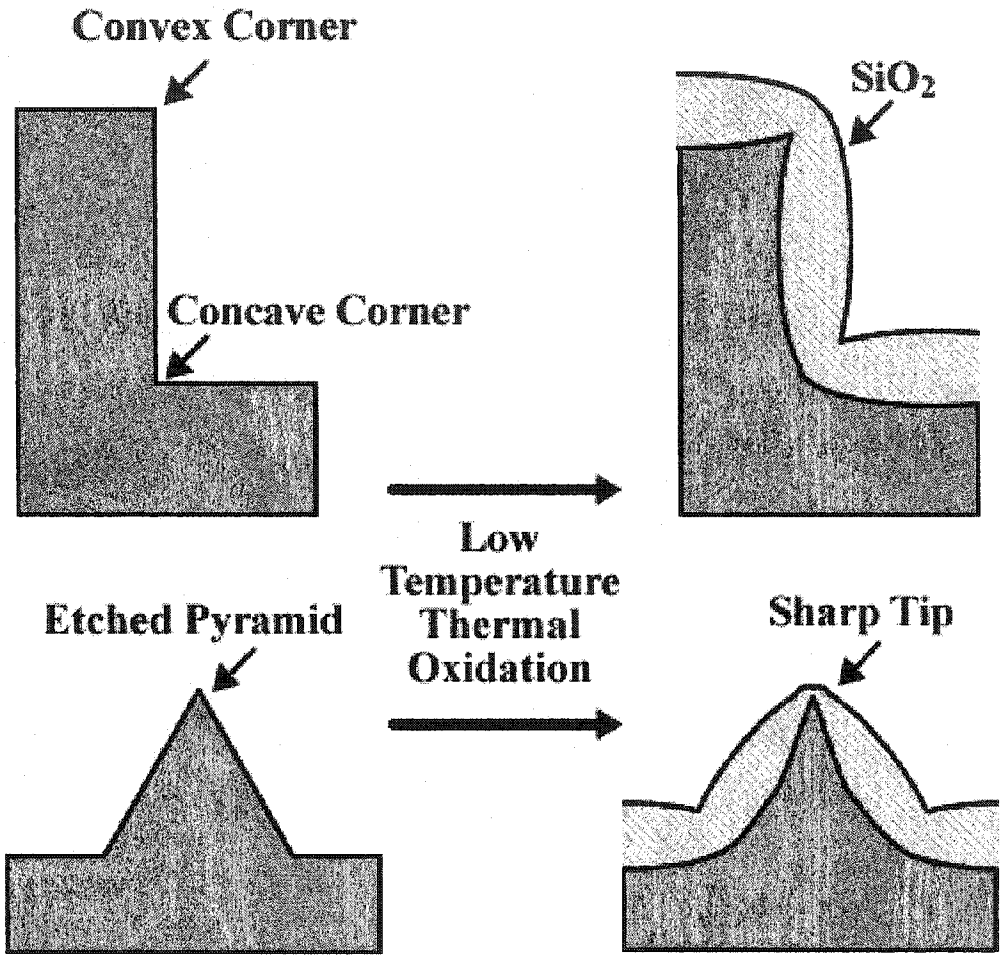


Figure 2.13: Low-temperature thermally grown silicon dioxide at concave and convex corners exhibit an anomalous thinning. This thinning can be employed to apertures, and sharp tip structures for use in atomic force and near-field optical microscopy.

layer. The direct result of this thinning process has a single benefit when discussing the fabrication of near-field probes. It enables the creation of sharp tips as seen in Figure 4.2, and extremely small apertures as will be subsequently discussed.

2.7.2 Photolithography

Photolithography remains the fundamental component to silicon microfabrication, as it allows for the transfer of precise micron and sub-micron patterns from a predefined mask onto a thin photoresist layer covering various types of substrates. It is this layer of photoresist that, after pattern transfer has occurred, acts as a masking material for the subsequent etching or deposition step on the underlying substrate.

The process of conventional lithography is best described by an example of pattern transfer from a chromium mask onto a silicon substrate as illustrated in Figure 2.14. The desired pattern is first written on a chromium mask blank leaving either transparent or opaque features on the mask. Using a mask aligner, the mask is brought into direct contact with a photoresist covered substrate where it is subsequently exposed to ultraviolet (UV) radiation. The transparent portions of the mask permit passage of the UV light thereby inducing a photo-chemical reaction in the photoresist layer. This reaction has the direct effect of altering the solubility of the resist in a specific solvent. If a positive photoresist is employed, the UV light will degrade the polymer resist leading to increased solubility. However, if a negative resist is utilized, the UV light will facilitate polymer cross-linking thereby decreasing the solubility of the resist. After developing the resist in the appropriate solvent, the wafer can be further processed using the patterned photoresist layer as a mask for etching or deposition.

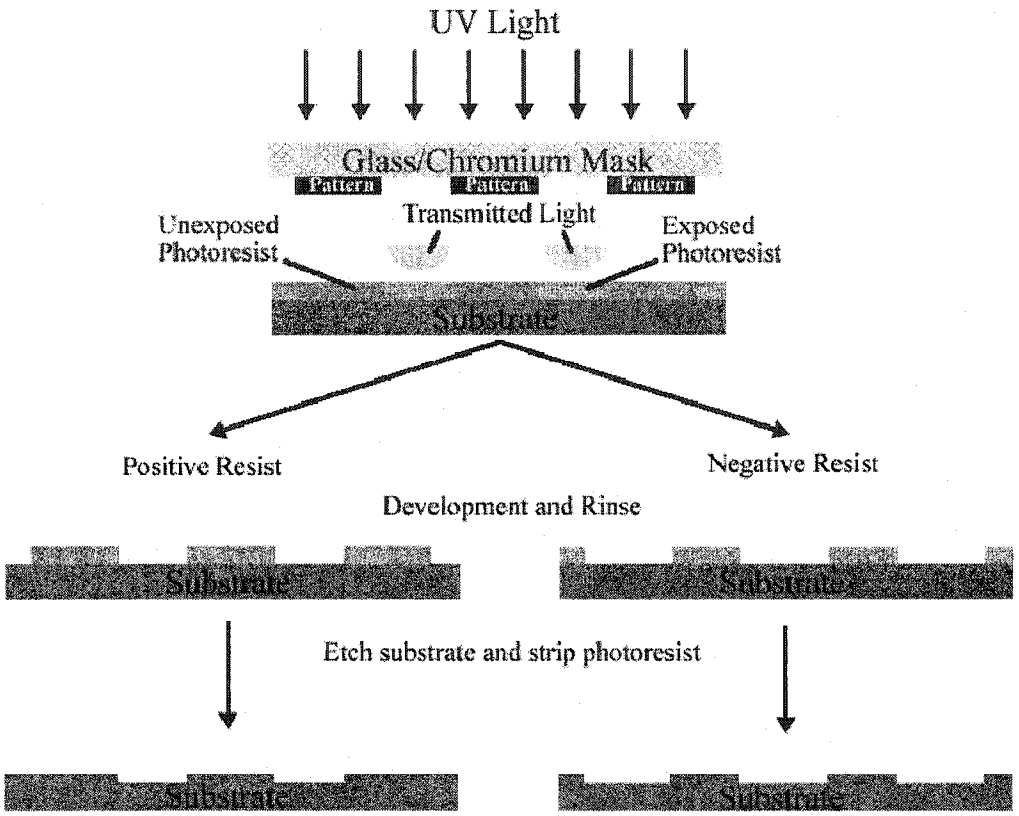


Figure 2.14: Ultraviolet photolithography process in which a positive/negative photoresist is employed as a substrate masking layer. The photoresist is either degraded or cross-linked after exposure to UV light, and the pattern is subsequently revealed after developing.

2.7.3 Silicon Oxide Etching

Often it is necessary to use a secondary masking layer, as photoresist tends to dissolve in several of the common etchants employed in silicon micromachining; silicon dioxide is the most common "hard" masking layer used in place of photoresist. Thus, it is necessary to understand the etching process for SiO_2. Wet etching of SiO_2 is commonly executed using a solution of hydrofluoric acid (HF). The etch rate and etch characteristics can be widely tailored by employing different additives such as nitric acid (HNO_3), orthophosphoric acid (H_3PO_4), or ammonium fluoride (NH_4F) [65] as well as by varying the solution's temperature or pH.

SiO_2 etching in this work was accomplished via a solution of buffered HF or buffered oxide etchant (BOE) consisting of 6 parts 40% ammonium fluoride to 1 part 49% hydrofluoric acid. BOE wet etching is desirable because of its moderate etch rate (~35 nm / minute) and the marked lack of undercutting. The etching mechanism has been elucidated by several researchers [66] revealing that the etching solution consists of several distinct species (H^+, F^-, HF_2^-, NH_4^+, HF) formed via the following relations

$$HF \leftrightarrow H^+ + F^- \tag{2.20}$$

$$NH_4F \leftrightarrow NH_4^+ + F^- \tag{2.21}$$

$$HF + F^- \leftrightarrow HF_2^-. \tag{2.22}$$

Overall, the chemical reaction responsible for the etching of silicon dioxide was determined to be

$$SiO_2 + 6HF \rightarrow 2\,H^+ + SiF_6^{-2} + 2H_2O. \tag{2.23}$$

Using this etching technique, pattern transfer from the soft photoresist mask to a hard silicon dioxide mask is easily accomplished.

2.7.4 Anisotropic Wet Chemical Etching

For the work described herein, (100) oriented silicon wafers are utilized. As it is well established that the etching of silicon in several types of liquid etchants is highly anisotropic, etching of (100) Si can be effectively employed to create a wide variety of structures including trenches, pyramidal etch pits, thin membranes, vias, and mesa structures. The anisotropy in wet Si etching is due to the slower etch rate of the (111) plane compared to that of the (110) and (100) planes in aqueous solutions of either potassium hydroxide (KOH), tetramethyl ammonium hydroxide (TMAH) or ethylene diamine pyrochatechol (EDP).

The wet anisotropic silicon etching performed during the fabrication of the near-field probes was accomplished solely using TMAH [$(CH_3)_4NOH$], as TMAH presents several advantages over its counterparts. First TMAH is relatively non-toxic and is compatible with existing CMOS processing technologies. Second, TMAH facilitates the employment of a SiO_2 hard mask as the selectivity between Si/SiO_2 is greater than 2000:1. Third, the smoothness of the Si surface can be tailored via the addition of isopropyl alcohol (IPA). Finally, the etch rate of TMAH is relatively constant at 0.5 μm/min at 110°C. Etch profiles of a typical TMAH etch can be seen in Figure 2.15 where the formation of pyramidal pits and V-grooves, having a base angle of 54.74°, is clearly evident.

The chemical reaction underlying the wet chemical etching of silicon [65] has been previously been found to be

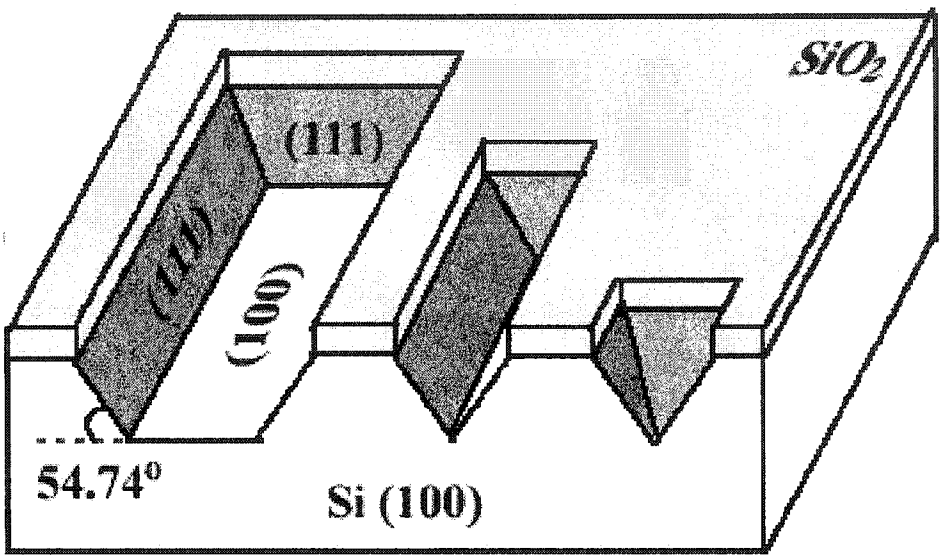


Figure 2.15: Anisotropic etching of silicon, as depicted here, is able to create trenches, V-grooves, and pyramidal etch pits of varying sizes. A hard mask layer of silicon dioxide or silicon nitride is necessary.

$$Si + 2OH^- \rightarrow Si(OH)_2^+ + 4e^- \quad (2.24)$$

$$4H_2O + 4e^- \rightarrow 4OH^- + 2H_2 \quad (2.25)$$

Here Si atoms at the wafer surface react with hydroxyl ions to generate electrons which, in turn, react with water to form hydrogen gas, and more hydroxyl ions. As this sequence is repeated, layers of Si are gradually removed. A highly doped layer containing Boron atoms ($P^{++} > 10^{20}$) can be effectively used as an etch stop for this process as the electrons generated by the reaction in equation 2.25 will recombine with holes in the P^{++} layer thereby preventing the occurrence of any further etching.

2.7.5 Deep Reactive Ion Etching

A dry anisotropic etching technique was also employed in this work. Instead of the V grooves or pyramidal etch pits mentioned above, inductively coupled plasma reactive ion etching (ICP RIE), also known as deep reactive ion etching (DRIE) is a highly anisotropic etch revealing nearly vertical sidewalls as seen in Figure 2.16. The etching process, also portrayed in Figure 2.16, is a time-multiplexed process consisting of alternating etching and passivation stages [67]. The etching and passivation gases are SF_6 and C_4F_8 respectively. During the passivation phase, the passivation gas is dissociated within the plasma to form both ionic and radical species. The passivation layer (nCF_2), formed by these species, is deposited on the surface of both the masking layer and silicon. The subsequent etching phase consists of an immediate replacement of the passivation gas with the etching gas, SF_6. Again this gas is dissociated in the plasma and the directional ion bombardment from the fluorine radicals promotes the preferential removal of the passivation film as well as the silicon from all horizontal surfaces. Undercutting during the silicon etch leads to scalloping or the sidewalls

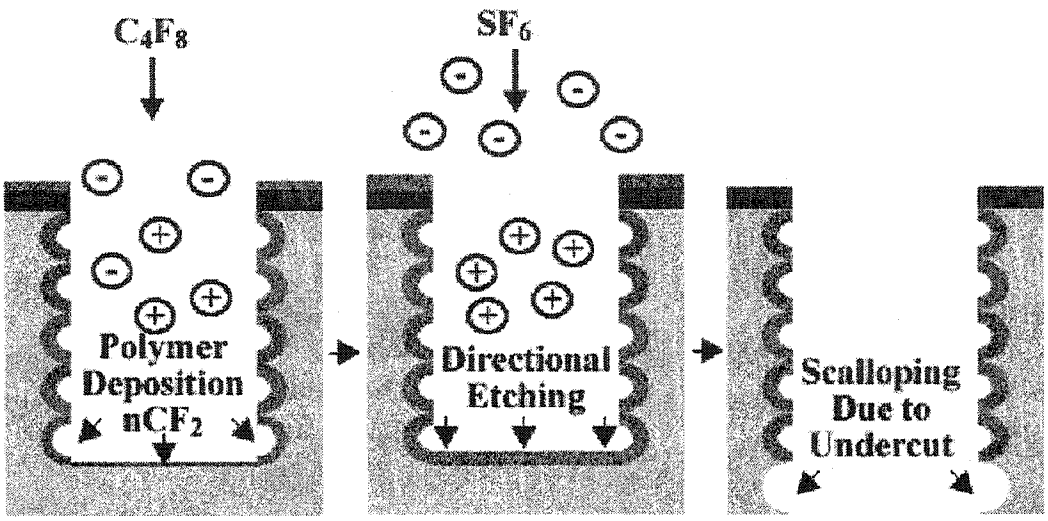


Figure 2.16: The time-multiplexed etching process known as ICP RIE. A fluorocarbon polymer is first deposited onto the substrate coating both masking material and exposed silicon. A directional etch of positive ions formed from SF_6 is then used to first etch the polymer and then the underlying silicon on any horizontal surface. Scalloping occurs due to horizontal undercutting during the etching process.

although this effect can be minimized by reducing the silicon etch rate. Known as the Bosch process, this separation of passivation and etching cycles can be precisely regulated to achieve a desired etch rate, uniformity, aspect ratio, sidewall angle, and mask selectivity. Etch rates in excess of 8 μm/minute, photoresist soft mask selectivity approaching 100:1, and aspect ratios greater than 20:1 have all been demonstrated [68].

2.7.6 Metallization via Magnetron RF Sputtering

Magnetron radio-frequency (RF) sputtering offers several advantages over its counterpart, electron beam evaporation. A wider selection of sputter targets exists, and the deposition process affords better step coverage and adhesion to the underlying substrate. During the sputtering process, the deposition chamber is typically evacuated to a base pressure below 10^{-6} torr at which time Argon gas (Ar) is injected into the chamber until a constant pressure of 10^{-3} torr is reached. A plasma is ignited and positively charged argon ions formed within the plasma subsequently bombard the metal target that is biased to a large negative potential. Enough momentum and energy are transferred from the incident argon ions to the metal atoms that the metal atoms are knocked off the target to be deposited on the substrate. Accurate control over the thickness of the metal film is achieved by altering the bias voltage, the sputtering time, the power of the RF plasma or the pressure within the vacuum chamber.

2.7.7 Liftoff

The lift-off process is very simple and straightforward. It is widely employed in the formation of metallic patterns on a substrate. After lithography, metal is deposited over the exposed and unexposed areas. A solvent, such as acetone, is then used to remove the

remaining photoresist from underneath the deposited metal, essentially using the photoresist as a sacrificial layer. Once complete, only the metal pattern remains on the substrate.

2.8 Review of Nano-Optics

2.8.1 Optical Transmission of Periodic Metallic Nano-Structures

Recent experiments in near-field photonics have produced evidence in direct contradiction with classical diffraction theory [69]. Most notably, significant enhancements in transmission have been demonstrated for both metallic periodic aperture arrays [70], and single apertures surrounded by periodically corrugated metallic nano-structures [71]. According to subwavelength diffraction theory, optical transmission through a subwavelength aperture ($\lambda >> a$) in an infinitely thin, perfectly conductive metallic film is inversely proportional to the fourth power of the aperture radius [5]. However, surface plasmons excited at a metal-dielectric interface have a significant effect on the transmission properties of any aperture array. Notably, by coupling to the opposite surface via the nano-apertures, the SPs are eventually re-emitted as far-field light at several discrete wavelengths [72-74]. This re-radiation process is prompted by the subsequent diffraction of the SPs off of the corner at the far edge of the aperture. This results in a normalized optical transmission in excess of 80% and 7 times greater than that predicted by conventional diffraction theory [75]. Geometrical factors such as nano-structure periodicity, angle of incidence, and film thickness have a pronounced affect on the level of SP coupling and thus on the optical transmission.

It has also been demonstrated that these sub-wavelength structures have a significant effect on the temporal characteristics of an ultrashort pulse. Homogeneous broadening, as well as significant temporal delay, have been measured for pulses >100 fs [76]. Notably, a 100 fs pulse transmitted through a periodic array of 300 nm apertures suffered a delay of 11 fs. Numerical modeling of the optical transmission has shown that both the enhanced transmission and the pulse shaping characteristics of the periodic nano-structures are caused by the collective effect of EM coupling between the nano-structures and SP waves. Calculated spectra, published elsewhere [77, 78], are in a good agreement with the experimental data.

2.8.2 Modeling of Near-Field Optical Microscope Probes

Due to the evanescent nature of near-field light, near-field optical microscopy, like many forms of scanning probe microscopy (SPM), continues to suffer from very low signals. While current near-field far-field techniques yield sub-100 nm resolution [23], their very low intensity throughput (10^{-6} to 10^{-5}) [30] is inherent in the design. Furthermore, solid immersion lens techniques, employing a micron-scale hemispherical high-index lens in contact with sample to be imaged, as seen in Figure 2.17, have achieved a numerical aperture approaching 2, yielding 139 nm resolution; however, solid immersion lenses have yet to attain sub-100 nm spatial resolution accessible via near-field far-field optical probes.

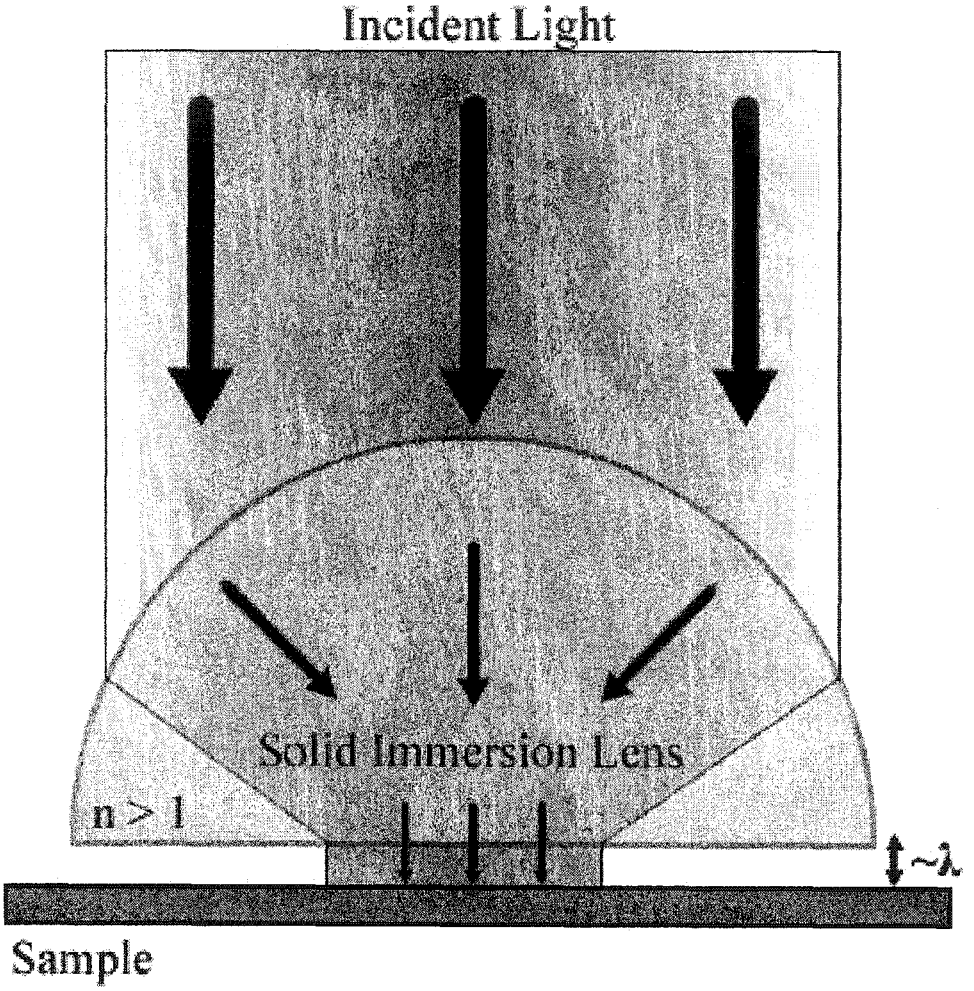


Figure 2.17: Solid immersion lens system. The lens, in close proximity to the sample, is used to focus light to a small spot.

Owing to the complexity of exactly solving Maxwell's equations in the near-field, several numerical methods have been used to model the specific geometry of the NSOM probe. Simplistic models such as the Bethe-Bouwkamp technique have provided a detailed two and three dimensional analysis of the radiation patterns within and surrounding far-field NSOM probes. From these studies, the transmission for a typical probe geometry ranges from approximately 10^{-3} to 10^{-12} as the diameter of fiber core is reduced from 200 to 20 nm [79]. Unfortunately, these simplistic models do not contain a complete picture of the interaction of EM radiation with the probe in the near-field, and more rigorous numerical models, such as the finite-difference time-domain (FDTD) algorithm have since been employed in the study of NSOM probes.

2.8.3 Applications of Near-Field Optical Microscopy

The ultimate goal of near-field scanning optical microscopy seems to be the in situ investigation of living cells in the nanometer regime. Although yet unattainable, the pursuit of this goal has led to the development of several novel near-field imaging techniques.

Near-field fluorescence spectroscopy represents the simplest and most informative NSOM method. Providing high optical contrast, fluorescence imaging has largely been used in single molecule and biological studies [80-84]. Spectral investigation of single molecules eliminates the averaging of material properties encountered when observing bulk samples. Thus, probing single molecules offers new possibilities in determining molecular dynamics that would otherwise be unobservable [85]. Single molecule NSOM studies typically involve the measurement of fluorescence lifetimes of single fluorophores, or dye-labeled molecules. These studies have been used to elucidate

a reduction effect in the fluorescence signal due to tip-sample interactions [86]. When the molecule is beneath the metal coating of the tip, non-radiative energy transfer between molecule and metal occurs leading to a dramatic decrease in fluorescence. Other practical single molecule studies include the diffusion analysis of rhodamine-6G dye molecules in thin organic films [87], and the study of human chromosomes [88, 89].

Contrast mechanisms such as transmission, emission, absorption, fluorescence and polarization also provide a rich set of tools for thin film optical analysis in addition to biological studies [84, 88, 90]. Electroluminescent organic thin films present considerable potential for their possible role in light emitting diodes, and photodetector applications. Mainstream adoption of luminescent thin films, however, is minimal due to their current low luminescence efficiencies. NSOM provides a non-invasive method for examining thin film photophysics [91-94], which may potentially lead to increased efficiencies. Thin film near-field optical analysis is also important due to the increasing demands for nano-optoelectronic devices and high-density data storage media [95-97].

Polarization contrast NSOM presents a unique method for studying nanometer scale optical anisotropy and magneto-optical effects. Conveniently, the polarization of light emitted at an NSOM aperture, as measured in the far-field, can be adjusted to any desired state [47]. Notably, polarization NSOM has been utilized in the study of the electric field response of liquid crystals [98]. Under an applied voltage, light transmitted by the liquid crystal undergoes a polarization change that can be subsequently detected by a near-field optical microscope. Polarization NSOM has also been well received as a means for examining localized Faraday rotation in iron-garnet films [99], leading to the quantitative evaluation of magneto-optical effects on the nanometer scale.

2.9 References

1. Abbe, E., *Beiträge zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung.* Arch. Mikrosk. Anatomie, 1873. **9**: p. 413.

2. Dorn, R., Quabis, S., and Leuchs, G., *Sharper Focus for a Radially Polarized Light Beam.* Physical Review Letters, 2003. **91**: p. 233901.

3. Massey, G.A., *Microscopy and pattern generation with scanned evanescent waves.* Applied Optics, 1984. **23**(5): p. 658.

4. Synge, E.H., *A suggested method for extending microscopic resolution into the ultra-microscopic region.* Philosophy Magazine, 1928. **6**: p. 356.

5. Bethe, H.A., *Theory of Diffraction by Small Holes.* Physial Review, 1944. **66**(7-8): p. 163–182.

6. Born, M.a.W., E., *Principles of Optics: Electromagnetic Theory of Propagation, Interference, and Diffraction of Light.* 1999, Cambridge: Cambridge University Press.

7. Bouwkamp, C.J., *On the diffraction of electromagnetic waves by small circular discs and holes.* Phillips Research Reports, 1950. **5**: p. 321.

8. Leviatan, Y., *Study of near-zone fields of a small aperture.* Journal of Applied Physics, 1986. **60**(5): p. 1577.

9. Raether, H., *Surface Plasmons on Smooth and Rough Surfaces and on Gratings.* Springer Tracts in Modern Physics. Vol. 111. 1988, Berlin: Springer Verlag.

10. Kretschmann, E.a.R., H., *Radiative decay of non-radiative surface plasmons excited by light.* Z. Naturforsch, 1968. **A 23**: p. 2135.

11. Boardman, A.D., *Electromagnetic Surface Modes*. 1982, New York: Wiley.

12. Xiao, M., Zayats, A. V. and Siqueiros, J., *Scattering of surface-plasmon polaritons by dipoles near a surface: Optical near-field localization.* Physical Review B, 1997. **55**: p. 1824.

13. Shchegrov, A.V., Nvikov, I. V. and Maradudin, A. A., *Scattering of Surface Plasmon Polaritons by a Circularly Symmetric Surface Defect.* Physical Review Letters, 1997. **78**: p. 4269.

14. Moskovits, M., *Surface-enhanced spectroscopy.* Rev. Mod. Phys., 1985. **57**: p. 783.

15. Maier, S.A., Kik, P. G., Atwater, H. A., Meltzer, S., Harel, E., Koel, B. E. and Requicha, A. G., *Local detection of electromagnetic energy transport below the*

diffraction limit in metal nanoparticle plasmon waveguides. Nature Materials, 2003. **2**: p. 229.

16. Maier, S.A., Kik, P. G., and Atwater, H. A., *Optical Pulse Propagation in Metal Nanoparticle Chain Waveguides.* Physical Review B, 2003. **67**: p. 205402.

17. Ash, E.A. and G. Nicholls, *Super-resolution aperture scanning microscope.* Nature, 1972. **237**(5357): p. 510.

18. Dürig, U., D.W. Pohl, and F. Rohner, *Near-field optical-scanning microscopy.* Journal of Applied Physics, 1986. **59**(10): p. 3318.

19. Pohl, D.W., W. Denk, and M. Lanz, *Optical stethoscopy: Image recording with resolution /20.* Applied Physics Letters, 1984. **44**(7): p. 651.

20. Betzig, E.I., J. and Lewis, A., *Collection mode near-field scanning optical microscopy,.* Appl. Phys. Lett., 1987. **51**: p. 2088.

21. Valle Pedro, J., R. Carminati, and J. Greffet Jean, *Contrast mechanisms in illumination-mode SNOM.* Ultramicroscopy, 1998. **71**(1-4): p. 39.

22. Cline, J.A., H. Barshatzky, and M. Isaacson, *Scanned-tip reflection-mode near-field scanning optical microscopy.* Ultramicroscopy, 1991. **38**: p. 3.

23. Betzig, E. and J.K. Trautman, *Near-field Optics: Microscopy, Spectroscopy, and Surface Modification Beyond the Diffraction Limit.* Science, 1992. **257**(5067): p. 189.

24. Karrai, K. and R. Grober, *Piezo-electric tuning fork tip-sample distance control for near field optical microscopes.* Ultramicroscopy, 1995. **61**(1): p. 197.

25. Binnig, G., C.F. Quate, and C. Gerber, *Atomic Force Microscope.* Physical Review Letters, 1986. **56**(9): p. 930.

26. Garcia Parajo, M., T. Tate, and Y. Chen, *Gold-coated parabolic tapers for scanning near-field optical microscopy: fabrication and optimisation.* Ultramicroscopy, 1995. **61**(1-4): p. 155.

27. Nakamura, H., et al., *FDTD simulation of tapered structure of near-field fiber probe.* Computer Physics Communications, 2001. **142**(1-3): p. 464.

28. Paesler Michael, A. and J. Moyer Patrick, *Near-field Optics: Theory, Instrumentation and Applications.* 1996, New York: John Wiley & Sons Inc. 355.

29. Betzig, E., et al., *Breaking the diffraction barrier: Optical microscopy on a nanometric scale.* Science, 1991. **251**: p. 1498.

30. Valaskovic, G.A., Holton, M. and Morrison, G. H., *Parameter control, characterization, and optimization in the fabrication of optical fiber near-field probes.* Applied Optics, 1995. **34**(7): p. 1215.

31. Yakobson, B.I., et al., *Thermal/optical effects in NSOM probes.* Ultramicroscopy, 1995. **61**(1-4): p. 179.

32. Novotny, L., and Hafner, C., *Light propagation and confinement in a cylindrical waveguide with a complex metallic dielectric function.* Physical Review E, 1994. **50**: p. 4094.

33. Mononobe, S. and M. Ohtsu, *Fabrication of a pencil-shaped fiber probe for near-field optics by selective chemical etching.* Journal of Lightwave Technology, 1996. **14**(10): p. 2231.

34. Mononobe, S. and M. Ohtsu, *Model based on geometrical construction in designing a pencil-shaped fiber probe for near-field optics.* Journal of Lightwave Technology, 1997. **15**(6): p. 1051.

35. Mononobe, S., et al., *Fabrication of a triple tapered probe for near-field optical spectroscopy in UV region based on selective etching of a multistep index fiber.* Optics Communications, 1998. **146**(1-6): p. 45.

36. Lambelet, P., et al., *Chemically Etched Fiber Tips for Near-Field Optical Microscopy: A Process for Smoother Tips.* Applied Optics, 1998. **37**(31): p. 7289.

37. Lazarev, A., Fang, N., Lio, Q., and Zhang, X., *Formation of fine near-field scanning optical microscopy tips.* Review of Scientific Instruments, 2004. **74**: p. 3679.

38. Weston Kenneth, D., A. DeAro Jessie, and K. Buratto Steven, *Near-field Scanning Optical Microscopy in Reflection - A Study of Far-field Collection Geometry Effects.* Review of Scientific Instruments, 1996. **67**(8): p. 2924.

39. Dziomba, T., et al., *Ion beam-treated silicon probes operated in transmission and cross-polarized reflection mode near-infrared scanning near-field optical microscopy (NIR-SNOM).* Surface and Interface Analysis, 1999. **27**(5): p. 486.

40. Muranishi, M., et al., *Control of aperture size of optical probes for scanning near-field optical microscopy using focused ion beam technology.* Japanese Journal of Applied Physics, Part 2: Letters, 1997. **36**(7B): p. L942.

41. Veerman, J., et al., *High definition aperture probes for near-field optical microscopy fabricated by focused ion beam milling.* Journal of Applied Physics, 1998. **72**(24): p. 3115.

42. Schmidt, J.U., H. Bergander, and L.M. Eng, *Experimental and theoretical analysis of shear-force interaction in the non-contact regime with 100 pN force resolution.* Applied Surface Science, 2000. **157**(4): p. 295.

43. Akamine, S., H. Kuwano, and H. Yamada, *Scanning near-field optical microscope using an atomic force microscope cantilever with integrated photodiode.* Applied Physics Letters, 1996. **68**(5): p. 579.

44. Bauer, P., B. Hecth, and C. Rossel, *Piezoresistive cantilevers as optical sensors for scanning near-field microscopy.* Ultramicroscopy, 1995. **61**(1-4): p. 127.

45. Danzebrink Hans, U., O. Ohlsson, and G. Wilkening, *Fabrication and characterization of optoelectronic near-field probes based on an SFM cantilever design.* Ultramicroscopy, 1995. **61**(1-4): p. 131.

46. Drews, D., et al., *Nanostructured probes for scanning near-field optical microscopy.* Nanotechnology, 1999. **10**(1): p. 61.

47. Eckert, R., Freyland, J. M., Gersen, H., Heinzelmann, H., Schu È Rmann, G., Noell, W., Staufer, U. and de Rooij, N. F., *Near-field optical microscopy based on microfabricated probes.* Journal of Microscopy, 2001. **202**(1): p. 7.

48. Goettlich, H. and W.M. Heckl, *Novel probe for near field optical microscopy based on luminescent silicon.* Ultramicroscopy, 1995. **61**(1-4): p. 145.

49. Jung, M.Y., S.S. Choi, and I.W. Lyo, *Micromachined Si sub 3N sub 4-tip on cantilever for parallel SFM and NSOM applications.* Microelectronic Engineering, 1999. **46**(1): p. 427.

50. Mihalcea, C., et al., *Multipurpose sensor tips for scanning near-field microscopy.* Applied Physics Letters, 1996. **68**(25): p. 3531.

51. Minh, P.N., T. Ono, and M. Esashi, *Microfabrication of miniature aperture at the apex of SiO sub 2 tip on silicon cantilever for near-field scanning optical microscopy.* Sensors and Actuators, A: Physical, 2000. **80**(2): p. 163.

52. Oesterschulze, E., et al., *Cantilever probes for SNOM applications with single and double aperture tips.* Ultramicroscopy, 1998. **71**(1-4): p. 85.

53. Schurmann, G., et al., *Microfabrication of a combined AFM-SNOM sensor.* Ultramicroscopy, 2000. **82**(1): p. 33.

54. Sugimura, H.a.N., N., *AFM lithography in constant current mode.* Nanotechnology, 1997. **8**: p. A15.

55. Hosaka, S., et al., *SPM-based data storage for ultrahigh density recording.* Nanotechnology, 1997. **8**(3A): p. A58.

56. Hosaka, S., et al., *Nanometer-sized phase-change recording using a scanning near-field optical microscope with a laser diode.* Japanese Journal of Applied Physics, Part 1: Regular Papers and Short Notes and Review Papers, 1996. **35**(1B): p. 443.

57. Wei, P.K., et al., *Surface modification and patterning of conjugated polymers with near-field optical microscopy.* Advanced Materials, 1996. **8**(7): p. 573.

58. Oesterschulze, E., *Novel probes for scanning probe microscopy.* Applied Physics A: Materials Science and Processing, 1998. **66**: p. S3.

59. Sasaki, M., K. Tanaka, and K. Hane, *Cantilever probe integrated with light-emitting diode, waveguide, aperture, and photodiode for scanning near-field optical microscope.* Japanese Journal of Applied Physics, Part 1: Regular Papers and Short Notes and Review Papers, 2000. **39**(12): p. 7150.

60. Ito, K., et al., *Cavity-SNOM (scanning near-field optical microscopy) head using a laser diode.* Japanese Journal of Applied Physics, Part 1: Regular Papers and Short Notes and Review Papers, 1998. **37**(6B): p. 3759.

61. Muramatsu, H., et al., *Self-sensitive probe composed of a piezoelectric tuning fork and a bent optical fiber tip for scanning near-field optical/atomic force microscopy.* Japanese Journal of Applied Physics, Part 1: Regular Papers and Short Notes and Review Papers, 1997. **36**(9A): p. 5753.

62. Satoh, N., et al., *Dynamic-mode AFM using the piezoelectric cantilever: Investigations of local optical and electrical properties.* Applied Surface Science, 2002. **188**(3-4): p. 425.

63. Shang, G., et al., *Detection of shear force with a piezoelectric bimorph cantilever for scanning near-field optical microscopy.* Surface and Interface Analysis, 2001. **32**(1): p. 289.

64. Deal, B.E., and Grove, A. S., *General relationship for the thermal oxidation of silicon.* Journal of Applied Physics, 1965. **36**: p. 3770.

65. Kovacs, G.T.A., Maluf, N. I., and Peterson, K. E., *Bulk micromachining of silicon.* Proceedings of the IEEE, 1998. **86**: p. 1536.

66. Proksche, H., Magorsen, G. , and Ross, D., *The influence of NH_4F on the etch rates of undoped SiO_2 in buffered oxide etch.* Journal of the Electrochemical Society, 1992. **139**: p. 521.

67. Larmer, F.a.S., A., *Method for anisotropically etching silicon.* 1992: Germany.

68. McAuley, S.A., Ashraf, H., Atabo, L., Chambers, A., Hall, S., Hopkins, J. and Nicholls, G., *Silicon micromachining using a high-density plasma source.* Journal of Physics D: Applied Physics, 2001. **34**: p. 2769.

69. Ebbesen, T.W., et al., *Extraordinary optical transmission through sub-wavelength hole arrays.* Nature, 1998. **391**: p. 667.

70. Ghaemi, H., et al., *Surface Plasmons Enhance Optical Transmission Through Sub-wavelength Holes.* Physical Review B, 1998. **58**: p. 6779.

71. Grupp, D.E., et al., *Beyond the Bethe Limit: Tunable Enhanced Light Transmission Through a Single Subwavelength Aperture.* Advanced Materials, 1999. **11**: p. 860.

72. Wang, G.P., Yi, Y. and Wang, B, *Evanescent coupling of transmitted light through an array of holes in metallic film assisted by transverse surface current.* Journal of Physics: Condensed Matter, 2003. **15**: p. 8147.

73. Krishnan, A., Thio, T., Kim, T. J., Lezec, H. J., Ebbesen, T. W., Wolff, P. A., Pendry, J., Martin-Moreno, L. and Garcia-Vidal, F. J., *Evanescently coupled resonance in surface plasmon enhanced transmission.* Optics Communications, 2001. **200**: p. 1.

74. Martin-Moreno, L., Garcia-Vidal, F. J., Lezec, H. J., Pellerin, K. M., Thio, T., Pendry, J. and Ebbesen, T. W., *Theory of Extraordinary Optical Transmission through Subwavelength Hole Arrays.* Physical Review Letters, 2001. **86**(6): p. 1114.

75. Lezec, J.a.T., T., *Diffracted evanescent wave model for enhanced and suppressed optical transmission through subwavelength hole arrays.* Optics Express, 2004. **12**: p. 3629.

76. Dogariu, A., Thio, T., Wang, L. J., Ebbesen, T. W. and Lezec, H. J., *Delay in Light Transmitted through very Small Apertures.* Optics Letters, 2001. **26**: p. 450.

77. Stavrinou, P.N., and Solymar, L., *Pulse delay and propagation through subwavelength metallic slits.* Physical Review E, 2003. **68**: p. 066604.

78. Porto, J.A., Garcia-Vidal, F. J., and Pendry, J. B., *Transmission Resonances on Metallic Gratings with Very Narrow Slits.* Physical Review Letters, 1999. **83**(14): p. 2845.

79. Novotny, L., W. Pohl Dieter, and B. Hecht, *Light confinement in scanning near-field optical microscopy.* Ultramicroscopy, 1995. **61**(1-4): p. 1.

80. Liao, X. and A. Higgins Daniel, *Near-field scanning optical microscopy studies of a fluorescent polyelectrolyte-surfactant complex.* Langmuir, 2001. **17**(20): p. 6051.

81. Kwak, E.S., K. Tai Jong, and D.A. Vanden Bout, *Fluorescence lifetime imaging with near-field scanning optical microscopy.* Analytical Chemistry, 2001. **73**(14): p. 3257.

82. Hosaka, N. and T. Saiki, *Near-field fluorescence imaging of single molecules with a resolution in the range of 10 nm.* Journal of Microscopy, 2001. **202**(2): p. 362.

83. Hamann, H.F., et al., *Molecular fluorescence in the vicinity of a nanoscopic probe.* Journal of Chemical Physics, 2001. **114**(19): p. 8596.

84. Enderle, T., et al., *Near-field fluorescence microscopy of cells.* Ultramicroscopy, 1998. **71**(1-4): p. 303.

85. Ambrose, W.P., et al., *Single molecule detection and photochemistry on a surface using near-field optical excitation.* Physical Review Letters, 1994. **72**(1): p. 160.

86. Pagnot, T., et al., *Use of a scanning near-field optical microscope architecture to study fluorescence and energy transfer near a metal.* Optics Letters, 1997. **22**(2): p. 120.

87. Bopp, A.M., et al., *Super-resolution fluorescence imaging of single dye molecules in thin polymer films.* Journal of Vacuum Science and Technology A, 1997. **15**(3): p. 1423.

88. Beuthan, J., O. Minet, and G. Muller, *The spatial resolution of near-field optical microscope on chromosomes and cell traces.* IEEE Journal on Selected Topics in Quantum Electronics, 2001. **7**(6): p. 894.

89. Wiegraebe, W., et al., *Scanning near-field optical microscope: A method for investigating chromosomes.* Surface and Interface Analysis, 1997. **25**(7-8): p. 510.

90. Subramaniam, V., et al., *Scanning near-field optical microscopy and microspectroscopy of green fluorescent protein in intact Escherichia coli bacteria.* Journal of Fluorescence, 1997. **7**(4): p. 381.

91. Cordero Steven, R., D. Weston Kenneth, and K. Buratto Steven, *Near-field microscopy of collapsed Langmuir-Blodgett films.* Thin Solid Films, 2000. **360**(1-2): p. 139.

92. Vaccaro, L., et al., *Near field optical investigations of Langmuir-Blodgett monolayers in liquid environment.* Langmuir, 2000. **16**(7): p. 3427.

93. Kirsch, A.K., et al., *Fluorescence SNOM of domain structures of LB films containing electron transfer systems.* Ultramicroscopy, 1998. **71**(1-4): p. 295.

94. Adams David, M., et al., *Electric field modulated near-field photo-luminescence of organic thin films.* Journal of Physical Chemistry B, 2000. **104**(29): p. 6728.

95. Fontana, E., *Theoretical and Experimental Study of the Surface Plasmon Resonance Effect on a Recordable Compact Disk.* Applied Optics, 2004. **43**: p. 79.

96. Guentherodt, G., et al., *New concepts for magneto-optic data storage: MnBi revisited and magneto-optic SNOM.* Journal of Magnetism and Magnetic Materials, 1997. **175**(1-2): p. 63.

97. Betzig, E., et al., *Near-Field Magneto-Optics and High Density Data Storage.* Applied Physics Letters, 1992. **61**(2): p. 142.

98. Tadokoro, T., T. Saiki, and H. Toriumi, *Design and implementation of near-field scanning optical microscope for observation of interfacial liquid crystal orientation.* Japanese Journal of Applied Physics, Part 2: Letters, 2002. **41**(2): p. L152.

99. Eggers, G., et al., *Scanning near-field magneto-optic microscopy: Quantitative measurements of local Faraday effects.* Surface and Interface Analysis, 1997. **25**(7-8): p. 483.

Chapter 3: Modeling of Near-Field Optical Devices

3.1 Interaction of Light with Metallic Nano-Structures

When considering nano-scale optical devices and structures, it is extremely important to have a theoretical formulation that properly describes the characteristics of near-field light. The ability to model continuous wave (CW) and pulsed light propagation as well as the capacity to accurately describe the properties of dielectric and metal materials is crucial.

3.2 Finite-Difference Time-Domain Algorithm

Although static beam propagation modeling methods exist, the finite-difference time-domain algorithm (FDTD) is able to more accurately describe the time-dependent dynamics of both pulsed and CW light propagation. Moreover, the FDTD formalism is also conducive to arbitrary frequency dependent material parameters leading to a precise representation of conductive metallic materials, as well as frequency dependent dielectrics.

The FDTD algorithm was first proposed by Yee in 1967 as a second order accurate numerical solution to the differential form of Maxwell's equations [1]. Cartesian grids for the electric and magnetic fields, as seen in Figure 3.1 are offset by a half step in space and time yielding a complete representation of the EM field throughout the computational domain. In a leap-frog fashion, update equations are used to incrementally advance the electric and magnetic fields forward in time and space.

3.2.1 Numerical Solution of Maxwell's Equations

These update equations are extracted from Maxwell's equations in differential form

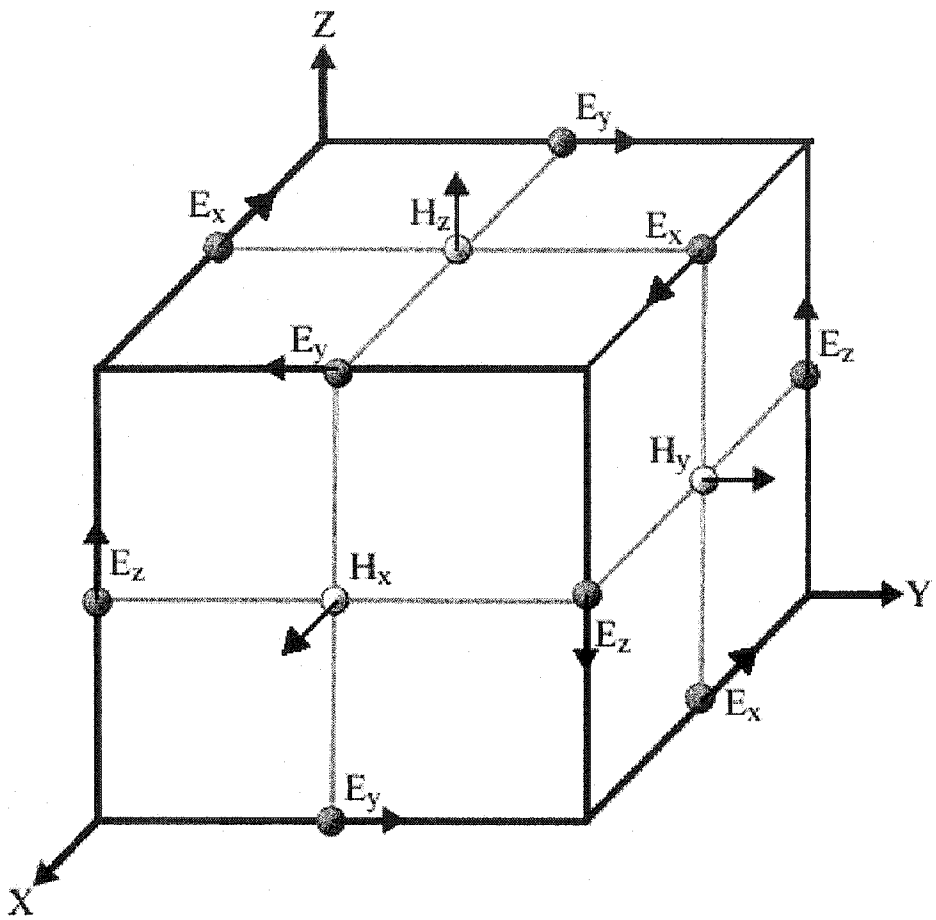


Figure 3.1: The Yee Cartesian unit cell. Electric and magnetic fields are a half-step apart in both space and time.

$$\vec{D} = \varepsilon(\omega)\vec{E}, \tag{3.1}$$

$$\frac{\partial \vec{D}}{\partial t} = \nabla \times \vec{H}, \tag{3.2}$$

and

$$\frac{\partial \vec{H}}{\partial t} = -\frac{1}{\mu_o}\nabla \times \vec{E}, \tag{3.3}$$

where $\vec{D}, \vec{E}$, and $\vec{H}$ are the displacement, electric and magnetic field vectors respectively, μ_o is the permeability of free space, and $\varepsilon(\omega)$ is the frequency dependent dielectric function. These three equations can then be written as the following set of nine scalar equations in Cartesian space:

$$E_x = \frac{1}{\varepsilon(\omega)} D_x \tag{3.4}$$

$$E_y = \frac{1}{\varepsilon(\omega)} D_y \tag{3.5}$$

$$E_z = \frac{1}{\varepsilon(\omega)} D_z \tag{3.6}$$

$$\frac{\partial D_x}{\partial t} = \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \tag{3.7}$$

$$\frac{\partial D_y}{\partial t} = \frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} \tag{3.8}$$

$$\frac{\partial D_z}{\partial t} = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \tag{3.9}$$

$$\frac{\partial H_x}{\partial t} = \frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} \tag{3.10}$$

$$\frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z} \tag{3.11}$$

$$\frac{\partial H_z}{\partial t} = \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}. \tag{3.12}$$

By defining the grid coordinates as $(i, j, k, n) = (i\Delta x, j\Delta y, k\Delta z, n\Delta t)$ where $\Delta s = \Delta x = \Delta y = \Delta z$ is one space step, and Δt is one time step, the spatial and temporal derivatives of any function can be written as

$$\frac{\partial F^n(i,j,k)}{\partial x} = \frac{F^n(i+1/2,j,k) - F^n(i-1/2,j,k)}{\Delta x}, \tag{3.13}$$

and

$$\frac{\partial F^n(i,j,k)}{\partial t} = \frac{F^{n+1/2}(i,j,k) - F^{n-1/2}(i,j,k)}{\Delta t}. \tag{3.14}$$

Discretizing the scalar Maxwell's equations using this formalism results in the following set of update equations used in the FDTD algorithm [2]:

$$\begin{aligned} D_x^{n+1}(i+1/2,j,k) = {} & D_x^n(i+1/2,j,k) \\ & + \frac{\Delta t}{\Delta s}[H_z^{n+1/2}(i+1/2,j+1/2,k) - H_z^{n+1/2}(i+1/2,j-1/2,k) \\ & + H_y^{n+1/2}(i+1/2,j,k-1/2) - H_y^{n+1/2}(i+1/2,j,k+1/2)] \end{aligned} \tag{3.15}$$

$$\begin{aligned} D_y^{n+1}(i,j+1/2,k) = {} & D_x^n(i,j+1/2,k) \\ & + \frac{\Delta t}{\Delta s}[H_x^{n+1/2}(i,j+1/2,k+1/2) - H_x^{n+1/2}(i,j+1/2,k-1/2) \\ & + H_z^{n+1/2}(i-1/2,j+1/2,k) - H_z^{n+1/2}(i+1/2,j+1/2,k)] \end{aligned} \tag{3.16}$$

$$\begin{aligned} D_z^{n+1}(i,j,k+1/2) = {} & D_z^n(i,j,k+1/2) \\ & + \frac{\Delta t}{\Delta s}[H_y^{n+1/2}(i+1/2,j,k+1/2) - H_y^{n+1/2}(i-1/2,j,k+1/2) \\ & + H_x^{n+1/2}(i,j-1/2,k+1/2) - H_x^{n+1/2}(i,j+1/2,k+1/2)] \end{aligned} \tag{3.17}$$

$$E_x^{n+1}(i,j,k) = \varepsilon(i,j,k) D_z^n(i,j,k) \tag{3.18}$$

$$E_y^{n+1}(i,j,k) = \varepsilon(i,j,k) D_z^n(i,j,k) \tag{3.19}$$

$$E_z^{n+1}(i,j,k) = \varepsilon(i,j,k) D_z^n(i,j,k) \tag{3.20}$$

$$\begin{aligned} H_x^{n+1/2}(i,j+1/2,k+1/2) &= H_x^{n-1/2}(i,j+1/2,k+1/2) \\ &+ \frac{\Delta t}{\Delta s \mu_o}[E_y^n(i,j+1/2,k+1) - E_y^n(i,j+1/2,k) \\ &+ E_z^n(i,j,k+1/2) - E_z^n(i,j+1,k+1/2)] \end{aligned} \tag{3.21}$$

$$\begin{aligned} H_y^{n+1/2}(i+1/2,j,k+1/2) &= H_y^{n-1/2}(i+1/2,j,k+1/2) \\ &+ \frac{\Delta t}{\Delta s \mu_o}[E_x^n(i+1/2,j,k) - E_x^n(i+1/2,j,k+1) \\ &+ E_z^n(i+1,j,k+1/2) - E_z^n(i,j,k+1/2)] \end{aligned} \tag{3.22}$$

$$\begin{aligned} H_z^{n+1/2}(i+1/2,j+1/2,k) &= H_z^{n-1/2}(i+1/2,j+1/2,k) \\ &+ \frac{\Delta t}{\Delta s \mu_o}[E_x^n(i+1/2,j+1,k) - E_x^n(i+1/2,j,k) \\ &+ E_y^n(i,j+1/2,k) - E_y^n(i+1,j+1/2,k)] \end{aligned} \tag{3.23}$$

The electric and magnetic fields are thus updated at each half time step resulting in a complete representation of the fields over a whole time step Δt. Boundary conditions at the edge of the computational space are accounted for via a perfectly matched boundary layer [3] in which incident EM waves are damped due to an impedance matched absorbing layer with exponentially increasing conductivity. Using this method, minimal reflections occur because of the impedance matching between the computational space and the boundary.

3.2.2 Light Sources

Although a complete picture of the electromagnetic field within the computational domain is helpful, a method to actually initiate an EM wave is crucial to accomplishing any useful study. Initially, a sinusoidal, oscillating dipole can be embedded into the FDTD grid by adding a generative term at a single point in space to any of equations 3.18 through 3.20 depending upon the polarization required. This gives the equation

$$E^{n+1}(i,j,k) = \varepsilon(i,j,k)D_z^n(i,j,k) + A(x,y,z,t)\sin(\omega n\Delta t) \tag{3.24}$$

where A is an amplitude function, and ω is the desired frequency. By expanding the source term into a two dimensional plane, and multiplying by the Gaussian amplitude function

$$\frac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/2\sigma^2} \tag{3.25}$$

a continuous wave Gaussian hard source is created where the full-width half-maximum of the beam is

$$FWHM = 2\sigma\sqrt{2\ln 2}. \tag{3.26}$$

Unfortunately, due to the nature of the hard source, the EM wave will propagate in both the forward and backward directions from the source plane.

To alleviate this problem, the total field, scattered field formalism is employed [4]. Here the computational grid is subdivided into two regions. The scattered field region contains only the scattered light and lies outside of the modeled object, whereas the total field region contains both the scattered and incident fields. The source term is specified only at the interface between the two regions thereby minimizing storage requirements, and a second equation,

$$H^{n+1/2}(i,j,k) = \mu(i,j,k)H_z^{n-1/2}(i,j,k) + B(x,y,z,t)\sin(k\Delta x - \omega n\Delta t), \quad (3.27)$$

specifying the contribution from the magnetic field is also required. Here, B is an amplitude function and k is the wave vector of the light source. Details regarding the modifications to the electric and magnetic fields at the interface can be found elsewhere. [2].

Finally, modeling of pulsed sources is easily accounted for by multiplying a time-dependent envelope function to the amplitude function in equation 3.24. For a pulsed Gaussian source, this function takes the form

$$\frac{1}{2\pi\sigma^2} e^{-t^2/2\tau^2} \quad (3.28)$$

where again the temporal FWHM of the electric field is $2\tau\sqrt{2\ln 2}$.

3.2.3 Material Properties, Drude Theory and the Auxiliary Differential Equation

Within the FDTD model, material properties such as refractive index, conductivity, and any non-linear electro-optic or magneto-optic response are accounted for via the material dependent permittivity, or permeability. Often a simple dielectric constant will suffice when dealing with materials in the CW regime as long as the wavelength of the light employed is far from any inherent resonances. When dealing with pulsed sources, however, accurately modeling the optical properties of frequency dependent or highly conductive materials becomes increasingly challenging. In this case, the permittivity is no longer constant and may often comprise a complex function leading to potential instabilities in the FDTD algorithm.

In the case of highly conductive materials, such as metals, the frequency dependent, complex dielectric function can be modeled in various ways, the simplest of which being the Drude model:

$$\varepsilon(\omega) = \varepsilon_{\circ}\varepsilon_{\infty} + \frac{\varepsilon_{\circ}\omega_p^2}{i\omega\nu - \omega^2}, \tag{3.29}$$

where ω_p is the plasma frequency, ν is the damping frequency, ε_{∞} is the dielectric constant as ω approaches infinity, and $\varepsilon_{\circ}$ is the permittivity of free space [5]. Incorporation of the Drude model into the FDTD algorithm is accomplished by means of the Auxiliary Differential Equation formalism [6]. By substituting equation 3.29 into equation 3.1, the following equation is derived:

$$\vec{E}\varepsilon_o\varepsilon_{\infty} i\omega\nu - \vec{E}\varepsilon_o\varepsilon_{\infty}\omega^2 + \vec{E}\omega_p^2\varepsilon_o = \vec{D}i\omega\nu - \vec{D}\omega^2. \tag{3.30}$$

By applying an inverse Fourier transform to equation 3.29, a second order differential equation in time,

$$\nu\frac{d\vec{D}}{dt} + \frac{d^2\vec{D}}{dt^2} = \omega_p^2\varepsilon_o\vec{E} + \nu\varepsilon_{\infty}\varepsilon_o\frac{d\vec{E}}{dt} + \varepsilon_{\infty}\varepsilon_o\frac{d^2\vec{E}}{dt^2}, \tag{3.31}$$

relates the electric field vector to the displacement vector. Finally, equation 3.30 can be discretized to yield

$$\vec{E}\Big|_{i,j,k}^{n+1} = \frac{(\nu\Delta t + 2)\vec{D}\Big|_{i,j,k}^{n+1} - 4\vec{D}\Big|_{i,j,k}^{n} + (2 - \nu\Delta t)\vec{D}\Big|_{i,j,k}^{n-1} + 4\varepsilon_o\varepsilon_{\infty}\vec{E}\Big|_{i,j,k}^{n} - (w_p^2\Delta t^2 - \nu\Delta t\varepsilon_{\infty} + 2\varepsilon_{\infty})\varepsilon_o\vec{E}\Big|_{i,j,k}^{n-1}}{(w_p^2\Delta t^2 + \nu\Delta t\varepsilon_{\infty} + 2\varepsilon_{\infty})\varepsilon_o}, \tag{3.32}$$

which can be employed in conjunction with the FDTD algorithm to model highly conductive objects as well as surface plasmon propagation along metallic surfaces.

3.2.4 FDTD Software Suite

Using the FDTD algorithm outlined above, along with the necessary additions for the source terms and metallic material properties, a software suite for 2D and 3D electromagnetic simulations was designed and developed in C++. All source code for the suite has been included in Appendix A. The FDTD suite employs a useful bitmap scheme for geometry input and electromagnetic field output. Cross-sectional cuts in both the space and time domains were also used as an output method with the data being manipulated and visualized in the Matlab environment. Initial testing of the FDTD algorithm was performed on several test cases including pulse propagation through a linearly dispersive media[2] as well as testing the reflection from a metallic half-plane. Fresnel's coefficients for transmission and reflection were also tested for the case of silica glass. In all cases, good agreement with theory was obtained.

3.3 Modeling of Solid Immersion Near-Field Optical Probes

Within a fiber probe, coupling to SP waves into the metal cladding is severely limited by both the acceptance cone angle of the tapered probe, and the propagating light mode within the fiber. However, when an apertured, metal-coated atomic force microscope (AFM) probe is used instead, the above constraints are eliminated resulting in enhanced intensity throughput (10^2 - 10^3). A modified AFM probe is also extremely conducive to integration with a solid immersion lens thereby decreasing the wavelength cutoff within the probe by a factor equivalent to the refractive index of the material employed. This leads to dramatic increases in intensity throughput. Without sacrificing image resolution and spot size, combining near-field and solid immersion techniques provides several advantages over their fiber counterparts including ease of integration into existing SPM

systems, batch fabrication, and SP-mediated intensity enhancements. In order to determine the optical enhancement engendered by a hybrid near-field solid immersion probe, two configurations are considered: a high index micro-layer probe, and a microsphere probe.

3.3.1 Geometry of the High Index Micro-Layer Near-Field Optical Probe

The micro-layer near-field probe, depicted in Figure 3.2, is comprised of a basic atomic force microscope silicon cantilever and conical, apertured tip [7]. The bi-layer structure of the tip is formed using 150 nm chromium (Cr) and 50 nm silver (Ag) films. The Cr, due to its high Young's modulus, serves as a durable protective layer for the Ag film. However, because of the dielectric function of Cr, as depicted in Figure 3.3, SP waves are not supported at visible wavelengths [8], and can only be coupled to the Ag layer. To effectively reduce the wavelength of the light as it propagates through the aperture, and to maximize intensity throughput, an additional, 1 μm thick, high index layer (n = 3.36) of gallium phosphide (GaP) is evaporated into the conical tip.

The GaP layer is crucial to the transmission enhancement as it provides both the mechanism for decreasing the wavelength cutoff of light within the probe, as well as a means to couple to SP modes. Thus, to develop an accurate representation of the micro-layer near-field probe, the deposition of the GaP film within the probe tip was simulated, by Dr. Steve Dew, using a thin film microstructure simulator (SIMBAD) [9]. The 2D Monte Carlo film growth model depicts the film as an aggregation of a large number ($\sim 5e10^5$) of identical disks, each representing the averaged behavior of many individual atoms. To capture the self-shadowing instability of film growth, these disks are serially launched from just above the film surface and follow straight-line trajectories until they

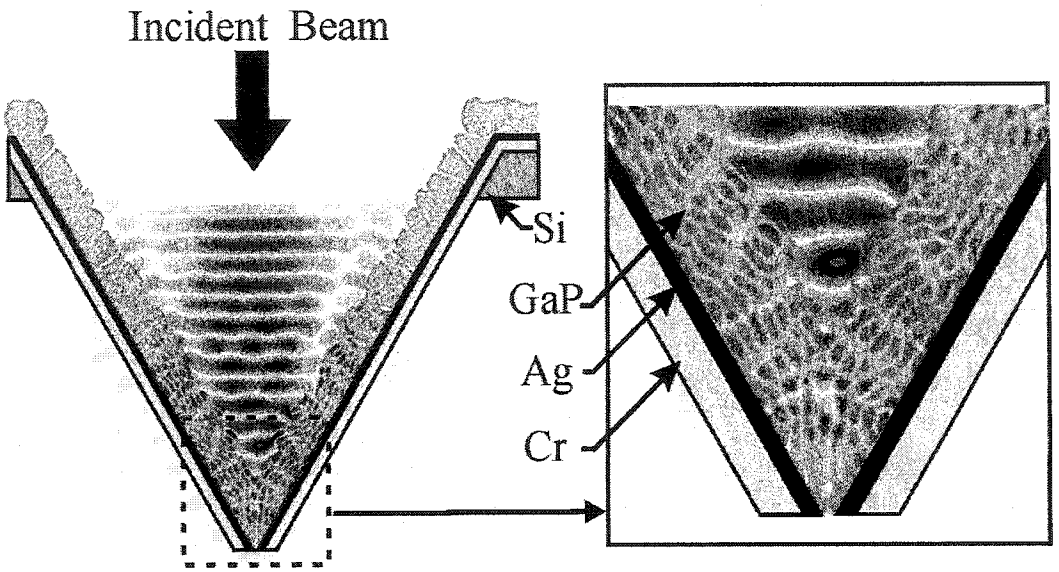


Figure 3.2: Cross-section of the high index micro-layer near-field probe incorporating a 1 μm thick GaP layer, and a dual layer of chromium and silver. TM polarized light is incident from the top.

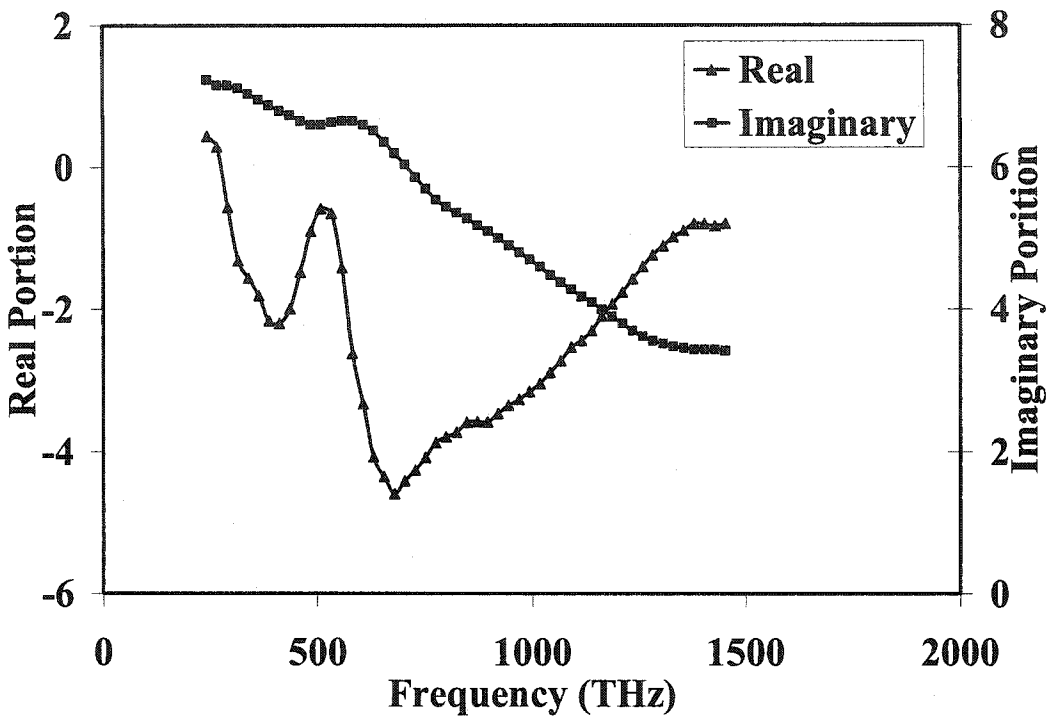


Figure 3.3: Frequency dependent dielectric function of chromium at optical frequencies. The real portion is represented by the squares and the imaginary portion by the triangles [16].

strike the substrate or an already-deposited disk. From the impact point, the disks are then able to diffuse over the surface to a cradle site with high coordination to minimize local surface curvature. The angular distribution used to initially launch the disks was calculated using the SIMSPUD [10] program which models the 3D transport of flux through a physical vapor deposition reactor, incorporating the effects of geometry and gas scattering. While the SIMBAD depiction is 2D, full 3D trajectories are accounted for within the model by assuming cylindrical symmetry, as is appropriate for the conical tip being considered.

3.3.2 Beam Propagation Results for the Micro-Layer Probe

Optical intensity throughput, I, was first determined for varying aperture sizes, a, for a typical near-field probe comprised of an empty Cr tip. Optical intensity throughput was measured as the intensity (W/m^2) transmitted through the probe and detected at a distance 25 nm from the probe apex. In all cases, 800 nm laser light with a 5 μm full-width half-max (FWHM) and a maximum intensity of 1 W/m^2 is incident upon the near-field probe structure. As can be shown in Figure 3.4, light intensity throughput increases with aperture size, closely matching the experimentally demonstrated fourth power dependence ($I \alpha a^4$) [11].

To demonstrate the intensity enhancement produced by the high-index GaP layer, Fig. 2 depicts the intensity throughput as a function of aperture diameter as the refractive index of the micro-layer is increased from 1 (Air) to 3.5 (GaP). As expected, since the 800 nm incident wavelength is effectively reduced to 238 nm within the GaP micro-layer, this probe displays a remarkable increase in transmitted intensity for all aperture sizes.

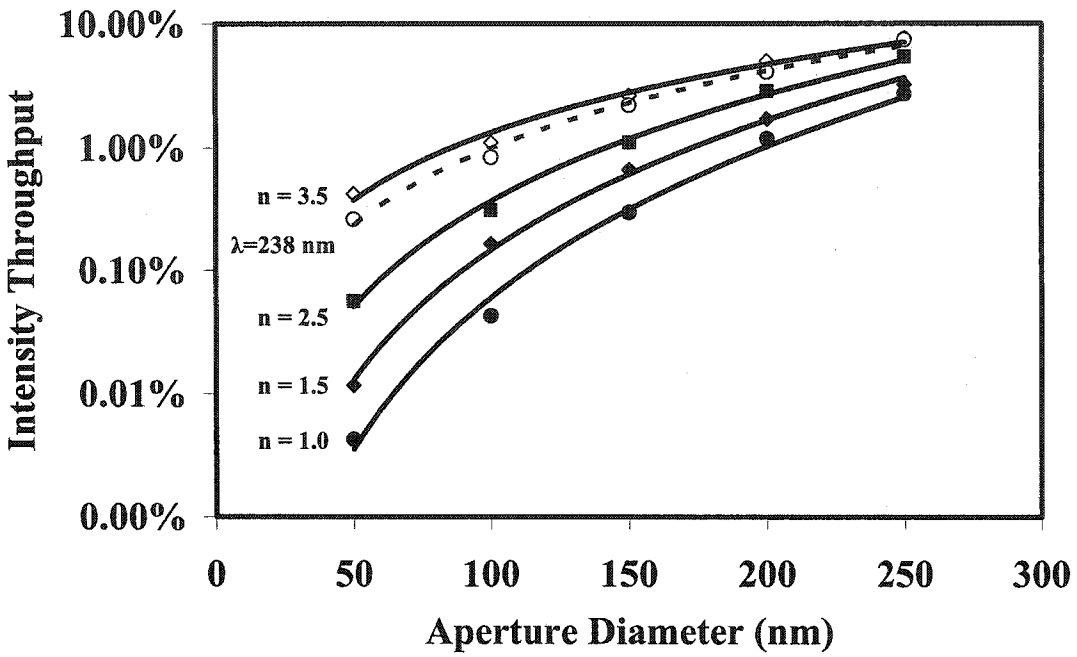


Figure 3.4: Optical intensity throughput as a function of aperture diameter for a dual layered Ag/Cr tip with a 1μm thick high index film with refractive index ranging from 1.0 to 3.5 (solid lines), and for an unfilled Cr tip with 238 nm incident light (dashed line).

The transmission enhancement ranges from a 100-fold increase in intensity throughput for a = 50 nm to a 3-fold enhancement at a = 250 nm. The optical transmission at λ=238 nm for an unfilled near-field probe, also shown in Fig. 3.4, agrees well with the transmission curve of the micro-layer probe. Clearly, this probe design will enable extremely high intensity throughput (10^{-3} to 10^{-1}) when compared to existing near-field fiber-based probes.

3.3.3 Geometry of the Microsphere Near-Field Optical Probe

Although a high index layer leads to increased optical intensity throughput, it should be noted that a fraction of the incident electromagnetic radiation is reflected from the sidewalls of the conical probe or scatters off of the heterogeneous GaP layer leading to non-optimal coupling efficiency. Evidently, an alternative configuration employing a solid immersion lens will increase the coupling efficiency. Displayed in Figure 3.5, the microsphere near-field probe employs a 10 μm diameter silica microsphere (n =1.5) instead of the GaP layer. The microsphere, acting to focus the incident light onto the apex of the probe, is embedded in the conical tip.

3.3.4 Beam Propagation Results for the Microsphere Probe

Transmitted light intensity is again investigated as a function of aperture diameter. The results are depicted in Figure 3.6 where the optical intensity throughput of an unfilled probe is displayed for comparison. When an embedded solid immersion lens is utilized, an extraordinary 800-fold increase in optical transmission is observed for the 50 nm aperture. As expected, larger aperture sizes indicate a tendency towards convergence of the two curves; however, greater than 10% transmission is observed for the microsphere

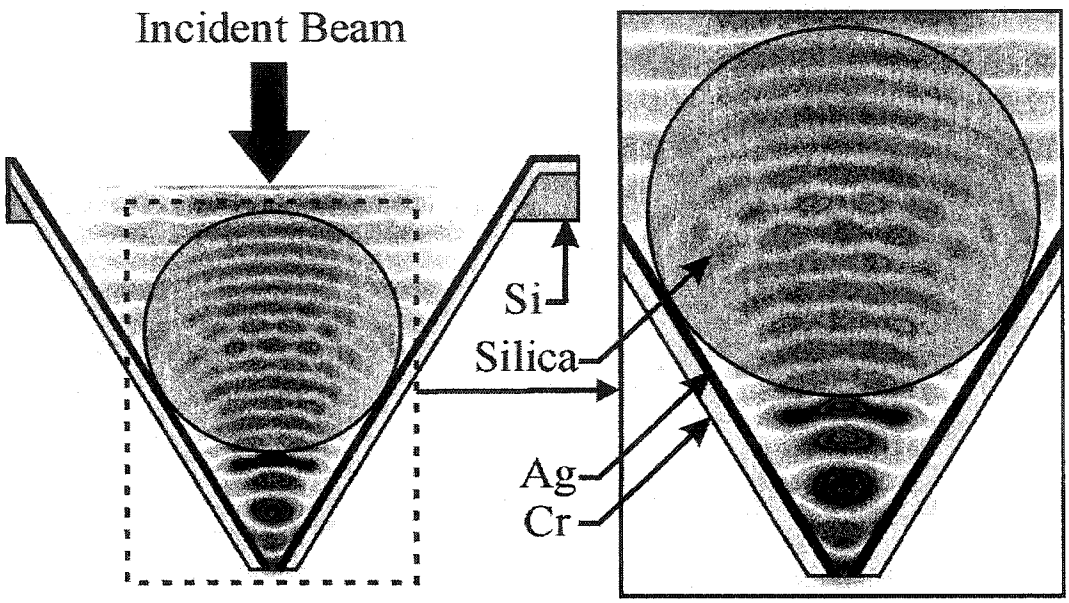


Figure 3.5: Cross-section of the microsphere near-field probe incorporating a 10 μm diameter silica microsphere solid immersion lens. Again the dual layer of chromium and silver is present and TM polarized light is incident from the top.

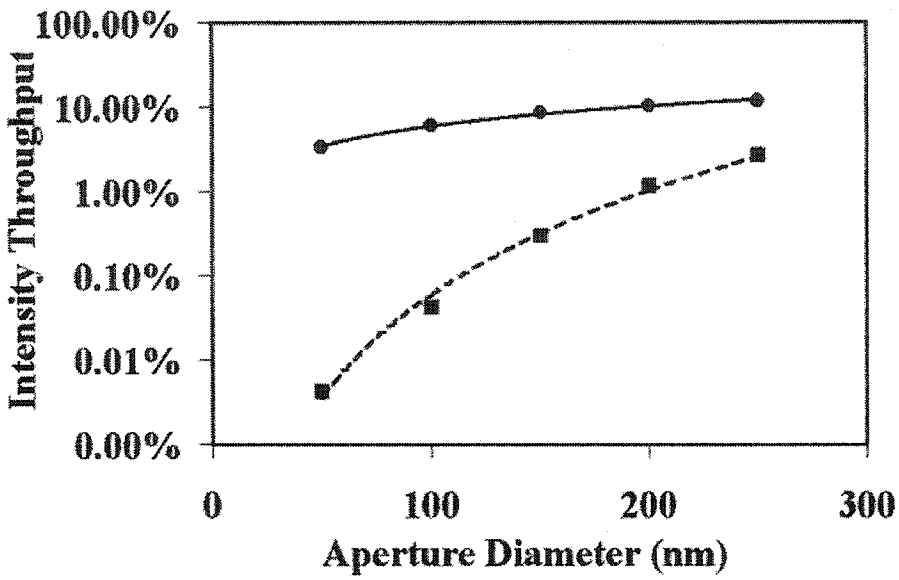


Figure 3.6: Optical intensity throughput as a function of aperture diameter for an unfilled tip (dashed/squares), and for a tip with an embedded silica microsphere solid immersion lens (solid/circles).

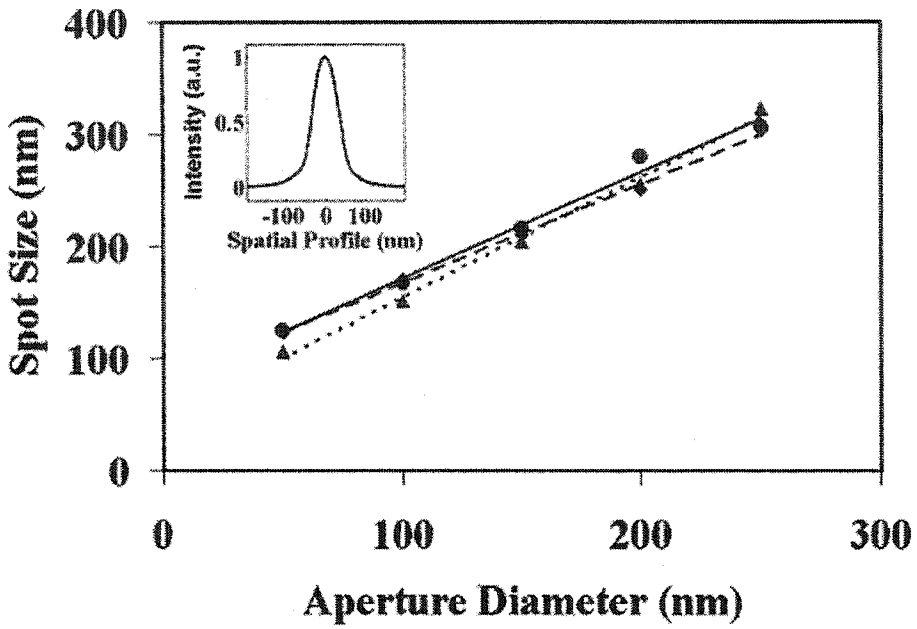


Figure 3.7: Spot size (FWHM) of transmitted light from all three near field optical probes as a function of aperture diameter circles/solid line, squares/dashed line, and triangles/dotted line are the micro-layer, microsphere and unfilled probes respectively. The inset illustrates the measured FWHM of a 50 nm aperture.

near-field probe with 250 nm aperture. This dramatic increase in optical intensity throughput, when compared to an unfilled probe, demonstrates the feasibility of high-throughput near-field optical microscopy.

An important parameter in near-field optical microscopy is the imaging resolution. In order to determine the spatial resolution effects of the hybrid microsphere, and micro-layer near-field probes, the spot size FWHM for both probes, along with an unfilled probe, are measured at a typical distance of 25 nm away from the apex of the probes. These measurements are plotted in Figure 3.7 as a function of aperture diameter. All three probes exhibit a similar linear trend in spot size. Spatial resolution, however, increases with decreasing distance from the probe apex since transmitted light will suffer less divergence. Typical near-field scanning optical microscopes operate at or near contact with the scanned sample; thus, the maximum obtainable resolution should be equal to the aperture diameter of the probe.

3.4 Modeling of Pulse Propagation in a Nano-Metallic Slit Array

Until now, no effort has been made to determine the propagation characteristics of pulses whose durations are comparable to the SP lifetime (< 50 fs). The section below demonstrates that coupling of ultrashort pulses to SPs in a sub-wavelength nano-slit array leads to several distinct re-radiating modes. The dispersion characteristics of the SP resonance also leads to several interesting effects such as a modification of the pulses group velocity, pulse splitting, pulse train re-radiation and even superluminal pulse propagation.

3.4.1 Geometry of the Nano-Metallic Slit Array

The two dimensional FDTD computational grid is composed of an array of 750x750 pixels where each pixel represents 30 nm. These dimensions correspond to one-tenth of the smallest feature size thereby satisfying the minimum grid spacing condition

$$\Delta t \leq \frac{1}{c}\left[\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\right]^{-\frac{1}{2}} \tag{3.33}$$

of the FDTD algorithm. The nano-array, depicted in Figure 3.8, comprises a set of 300 nm wide slits embedded in a 300 nm thick silver film whose plasma frequency, ω_p, and scattering frequency, ν, are taken to be 5.66×10^{15} radians/s and 2.244×10^{14} radians/s, respectively. TM polarized EM radiation in both pulsed and CW cases is normally incident upon the array of nano-slits. Each simulation is completed in approximately 20 minutes on a 2.4 GHz Pentium 4 Xeon processor with 1 gigabyte of random access memory. A typical illustration of the software's output can be seen in Figure 3.9.

3.4.2 Continuous Wave Beam Propagation Results for the Nano-Slit Array

The dispersion and transmission curves for the nano-array were first determined for a wide range of wavelengths spanning 600 nm to 1100 nm. A TM polarized continuous wave beam with a 5 μm FWHM was used as a source for SP excitation. The optical transmission is plotted as a function of wavelength in Figure 3.10 for three separate arrays with period (a_o) equal to 660 nm, 750 nm and 870 nm. Surprisingly, the transmission spectrum for a_o = 870 nm exhibits an extremely large transmission of 90% at a wavelength (λ = 800) that is 2.5 times the slit width. This prediction is very similar to previously reported results demonstrating near perfect transmission for subwavelength

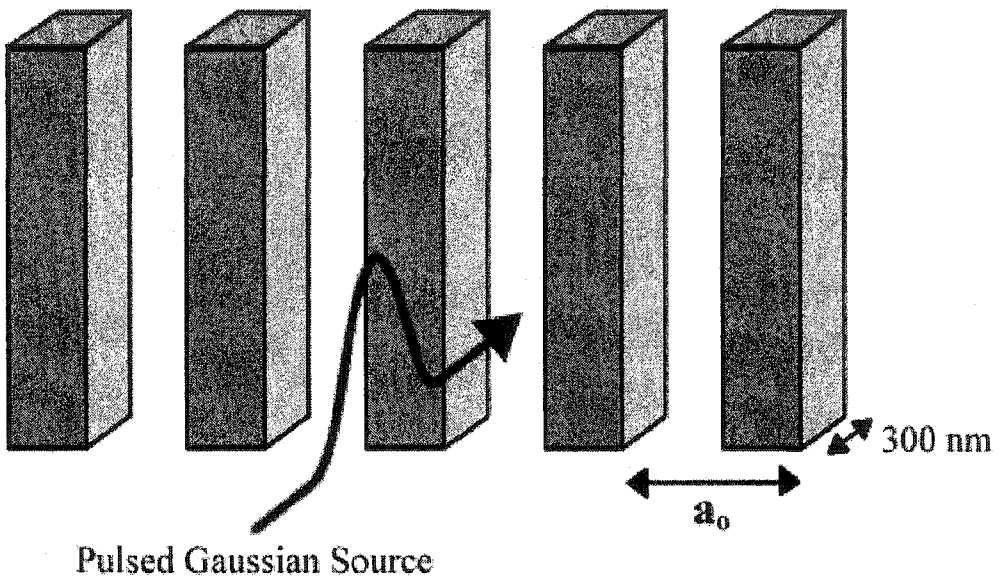


Figure 3.8: The metallic nano-slit array is composed of a set of periodic metallic posts with a 300 nm depth, an arbitrary width, and a periodicity of a_o.

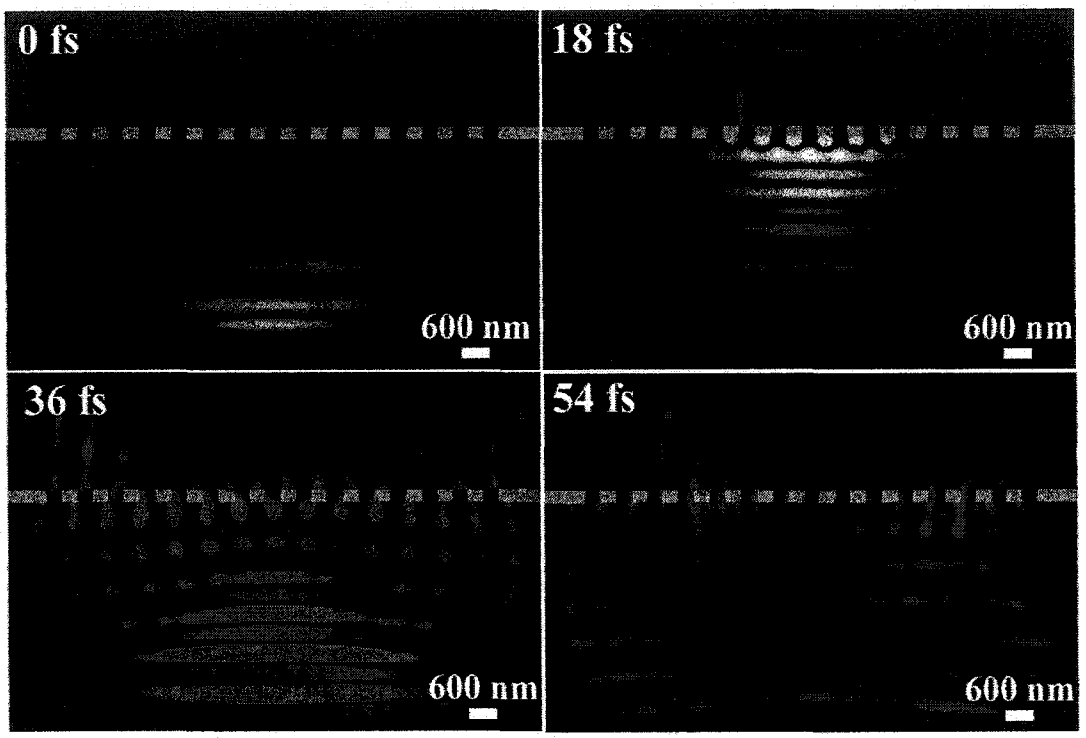


Figure 3.9: Evolution of a 10 fs pulse impingent upon a slit array with 660 nm period. At 0 fs, the approaching pulse is evident, at 18 fs coherent SP re-radiation can be seen, at 36 fs incoherent re-radiation is observed along with the reflected pulse, and at 54 fs only the SP remnants remain.

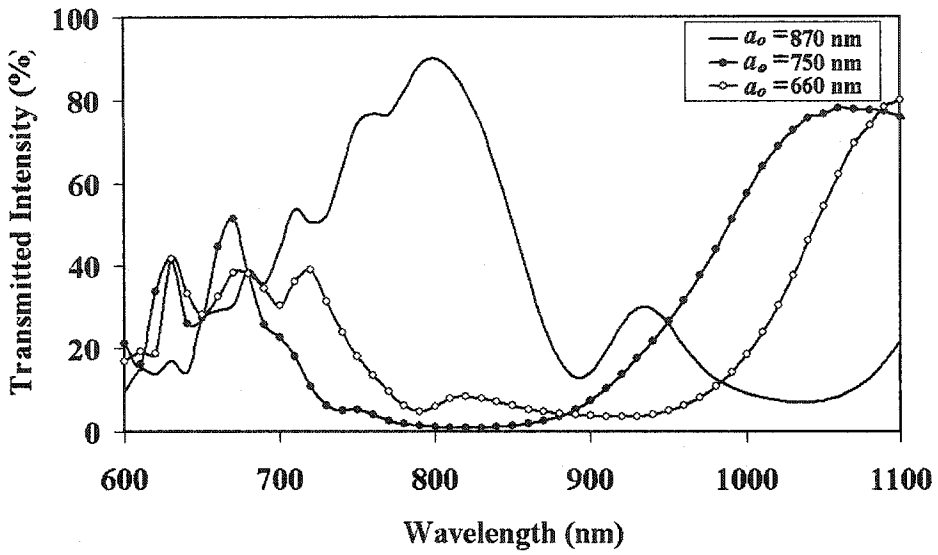


Figure 3.10: Transmitted intensity is plotted as a function of wavelength for 3 representative nano-slit arrays of period 660 nm, 750 nm, and 870nm. Transmission maxima at 800 nm, 1050 nm, and 1100 nm far exceed expectations with > 80% transmission.

metallic slits [12]. Reducing the periodicity of the array to a_o = 750 nm and 660 nm results in a red-shift of the maximum transmission peak (~80% transmission) to 1050 nm and 1100 nm respectively. This red-shift is caused by the modification in the structural line-width of the SP resonance due to the change in the gratings periodicity. The wavelength of the maximum transmission peak, thus strongly depends on the array's periodicity.

The phase delay as a function of wavelength is plotted in Figure 3.11. The relative phase of all three arrays over the wavelength range from 600 nm to 1100 nm is monotonically increasing which is indicative of anomalous dispersion. For the 660 nm periodic array, there exists a negative phase delay below 800 nm. According to Figure 3.11, wavelengths below 800 nm intercept will propagate more slowly than those above since the SP resonance shifts with wavelength. This relative shift from positive to negative delay will have a pronounced effect on an ultrashort pulse with a central wavelength corresponding to the intercept at 800 nm. Such an ultrashort pulse will experience sizeable temporal reshaping due to a shift in its group velocity, v_g.

3.4.3 Femtosecond Pulse Propagation Results for the Nano-Slit Array

This pulse reshaping process was verified for a wide range of pulse durations, each having a 5 um FWHM spot size and a central wavelength of λ = 800 nm corresponding to that of a Ti:Sapphire laser. The transmitted intensity as a function of laser pulse width is depicted in Figure 3.12. Increased transmission is evident for a_o = 870 nm at λ = 800 nm due to the large peak in transmitted intensity displayed in Figure 3.10. For a_o = 750 nm and a_o = 660 nm, the pulses central wavelength of 800 nm lies near transmission minima

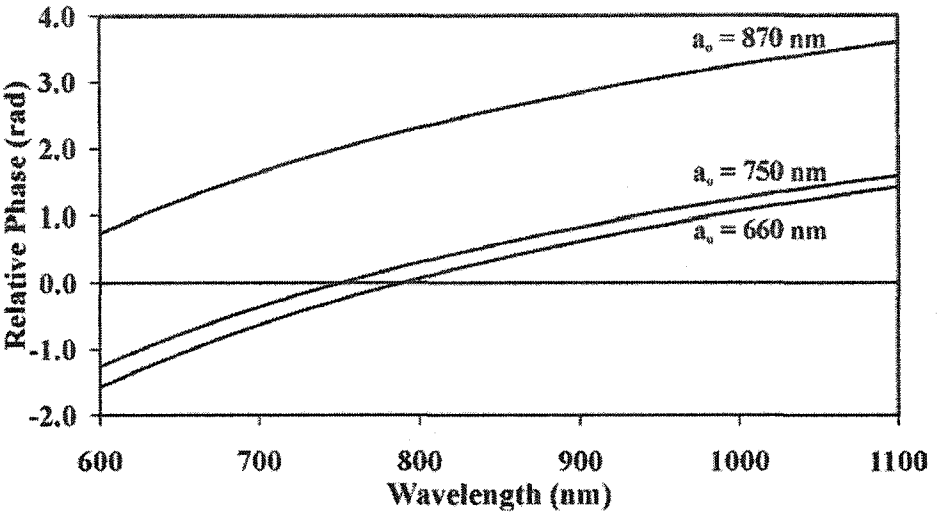


Figure 3.11: Phase for three silver slit arrays with periods of 660, 750 and 870 nm. Negative phase delays are observed for arrays with 660 nm and 750 nm periods.

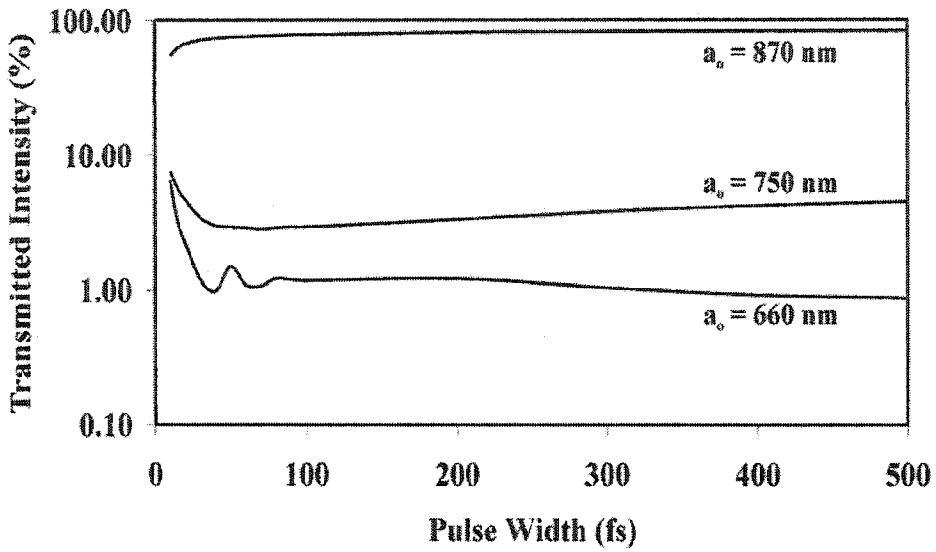


Figure 3.12: Absolute transmitted intensity as a function of incident pulse width is plotted for the three slit arrays. Increased transmission is evident for the array with 870 nm period due to the large peak in transmission at $\lambda = 800$ nm.

and decreased output is evident. The transmitted intensity also varies widely with slit periodicity as demonstrated in Figure 3.13. By normalizing the transmitted intensity throughput to the exposed slit area the increase in transmission is clearly indicated. A maximum in the normalized transmitted intensity is observed for a periodic array with a_o = 900 nm.

Superluminal group velocity ($v_g > c$) is clearly observed in Figure 3.14 that depicts the peak pulse delay for a 10 fs pulse as a function of slit period. Group velocity is stated as the change in frequency with respect to wave vector, $v_g = \frac{\partial \omega}{\partial k}$. Physically, group velocity translates into the speed at which the amplitude of the pulse travels. For anomalously dispersive materials exhibiting a resonance at a specific wavelength such as the metallic grating discussed herein, the group velocity does not equal the signal velocity and can in fact be greater than c [13]. The signal velocity or energy flow velocity only equals the group velocity for normally dispersive materials. For periodic arrays having a_o between 500 nm and 700 nm, pulse advancement is observed. Superluminal velocity is particularly prominent for the a_o = 660 nm array since the pulse's frequency spectrum lies within a broad region of negative phase delay. In the temporal domain this effectively denotes that the frequencies undergoing negative delay will be advanced in time compared to their lower frequency counterparts. Our results reveal that superluminal propagation does not violate the theory of relativity since the signal velocity of the pulse,

$$V_{energy} = \frac{P_{flow}}{P_{stored}} = \frac{\int_{-\infty}^{+\infty} \frac{1}{2} \mathrm{Re}\{E_y H_x^*\} dx}{\int_{-\infty}^{+\infty} \{\frac{\varepsilon_o}{4}|E_y|^2 + \frac{\mu_o}{4}|H_x|^2 dx}, \tag{3.34}$$

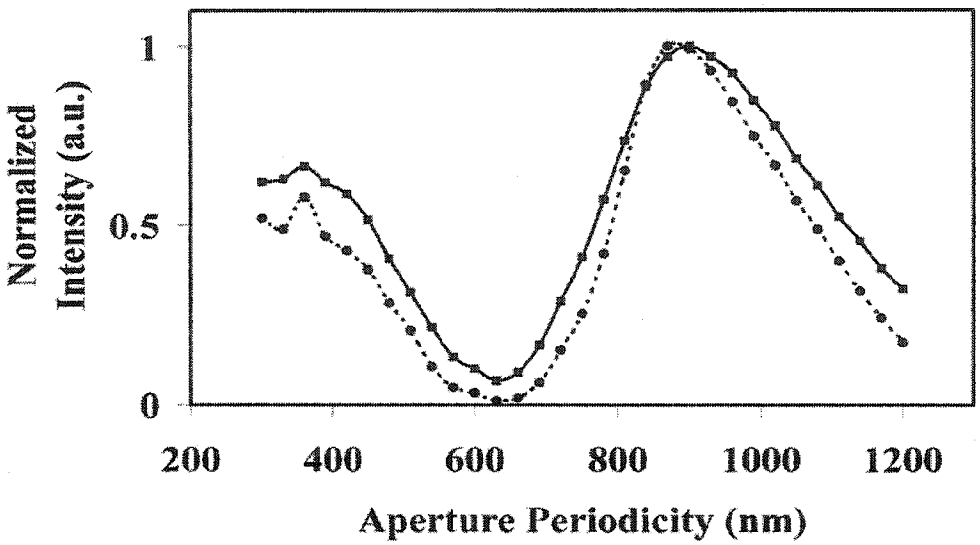


Figure 3.13: Transmitted intensity normalized to the slit area is plotted as a function of slit period for 20 fs (dotted) and 10 fs (solid) pulses centered at λ=800 nm. Larger periods containing less area for throughput actually demonstrate even greater transmission.

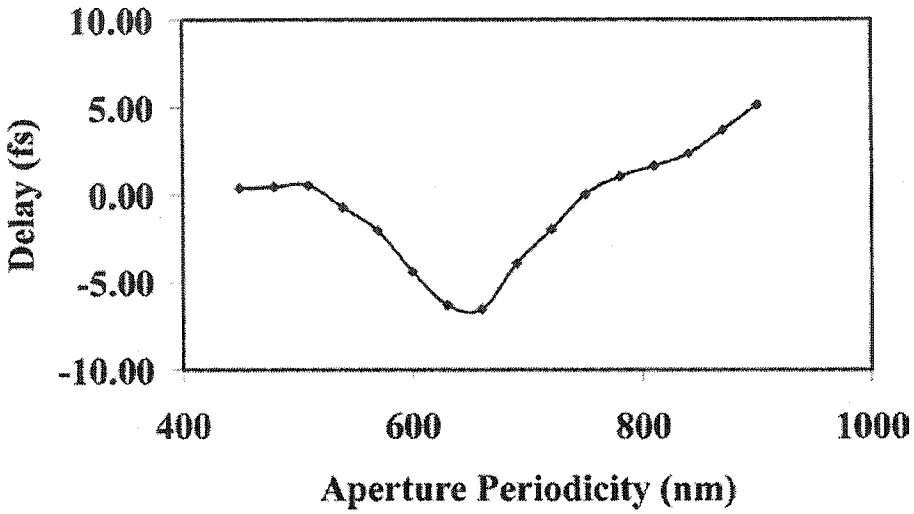


Figure 3.14: Pulse delay as a function of slit period for a 10 fs pulse. Superluminal flow is observed for slit periods between 500 and 750 nm.

was verified to travel at a speed less than the speed of light. The power flow through the nano-metallic slits is divided by the power stored in the wave revealing the true propagation speed of information through the structure, 0.68c.

Furthermore, when pulses on the order of the SP lifetime are used to excite the a_o = 660 nm metallic nano-array, the pulse tends to split into two separate high amplitude pulses, as seen in Figure 3.15, followed by a low amplitude pulse train. The SP generation appears to be coherent for time scales on the order of the 48 fs plasmon lifetime because the SP coherence length, 12.8 μm, is much larger than the slit periodicity [14]. Moreover, the measured SP propagation velocity, 0.45c, indicates that in 48 fs, the SP can propagate 6.48 μm, which again is much larger than the slit periodicity. On time scales larger than the plasmon lifetime (>50 fs), the SPs are incoherently coupled, and the subsequent pulse train is due to the collective interference pattern from equivalent point sources at the edge of each slit. The propagation of the SP along the transmission grating is then best described as a collection of localized surface plasmons that re-radiate at the grating gaps, and induce a secondary surface plasmon oscillation in the neighboring metallic column. For pulses greater than 50 fs, pulse broadening and delay is observed as seen in Figure 3.16. Arrays with a_o = 750 and a_o = 870 nm, as seen in Figure 3.16, reveal a positive pulse delay for all incident pulse widths. As the pulse widths increase towards the continuous wave regime (>0.5 ps), the optical delays induced by each nano-array approach a steady state.

Pulse broadening is also observed for pulse duration's greater than 50 fs for all three representative arrays. A 100 fs pulse traveling through the a_o = 660 nm array is broadened to 111 fs since its frequency components experience positive phase delay. In

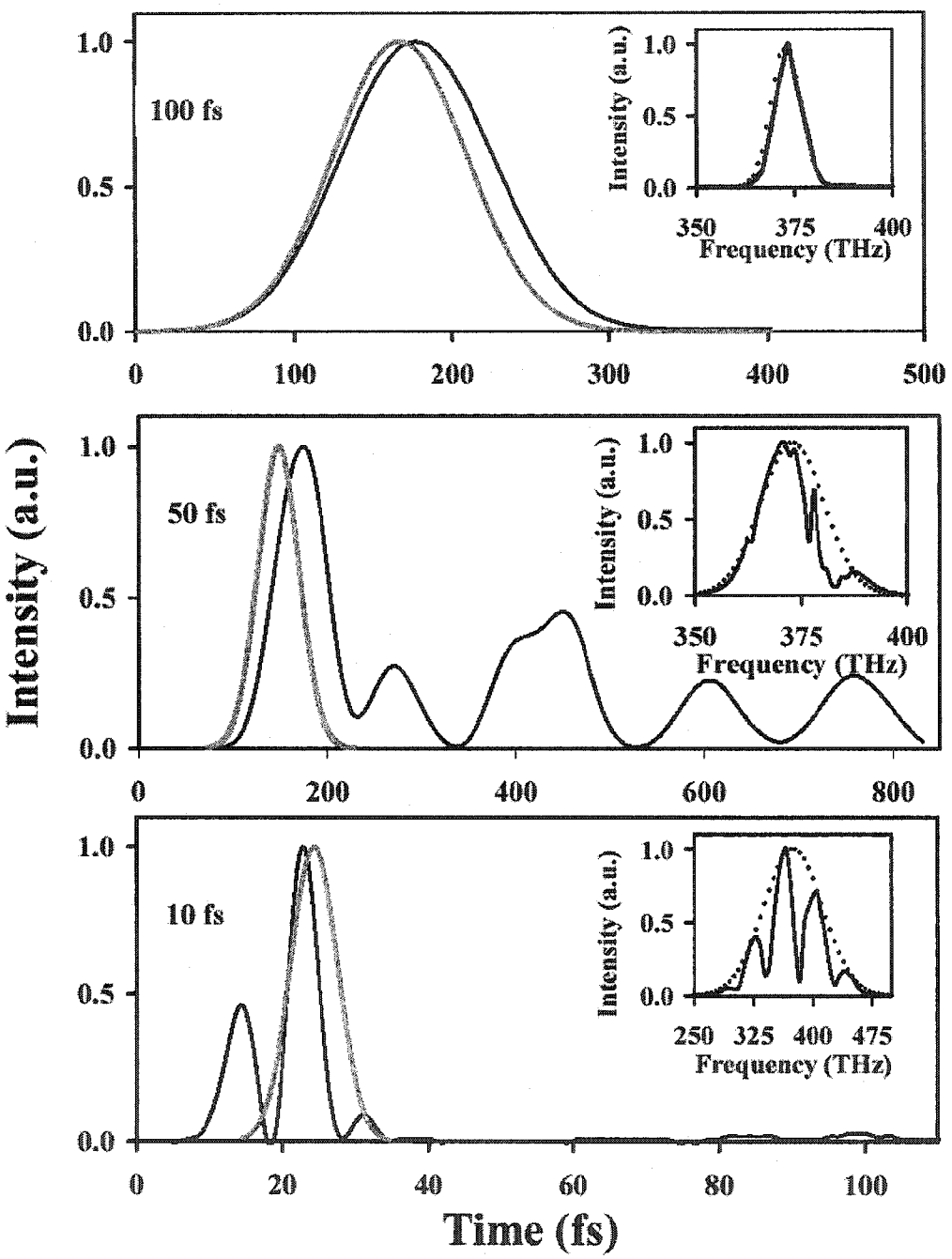


Figure 3.15: Transmitted (black) and incident (grey) pulse envelopes are plotted for pulse widths of 10, 50 and 100 fs at a_o = 660 nm. Pulse train re-radiation is observed for pulses < 50 fs in duration. Superluminal light flow is also apparent for the 10 fs optical pulse. The insets represent the frequency spectrum of the transmitted pulses. All are centered on 375 THz, although pulses less than 50 fs in duration exhibit a modulated spectrum.

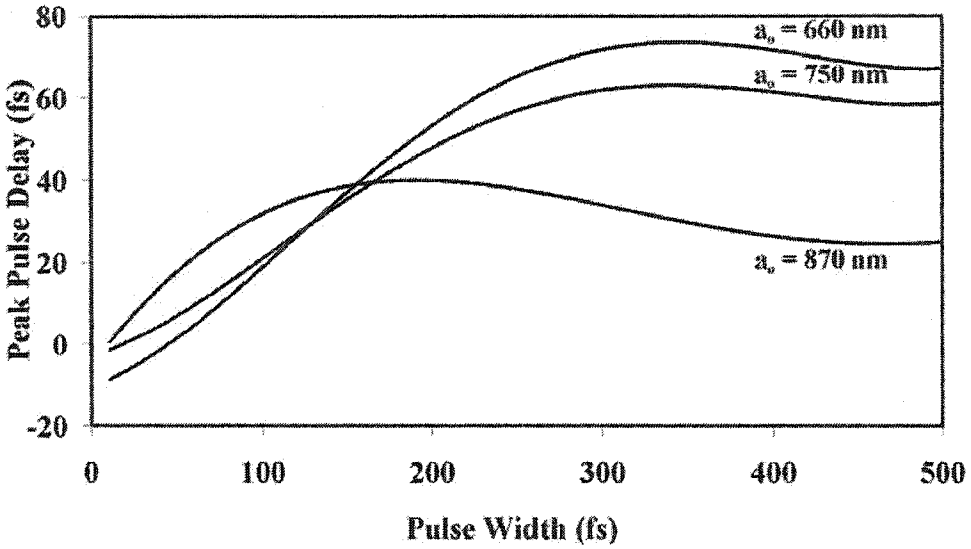


Figure 3.16: Pulse delay is plotted as a function of pulse width for the three nano-slit arrays. As the continuous wave regime is approached the pulse delay approaches a steady state.

addition, the peak of the pulse shows a transit time delay of 19 fs, corresponding to an effective group velocity of $v_g = 0.45c$. Previously reported delays for metallic slit arrays correspond well to the results depicted herein [15, 16].

3.5 Summary

The FDTD technique for modeling near-field nano-optic devices is derived and discussed in detail. Frequency dependent materials and pulsed sources are, as these features are fundamental in modeling and understanding optical phenomena in the near-field. The FDTD suite developed is used to characterize the ultrafast propagation of light in metallic nano-structures as well as to verify the operation of a newly designed set of near-field optical probes. The analysis used here provides a practical framework for designing and optimizing near-field continuous wave and pulsed optical nano-devices.

3.6 References

1. Yee, K.S., *Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media.* IEEE Trans Antennas Propag, 1966. **14**(3): p. 302.

2. Taflove, A., *Computational electrodyamics: The finite-difference time-domain method.* Vol. 6. 1995: Artech House.

3. Berenger, J.P., *A Perfectly Matched Layer for the Absorption of Electromagnetic Waves.* Journal of Computational Physics, 1994. **114**: p. 185.

4. Holland, R.a.W., J. W., *Total-field versus scattered-field finite-difference codes: A comparative assessment.* IEEE Trans. Nucl. Sci., 1983. **NS 30**: p. 4583.

5. Ashcroft, N.W., and Mermin, N. D., *Solid State Physics*. 1976, Saunders College Publishing: New York. p. 1.

6. Kashiwa, T., and Fukai, I., *A treatment by the FD-TD method of the dispersive characteristics associated with electronic polarization.* Microwave Optics Technology Letters, 1990. **3**(6): p. 203.

7. Minh, P.N., T. Ono, and M. Esashi, *Microfabrication of miniature aperture at the apex of SiO sub 2 tip on silicon cantilever for near-field scanning optical microscopy.* Sensors and Actuators, A: Physical, 2000. **80**(2): p. 163.

8. Weeber, J.C., Krenn, J. R., Dereux, A., Lamprecht, B., Lacroute, Y., and Goudonnet, J. P., *Near-field observation of surface plasmon polariton propagation on thin metal stripes.* Physical Review B, 2001. **64**: p. 045411.

9. Dew, S.K., Smy, T., Tait, R. N., Brett, M. J., *Modeling bias sputter planarization of metal films using SIMBAD.* Journal of Vacuum Science and Technology A, 1991. **9**: p. 519.

10. Dew, S.K., Smy, T., Brett, M. J., *Step Coverage, Uniformity and Composition Studies Using Integrated Vapour Transport and Film-Deposition Models.* Japanese Journal of Applied Physics, 1994. **33**: p. 1140.

11. Ebbesen, T.W., et al., *Extraordinary optical transmission through sub-wavelength hole arrays.* Nature, 1998. **391**: p. 667.

12. Porto, J.A., Garcia-Vidal, F. J., and Pendry, J. B., *Transmission Resonances on Metallic Gratings with Very Narrow Slits.* Physical Review Letters, 1999. **83**(14): p. 2845.

13. Stratton, J. A., *Electromagnetic Theory*. 1941, McGraw Hill Book Company Inc.: New York. p. 333.

14. van Exter, M., and Lagendijk, A., *Ultrashort Surface-Plasmon and Phonon Dynamics.* Physical Review Letters, 1988. **60**(1): p. 49.

15. Kim, D.S., Hohng, S. C., Malyarchuk, V., Yoon, Y. C., Ahn, Y. H., Yee, K. J., Park, J. W., Kim, J. Park, Q. H., and Lienau, C., *Microscopic origin of surface plasmon radiation in plasmonic band gap nanostructures.* Physical Review Letters, 2003. **91**: p. 143901.

16. Dogariu, A., Thio, T., Wang, L. J., Ebbesen, T. W. and Lezec, H. J., *Delay in Light Transmitted through very Small Apertures.* Optics Letters, 2001. **26**: p. 450.

17. Palik, E.D., *Handbook of Optical Constants of Solids II*. 1991, Academic Press Inc.: New York. p. 374.

Chapter 4: Microfabrication

4.1 Monolithic Cantilevered Near-Field Probes

Current NSOMs typically employ a tapered metal-coated optical fiber to confine and localize light on the nanometer scale. As previously acknowledged the fiber-tip fabrication process is cumbersome, not conducive to batch fabrication and suffers from low reproducibility [1]. Furthermore, the topographical image quality of the shear-force feedback mechanism employed by these fiber systems is inferior to that of a traditional AFM, as the lateral vibrational motion of the tip (~10 nm) tends to introduce artifacts into both optical and topographical images [2, 3]. Moreover, the fiber tip is usually fragile and prone to destruction when contacted with the sample.

Several researchers have, therefore, turned to combined AFM/NSOM systems [4, 5]. Although this approach has significant advantages over the fiber-based systems since wafer-based microfabrication techniques can be employed, several cantilever-based designs still rely heavily on dual wafer processes in which anodic bonding is used to join the cantilever and holder [6]. This practice results in probes with lower stress tolerances and less mechanical strength since the coefficients of thermal expansion for crystalline silicon and Pyrex glass differ dramatically depending on the temperature [7]. Between 0°C and 300 °C, for example, the linear coefficients of thermal expansion for silicon and glass are 2.6e-6 $^{\circ}C^{-1}$ and 3.25e-6 $^{\circ}C^{-1}$, respectively [8, 9]. Furthermore, the fracture strength of crystalline silicon is 7000 MPa, whereas for bonded silicon/glass the fracture strength is considerably less at 3.75 MPa when bonded at 450°C with an applied voltage of 700 V and a pressure of 1 atmosphere. Most cantilevered AFM/NSOM designs also rely on low throughput fabrication techniques such as electron beam lithography [10] and focused ion beam etching [11].

To circumvent the problems described above we have designed and fabricated a monolithic cantilevered probe and holder for use in either an AFM and/or NSOM system. Using a novel double-sided micromachining process, a hollow, pyramidal silver/chromium (Ag/Cr) tip is formed at the end of a silicon cantilever. The pyramidal tip is apertured to allow subwavelength light throughput, and the silicon cantilever, and holder are fabricated from a single silicon wafer in order to minimize stress within the structure. Moreover, since conventional micromachining techniques are employed, batch fabrication of multiple probes is possible.

After several life-cycle iterations, the final probe design consists of five major steps each encompassing a lithographic sub-step as well as an etching and/or deposition sub-step. An overview of the design is presented in Figure 4.1 along with a representational cross-section of the NSOM tip. Beginning with the formation of a pyramidal tip, the process then proceeds to the definition of the cantilever on the front-side of the wafer. A through wafer-via is then partially formed on the backside of the cantilever along with the definition of the alignment grooves. Finally, in a single step, the through-wafer via is completed and the cantilevers are released from the underlying substrate. The process is completed after deposition of a thin chromium silver-film.

4.1.1 Pyramidal Tip Formation

The process of forming a pyramidal tip structure involves several stages. The wafers employed were 100 mm wide, 500 μm thick, double side polished, (100) oriented, prime silicon wafers. They were initially cleaned for 30 minutes in Piranha solution consisting of a 3:1 mixture of 95% sulfuric acid and 30% hydrogen peroxide. A hard masking layer

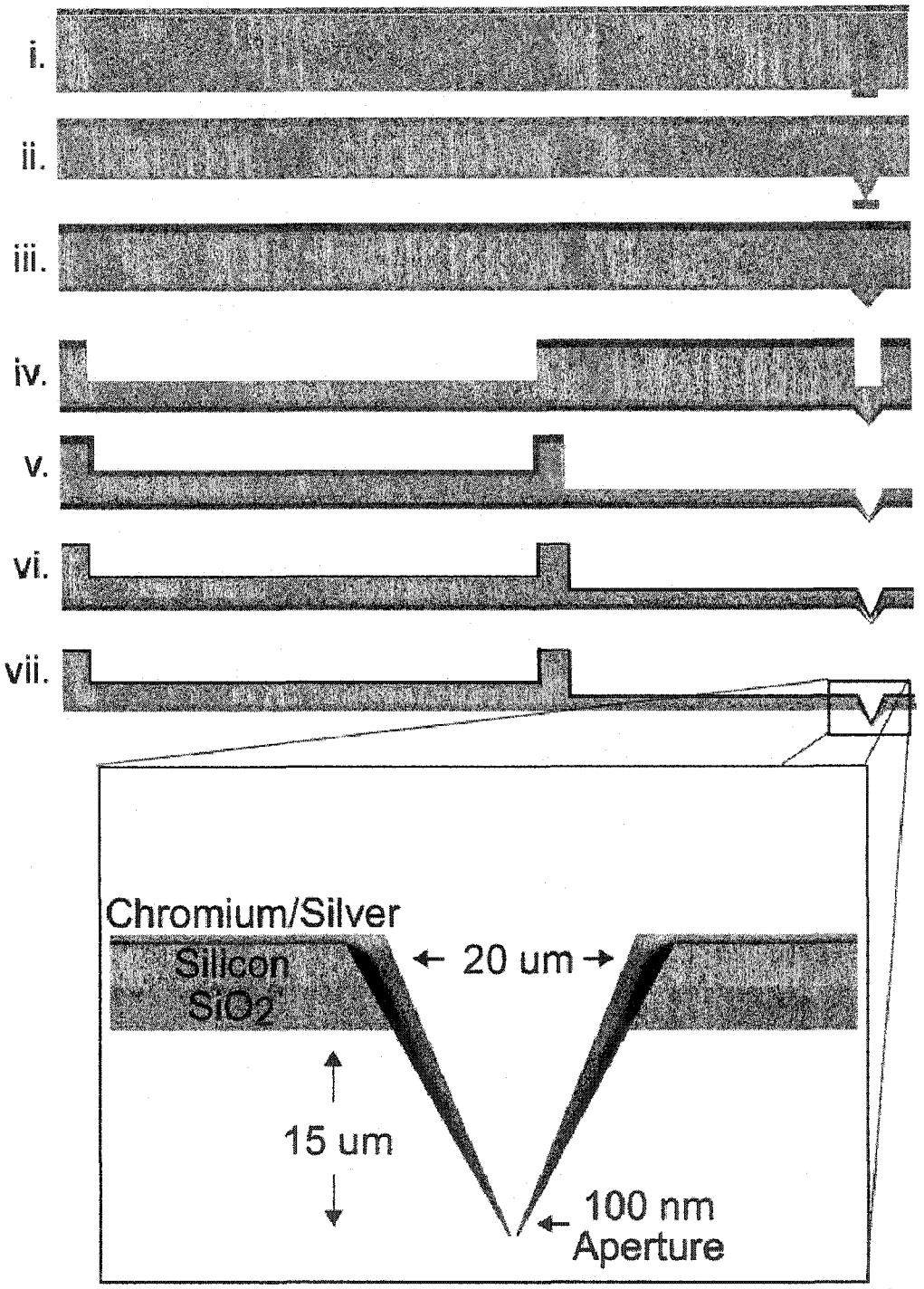


Figure 4.1: The near-field probe design consists of seven steps. A masking layer of SiO_2 is first formed (i), followed by the creation of the pyramidal tip (ii) and cantilever definition (iii). The back-side alignment grooves and through wafer via are then etched (iv), followed by the cantilever release (v). Finally a thin chromium/silver bi-layer is deposited onto the wafer after a brief dip in BOE to form the aperture (vi). The final product and a close up of the tip are seen in (vii).

of SiO_2 was then thermally grown on the wafers. SiO_2 was chosen as the masking layer throughout the fabrication process as silicon nitride deposition was not available and metallic masks are not compatible with anisotropic silicon etching. Thermal oxidation was performed at 1000°C for 8 hours with a constant water temperature of 95°C. These parameters yielded 1.1 μm thick oxide layers with a non-uniformity of less than 1%.

The first photolithographic step was performed using a chromium hard mask containing 96, 25 μm diameter circular masking pads, as seen in Figure 4.2, destined to become 96 devices. The diameter of the masking pads was chosen due to the undercut ratio of the TMAH etching procedure. The diameter chosen enabled tip heights between 10 - 15 μm. A positive photoresist, HPR 504, was spun on the front and back side of the wafers for 40 seconds at 3000 rpm to a final thickness of 0.5 μm. A 3 minute soft-bake of the wafers was conducted in order to harden and remove excess moisture from the resist. Contact lithography was performed using a UV AB-M® mask aligner with a UV wavelength of 435 nm for 4 seconds on the front side of the wafers. The photoresist was subsequently developed for 30 seconds in Microposit® developer 354.

Etching of the SiO_2 hard mask was undertaken for 30 minutes in a solution of BOE as previously described. , after which the photoresist was stripped from the wafer with acetone followed by a second Piranha bath to remove the excess resist and other organic impurities. A front side anisotropic silicon etch step was executed for 26 minutes in 22% TMAH at 110°C thus forming the pyramidal tips. Undercutting of the silicon dioxide circular masking pads was verified using an optical microscope that also served to verify the end point of the etching procedure. A typical SEM image of the resulting

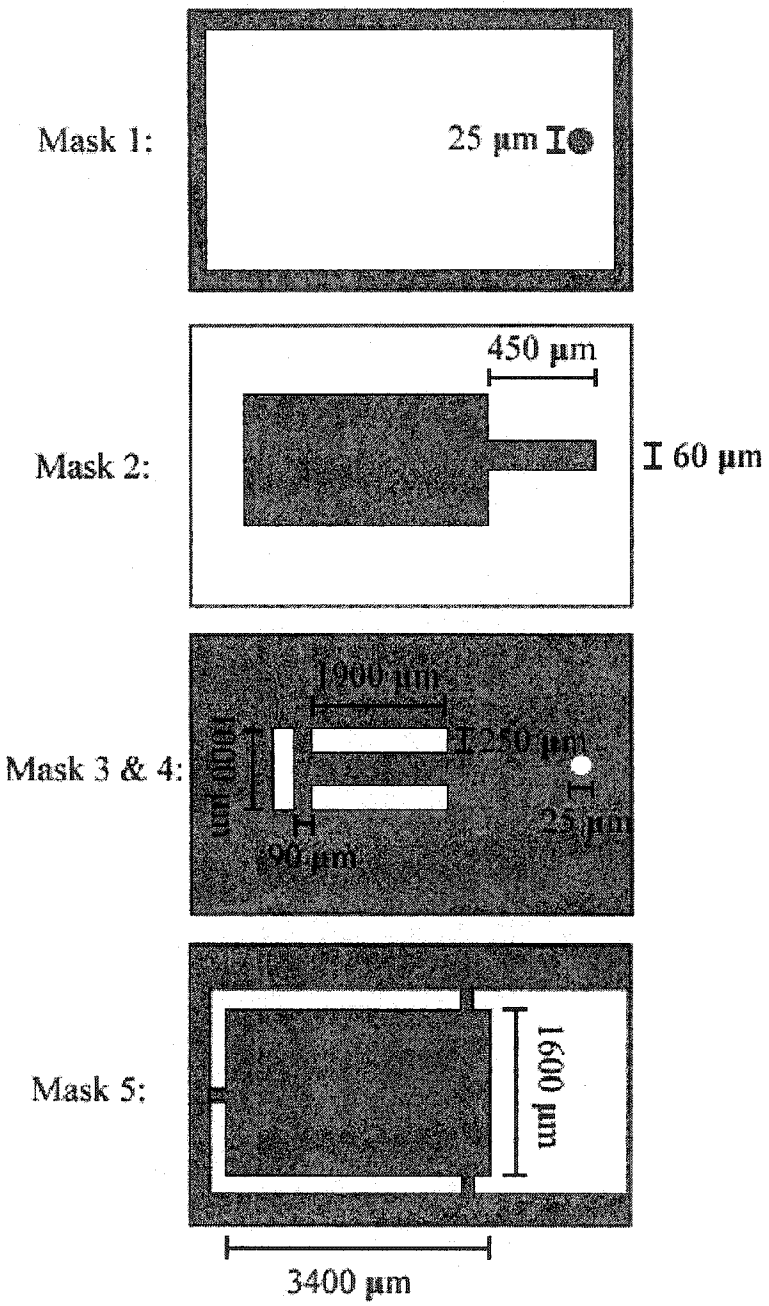


Figure 4.2: Mask designs for the 5 lithographic steps. The first mask contains 96 circular pads 25 µm in diameter used in the creation of the pyramidal tips. The second mask outlines the cantilever and its extension onto the holder. The third and fourth masks were performed in a single lithographic step and outline the through wafer via and backside alignment grooves. The fifth mask defines the holder and spaces for releasing the cantilever.

pyramidal tip can be seen in Figure 4.3. Pyramidal tips ranged in height from 10 – 15 μm and exhibited tip radii ranging in diameter from 100 – 400 nm.

4.1.2 Cantilever Definition

The wafers were re-oxidized for 3 hours at 1000°C in order to form a 500 nm thick masking layer as well as to sharpen the pyramidal tip according to the thermal oxidation anomaly described in Chapter 2. Since the HPR 504 photoresist thickness in the previous step was only 0.5 μm, a more viscous photoresist was employed for the remaining steps as complete structure coverage on the wafer is crucial. Shipley Microposit® 5740 photoresist was chosen for this purpose. This resist was spun on both sides at 1500 rpm for 15 seconds and left to sit for approximately 5 minutes until all air bubbles present in the resist have disappeared. A 6 minute nitrogen backed soft-bake was administered after each coating to solidify the resist. Nitrogen backing is essential if damage to the back-side resist is to be avoided.

Contact photolithography was carried out using the second of five masks as portrayed in Figure 4.2. The cantilever patterns were 450 μm in length by 60 μm in width and were connected to a front-side holding pad 1 mm in width by 2 mm in length. Exposure lasted for 30 seconds followed by development in Microposit® developer 354 for 10 minutes. After development it was noted that the final resist thickness was ~20 μm. The second silicon etch step was again realized in a 22% TMAH etching solution for 8 minutes at 110°C resulting in cantilevers with thicknesses ranging from 3 – 5 μm as seen in Figure 4.4.

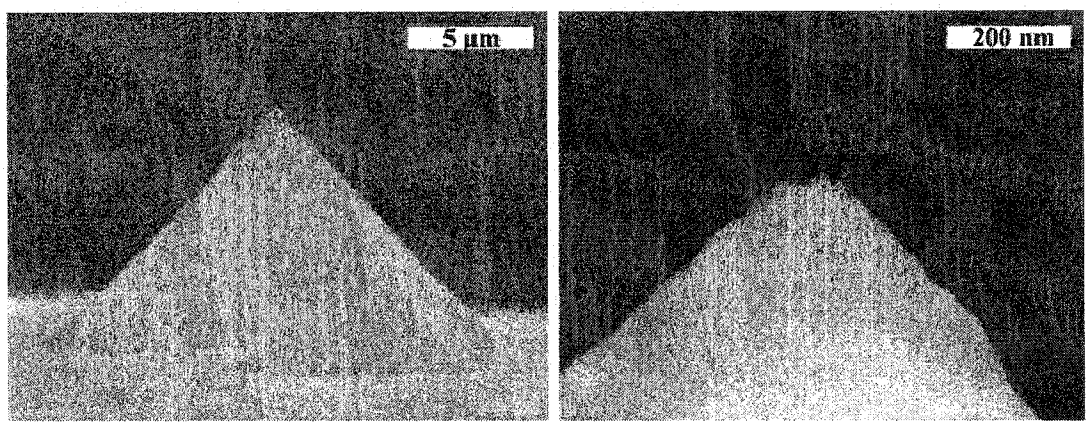


Figure 4.3: Initial results after etching of the pyramidal tip. A well-defined sharp tip with a 100 nm diameter is clearly visible.

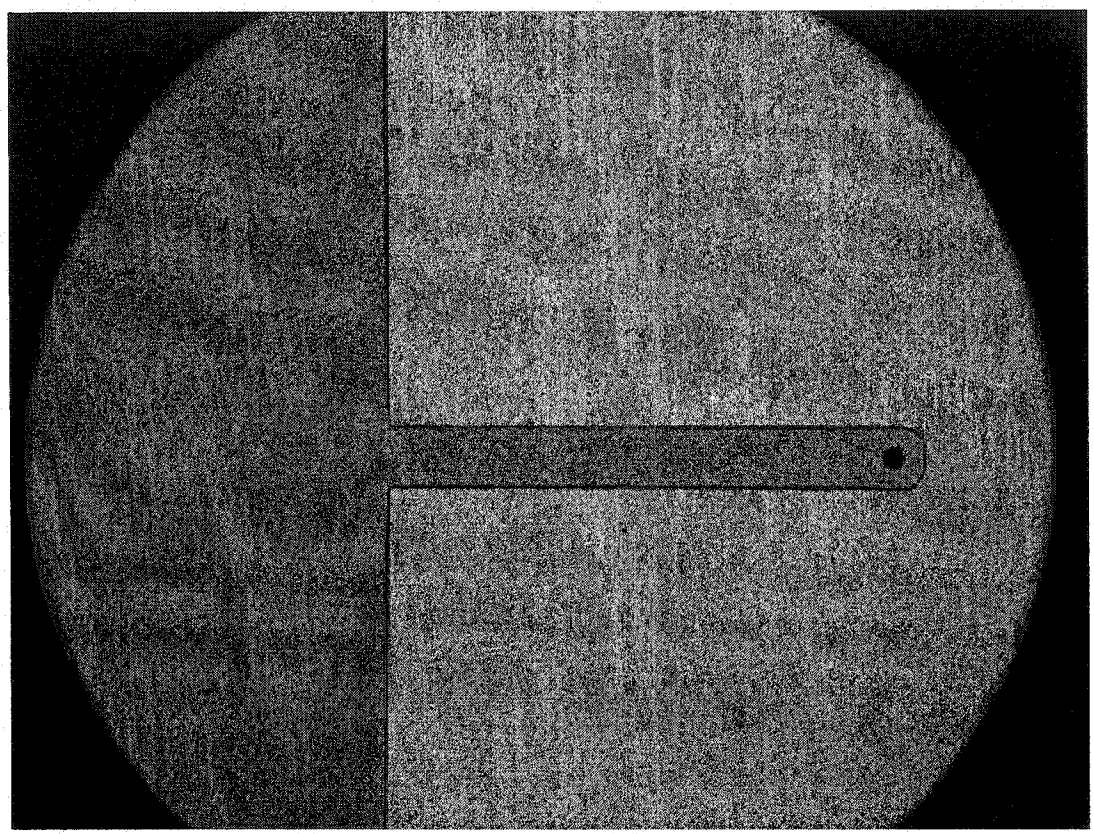


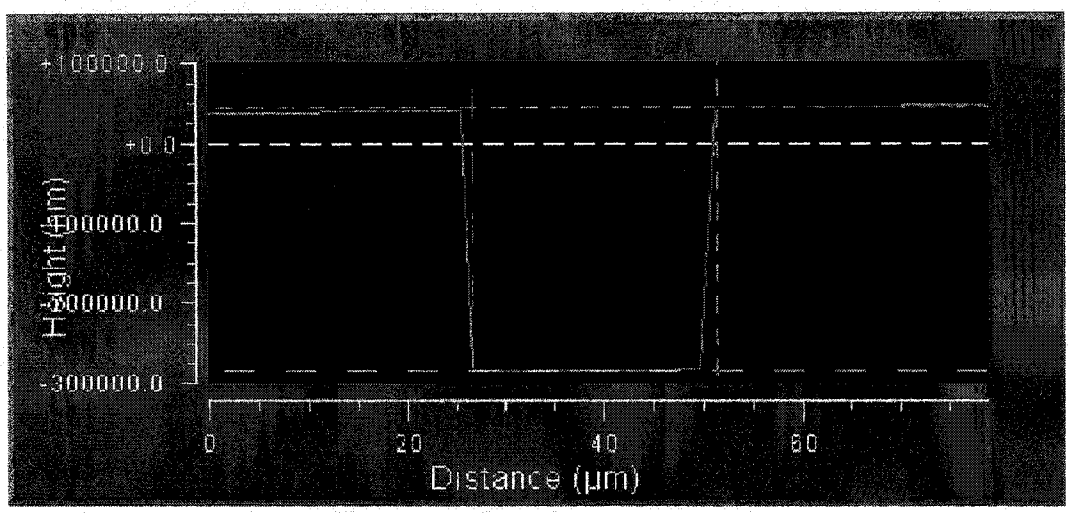
Figure 4.4: Optical microscope image of the newly defined cantilever. The cantilever thickness ranges between 3 and 5 μm. Undercutting of the SiO_2 is apparent at the far-edge of the cantilever.

4.1.3 Through Wafer Vias

Since the probe design entails a hollow metal pyramid on the front-side of the wafer, some means to excavate the silicon from the back-side of the pyramid is required. The solution to this problem took the form of a through-wafer via etch. Although this process was developed with some difficulty, 25 μm wide vias etched completely through the 500 μm wafer were eventually fabricated yielding an aspect ratio of approximately 20:1.

The silicon wafers were coated with a 20 μm layer of Shipley Microposit® 5740 photoresist, and lithography with infrared alignment was carried out for 30 seconds on the backsides of the wafers with both the third (via) and fourth (alignment groove) masks as seen in Figure 4.2. Via etching was completed using the Bosch process, and since selectivity to photoresist is greater that 50:1 no hard mask was needed. Gas flow rates during the IPC RIE process were 200 sccm, and 150 sccm for the C4F8 and SF6 respectively. Chamber pressure was kept constant at 30 mtorr, and 40 W RF power, 12 W RF power and 750 W IPC power were used for the etching, deposition, and inductively coupled plasma respectively. Each cycle consisted of a 6 second deposition phase and a 10 second etch phase.

Since the etch-rate decreases with increasing depth, in excess of 500 cycles was required to etch to a depth of 330 μm as seen in Figure 4.5; however, a single-step through-wafer etch was not necessary. Instead, a preliminary etch of 150 μm was performed followed by a subsequent etch to both complete the via and release the cantilever. The Bosch process was again used with the same parameters for a total of 200



yMax:	nm
yMin:	nm
xMax:	µm
xMin:	µm
yDst: 329599	nm
xDst: 24.8	µm
angle: 85.6976	degs

Figure 4.5: Initial attempts at a deep through-wafer backside etch directly overtop the pyramidal tip were promising. The above graph depicts an optical profilometer measurement revealing a 330 um deep, 25 um wide hole. The sidewall angle is approaching 86°.

cycles with an effective etch rate of 0.75 μm/cycle in order to complete this preliminary etch step.

4.1.4 Alignment Grooves

As the near-field probes were to be used in a pre-existing atomic force microscope arrangement, the holder apparatus had to conform to the existing setup. Backside alignment grooves were, therefore necessary. Alignment grooves on the backside of the wafer consisted of trenches, 250 μm wide and 150 μm deep arranged in a U shape as depicted in Figures 4.2 and 4.6. The bottom groove measured 1000 μm in length and was placed 400 μm from the back edge of the holder. The side grooves, however, measured 1900 μm in length and were placed 100 μm from the bottom groove, and 90 μm from the holder's sides. Because the through-wafer vias were initially etched to a depth of 150 μm, etching of the alignment grooves at the same time proved to be convenient.

4.1.5 Holder Definition and Cantilever Release

The holder structure consists of a large silicon block 3400 μm long by 1600 μm wide to which the cantilever is attached as portrayed in Figure 4.2. 50 μm wide connecting-walls were left standing thereby keeping the holders attached to the wafer until their eventual use. Etching of the holder structure had a two-fold effect. First it would allow the completion of the through wafer via, and second, it would release the silicon cantilever. As the vias would etch more slowly than the holders, as seen in Figure 4.6, the preliminary etch previously described was indeed necessary with an optimum etch depth of 150 μm. After the completion of the lithography phase using the fifth mask depicted in Figure 4.2, ICP RIE was carried out for a total of 450 cycles using the same parameters

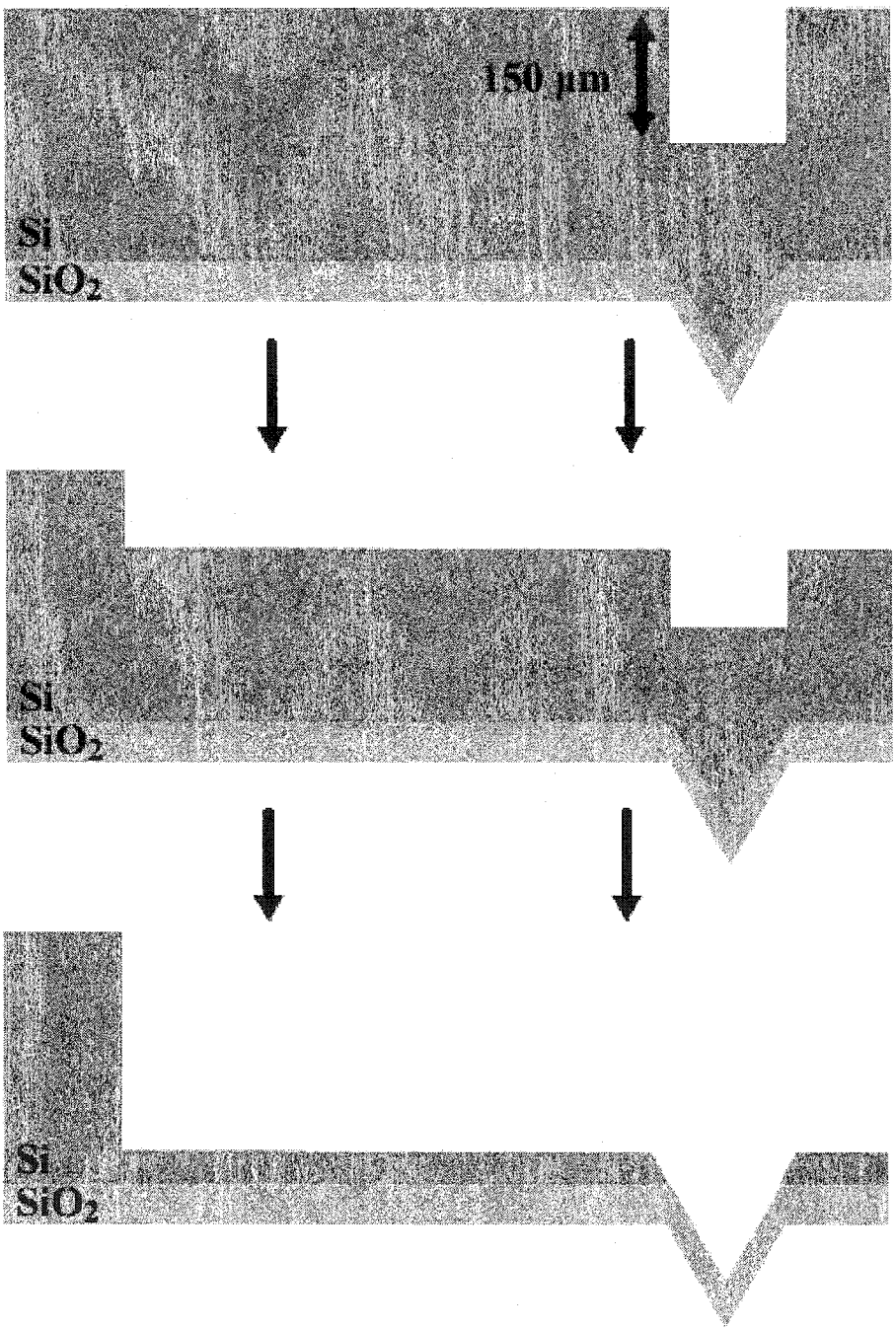


Figure 4.6: Completion of the through-wafer via etch and the release of the cantilever. The through-wafer via etches more slowly due to the increased initial depth.

as before. The final holder structure is portrayed in Figure 4.7, and images of the 450 μm long cantilevers are illustrated in Figure 4.8. The SEM image depicted in Figure 4.9 clearly shows the excavated pyramidal tip as seen from the top side of the cantilever thereby demonstrating that the double sided micromachining process was successful.

The fabricated cantilevers exhibit good, well defined mechanical properties as seen in cross-sectional view depicted in Figure 4.10. The spring constant (k) and resonant frequency (f_o) of the cantilevers described herein are calculated according to

$$f_o = \frac{1}{2\pi}\sqrt{\frac{k}{m}} \tag{4.9}$$

$$k = \frac{Ewt^3}{4L^3} \tag{4.10}$$

$$m = \rho wtL \tag{4.11}$$

where E is Young's modulus of crystalline Si (1.4×10^{11} N/m^2); L, w, and t are the length, width and thickness of the cantilever, respectively; m is the mass of the cantilever, and ρ is the density of Si (2330 Kg/m3). The resonant frequencies and spring constants for the cantilevers produced vary between 6.09 kHz and 0.18 N/m for the 2 μm thick cantilevers to 15.2 kHz and 2.88 N/m for the 5 μm thick cantilevers. For operation in contact AFM sensors, a spring constant in the range of 10^{-2} to 10^2 N/m and a resonant frequency greater than 10 kHz are desired. The cantilevers produced by this novel fabrication process will, therefore, effectively function as AFM probes.

4.1.6 Aperture Formation

Once the cantilevers are released, formation of the aperture at the apex of the SiO_2 tip was a simple procedure. Due to the thinning of the SiO_2 layer at convex corners during

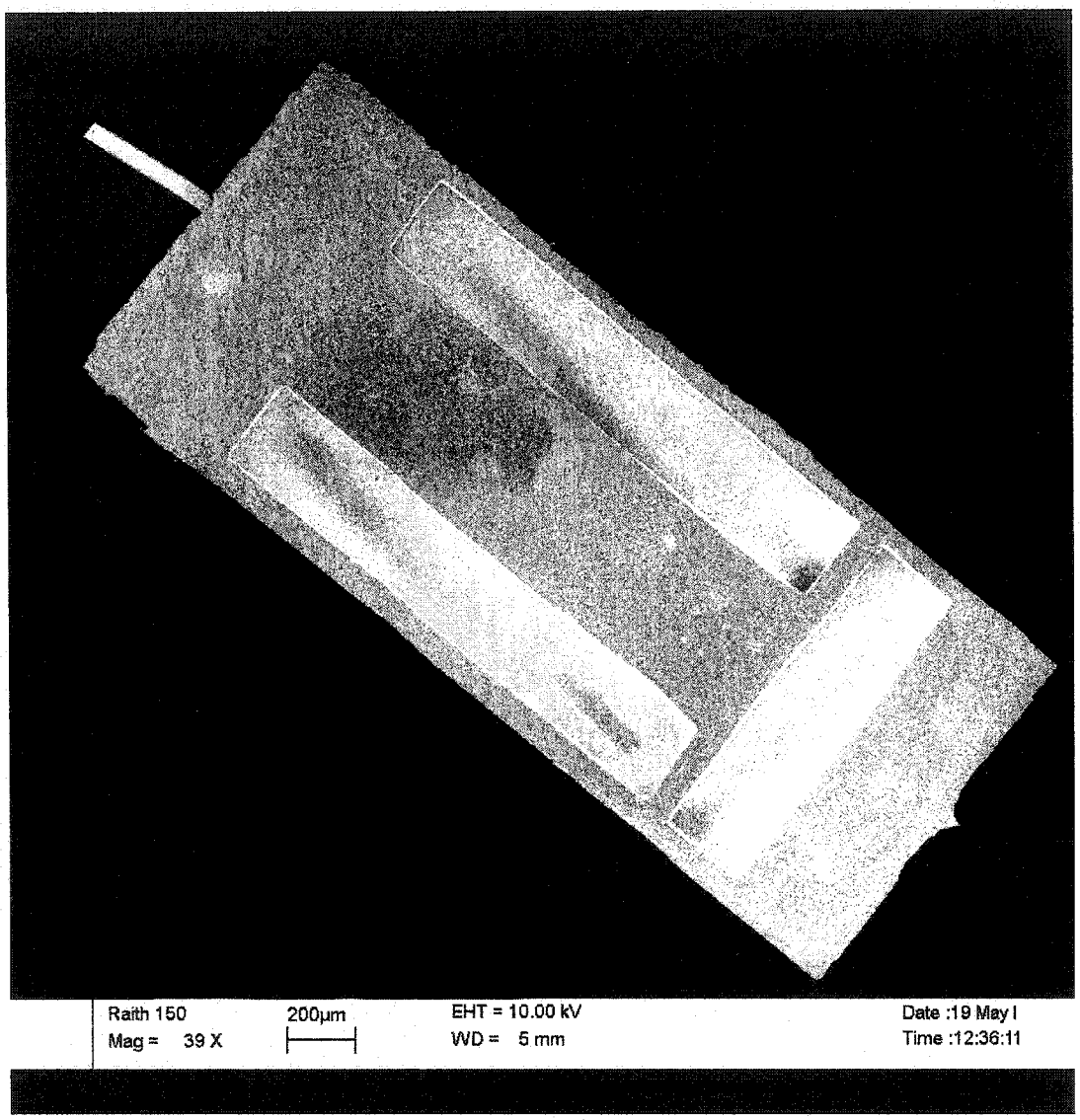


Figure 4.7: SEM photographs of the NSOM/AFM holder.

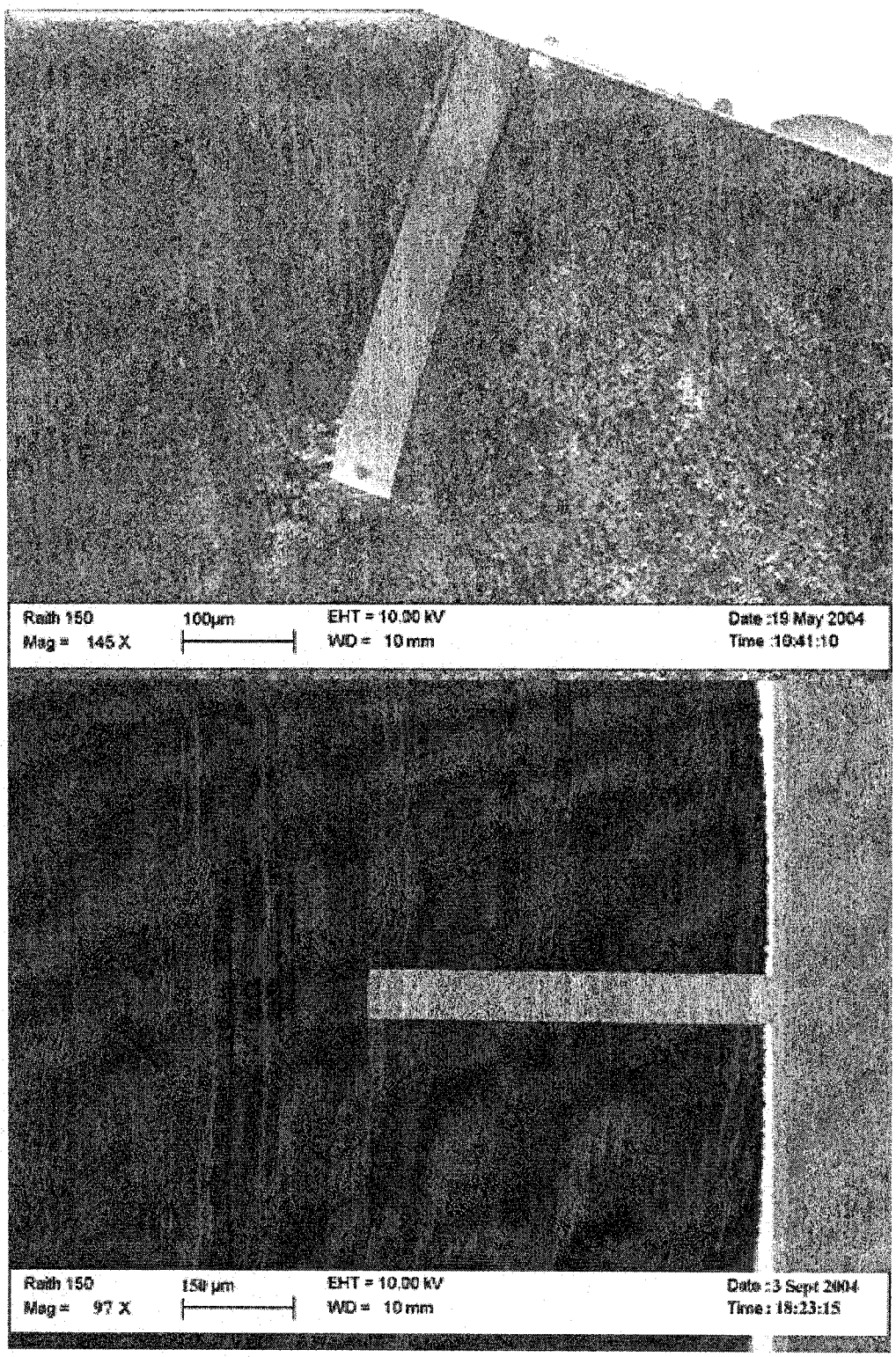


Figure 4.8: Top view, SEM images of the formed cantilevers exhibiting a 450 µm length and 60 µm width.

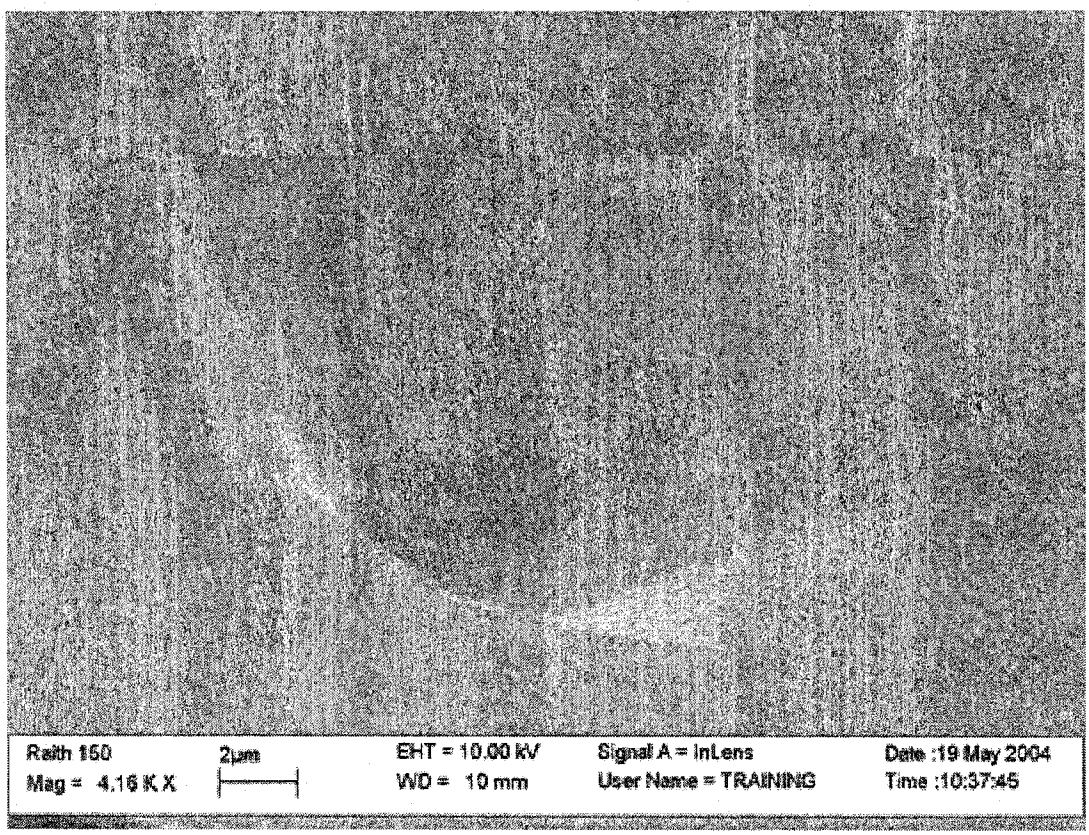


Figure 4.9: Top side of the cantilever and tip structure demonstrating the hollowed pyramidal tip which has been covered with a chromium/silver film.

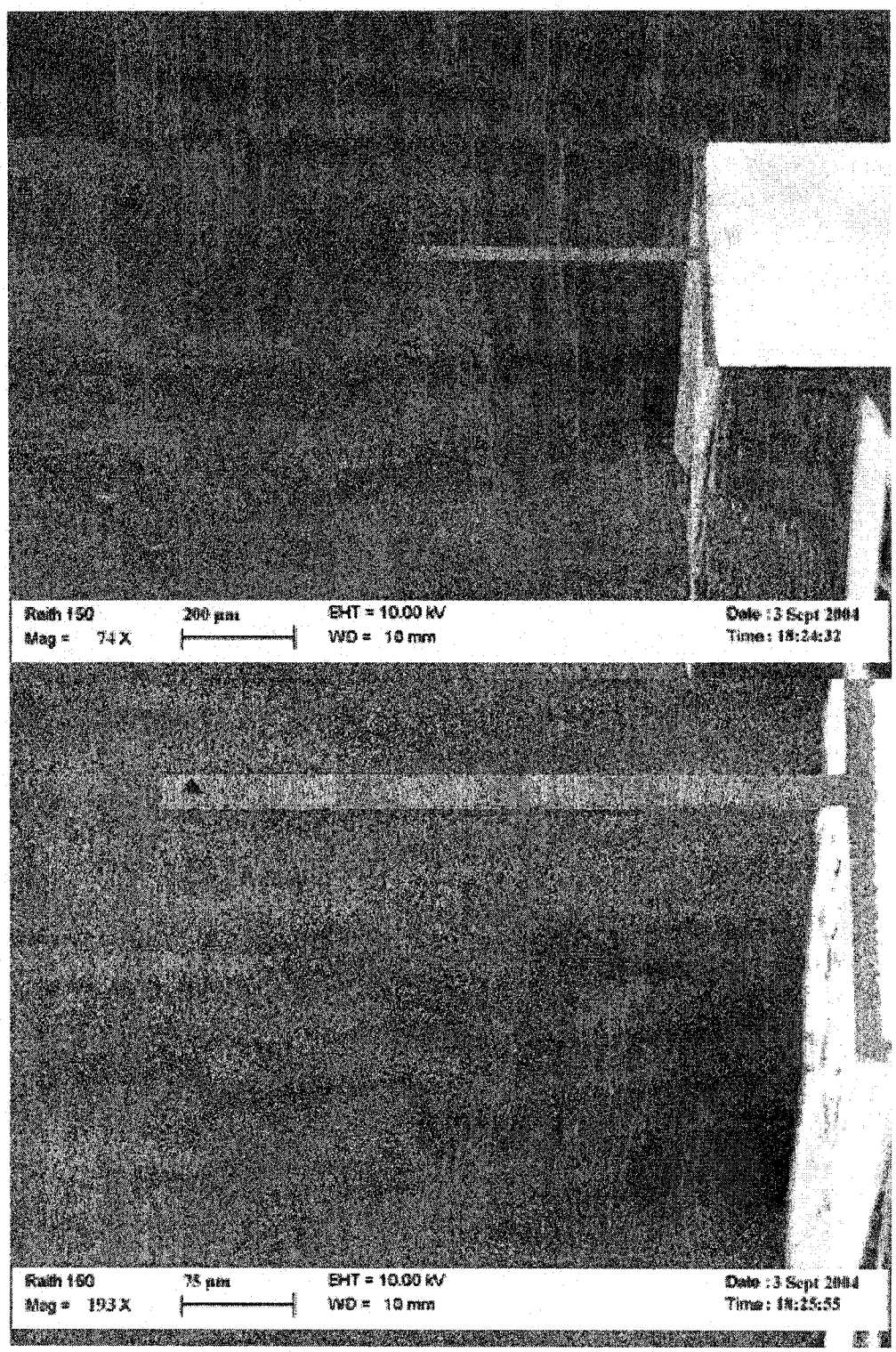


Figure 4.10: Cross section of the NSOM/AFM cantilever and holder demonstrating low stress due to the lack of curvature in the cantilever.

thermal oxidation, it is possible to selectively etch an arbitrarily small aperture at the apex of the tip using BOE [12]. An etch time of 2 minutes was used in this work resulting in apertures ranging from 200 – 400 nm in diameter. The completed tip, and a typical 200 nm aperture can be clearly seen in Figures 4.11, 4.12 and 4.13. Resolution in NSOM is mediated solely by the aperture diameter of the tip used and its distance from the sample. As apertures as small as 100 nm have been fabricated using this fabrication process, the theoretically obtainable resolution is 100 nm. Moreover, as the aperture diameter is regulated by the etching time in BOE, it will be possible to fabricate NSOM tips with apertures much smaller than 100 nm.

4.1.7 Silver-Chromium Bi-layer Deposition and Stress Compensation

The final step in the fabrication process is the deposition of the Cr/Ag bi-layer which is needed for light-confinement in the NSOM setup, as well as for stress compensation. Chromium is utilized due to its excellent mechanical properties and hardness; however in order to take advantage of surface plasmon effects, a thin silver layer is also needed as Chromium does not support SP coupling at optical wavelengths [13].

Stress compensation often plays an important role in any micromechanical device and it is especially important in cantilever based systems. Figure 4.14 depicts a cantilever without adequate stress compensation. Usually attributed to either the crystalline lattice or density mismatch between a deposited film and the underlying substrate, stress in cantilever devices can be either tensile or compressive depending upon the direction of curvature of the substrate after deposition. The relationship between stress, $\Delta\sigma$, and substrate curvature can be described by the Stoney equation [14],

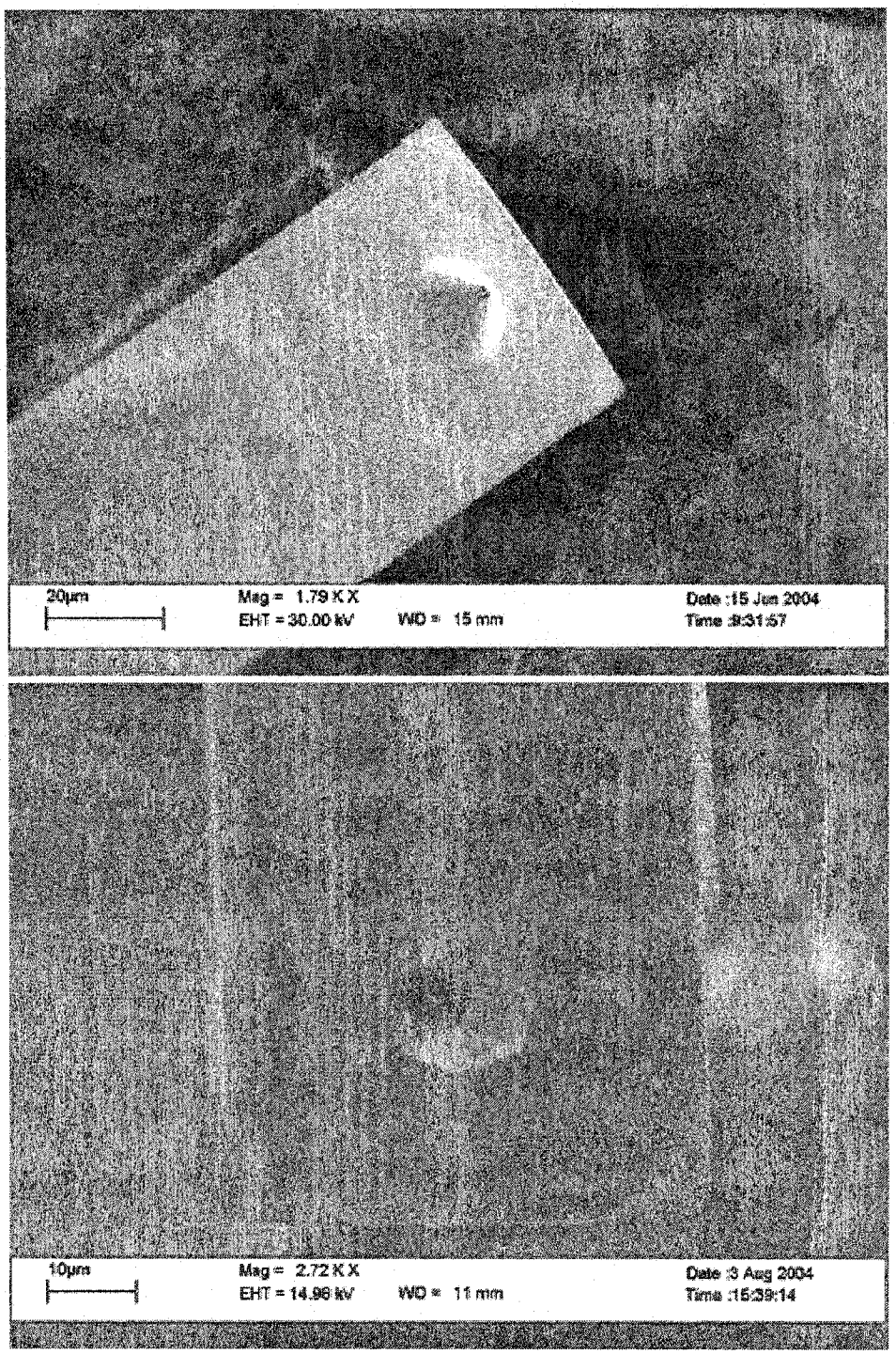


Figure 4.11: SEM images of the NSOM tips and cantilevers. In both images a small aperture can be resolved upon closer inspection.

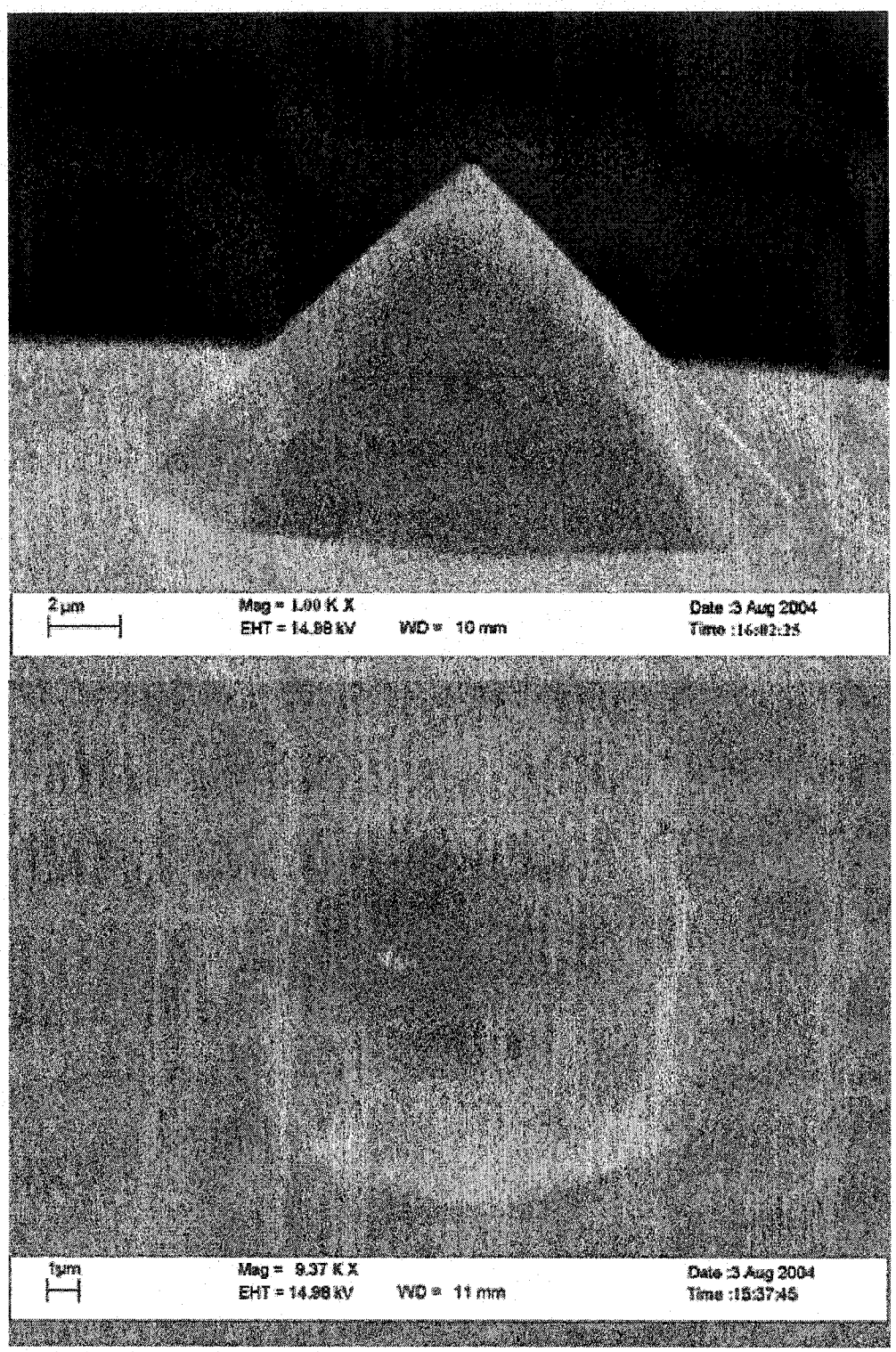


Figure 4.12: Close up images of an NSOM tip. Base widths of the tip were measured to be 20µm on average and tip heights were determined to be between 10 – 15 µm. The aperture diameter is clearly sub-micron.

Blank Page - No Text

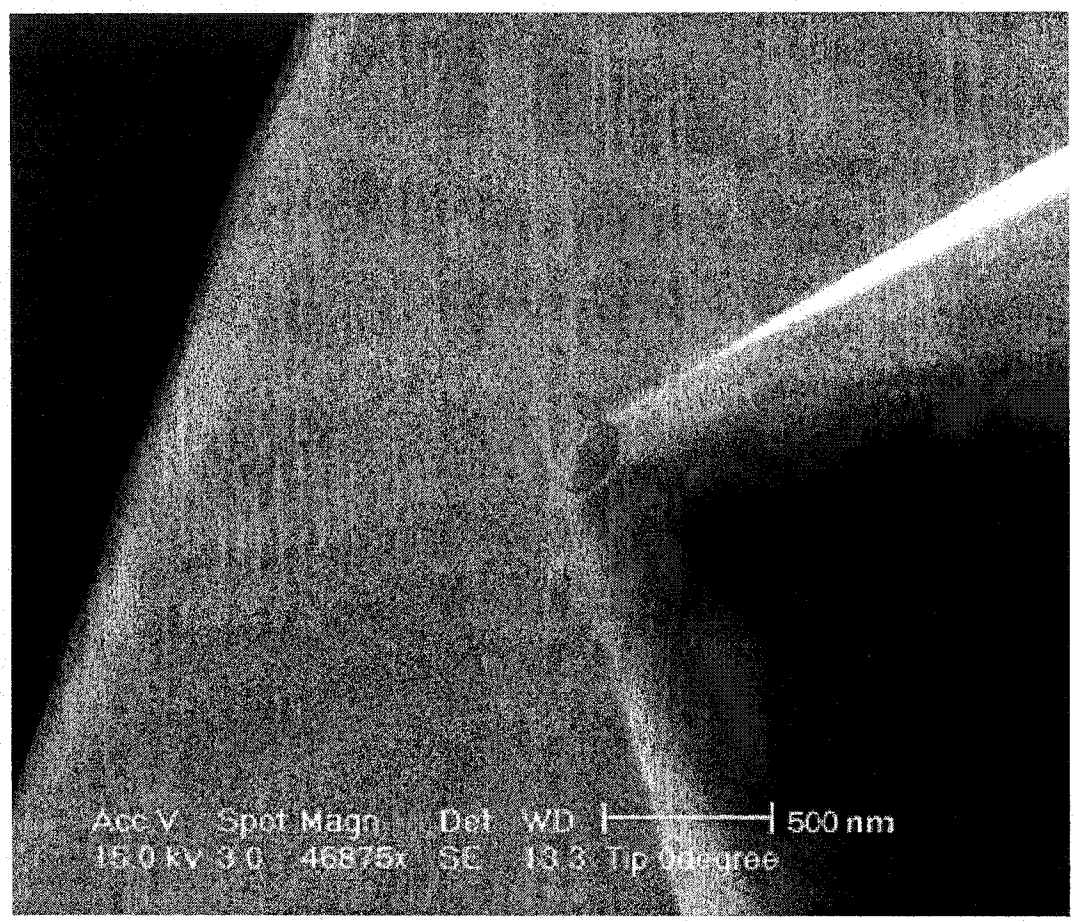


Figure 4.13: The actual aperture formed after the Cr/Ag deposition is revealed in this SEM image. The diameter is determined to be on the order of 200 nm, although aperture size can be tailored by varying the etching time in BOE prior to the Cr/Ag deposition.

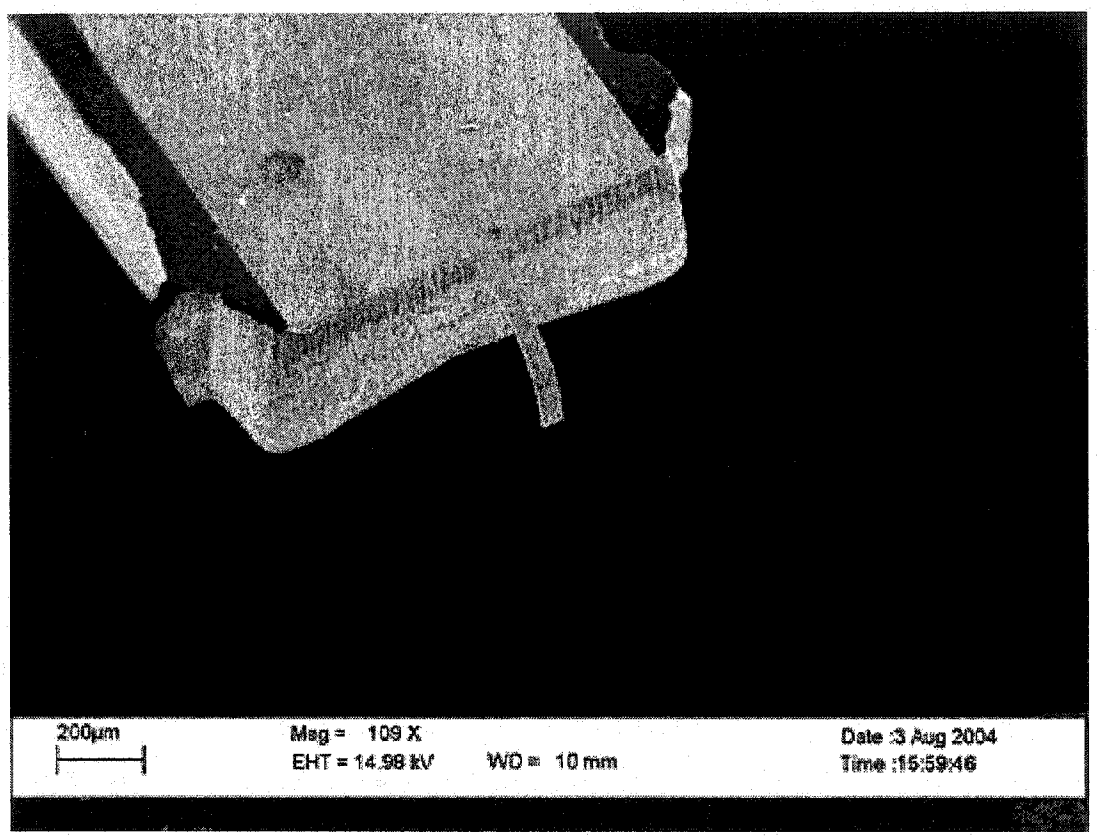


Figure 4.14: Image of a cantilever without adequate stress compensation. The cantilever is severely warped due to the compressive stress of the underlying oxide layer.

$$\Delta\sigma = \frac{\delta}{t}\frac{E_{film}}{(1-\nu_{film})}\frac{T^2}{3R^2}, \tag{4.13}$$

where E and ν are the Young's modulus and Poisson's ratio of the deposited material, T is the thickness of the substrate, t is the thickness of the film, R is the wafer radius and δ is the change in the deflection of the center of the substrate. In the cantilevered probes describe above, the compressive stress induced by the underlying silicon dioxide layer must be compensated for by the stresses associated with the chromium and silver thin films. Thermally wet-grown oxide has a Young's modulus and Poisson's ratio of 70 GPa and 0.17 respectively [15]. Sputtered chromium is usually under tensile stress ($-\delta$) and has a corresponding Young's modulus and Poisson's ratio of 185 GPa and 0.17 respectively [16]. However, silver films exhibit compressive stress with a Young's modulus of 83 GPa and a Poisson's ratio of 0.37 [17]. For zero cantilever stress, the sum of the stresses induced by all three deposited layers, $\Delta\sigma_{Ag} + \Delta\sigma_{Cr} + \Delta\sigma_{SiO_2}$, must equal zero. In order to compensate for the residual stress of the 500 nm oxide layer the necessary chromium film thickness can be determined from equation 4.13 using the parameters mentioned above. The silver film is assumed to have a constant thickness of 40 nm as SP coupling requires a relatively thin film. The desired chromium thickness is, therefore, 75 nm indicating a total metallic film thickness of 125 nm.

Chromium is first deposited into the hollow pyramidal tip using DC magnetron sputtering. The DC power is maintained at 300 W for the 450 second deposition duration. Silver is subsequently sputtered for 110 seconds at a DC power of 200 W. The predicted film thickness with these process parameters is 130 nm; however, the total metal film thickness after deposition was measured to be 127 nm using a contact profilometer. As

this value is much larger than the skin depth of light at optical wavelengths, no leakage is expected when the probe is used as an NSOM. After depositing the metal film the diameter of the aperture, d, is reduced according to

$$d = d_o - 0.5t \tag{4.12}$$

where d_o is the original aperture diameter and t is the thickness of the deposited layer [12]. A second method to tailor the aperture size is, therefore, available.

4.1.8 Fabrication Yield

Having only completed two iterations of the fabrication process, device yields for a single wafer are ~4%. Yield, in this case, refers to a completed cantilever with good mechanical properties still attached to the holder, a pyramidal tip between 10 and 15 μm tall, and a well defined aperture at the apex of the tip with a diameter between 200 and 400 nm. Increasing the yield will play a major role in the commercial viability of this near-field optical probe.

4.2 Summary

The design and fabrication process for a SPM probe enabling combined atomic force microscopic and near-field optical measurements has been described in detail. Microfabrication of the sensor is centered on a double-sided micromachining process incorporating several photolithographic, wet and dry anisotropic silicon etching steps. The cantilever and holder are formed from a single silicon wafer thereby maximizing the mechanical characteristics of the probe. The completed sensors are minimally stressed and demonstrate well-formed 200- 400 nm diameter apertures at the apex of the

pyramidal tip. This will eventually facilitate near-field optical and atomic force microscopic studies.

4.3 References

1. Betzig, E.I., J. and Lewis, A., *Collection mode near-field scanning optical microscopy,*. Appl. Phys. Lett., 1987. **51**: p. 2088.

2. Valle, P.J., Greffet, J. J., and Carminati, R., *Optical contrast, topographic contrast and artifacts in illumination-mode scanning near-field optical microscopy.* Journal of Applied Physics, 1999. **86**: p. 648.

3. Hecht, B., Bielefeldt, H., Inouye, Y., Pohl, D. W., and Novotny, L., *Facts and artifacts in near-field optical microscopy.* Journal of Applied Physics, 1997. **81**(6): p. 2492.

4. Mitsuoka, Y., et al., *Polarization properties of light emitted by a bent optical fiber probe and polarization contrast in scanning near-field optical microscopy.* Journal of Applied Physics, 1998. **83**(8): p. 3998.

5. Moers, M.H.P., et al., *Combined near field optical and force microscope.* Scanning Microscopy, 1993. **7**(3): p. 789.

6. Ruiter, A.G.T., et al., *Development of an integrated NSOM probe.* Ultramicroscopy, 1995. **61**(1-4): p. 139.

7. Cozma, A.a.P., B., *Characterization of the electrostatic bonding of silicon and Pyrex glass.* Journal of Micromechanics and Microengineering, 1995. **5**: p. 98.

8. Corning, *Glass Silicon Constraing Substrates*. 2003, Corning Inc.

9. Hull, R., *Properties of Crystalline Silicon.* EMIS Data Review Series. Vol. 20. 1999, London: INSPEC.

10. Mihalcea, C., et al., *Multipurpose sensor tips for scanning near-field microscopy.* Applied Physics Letters, 1996. **68**(25): p. 3531.

11. Dziomba, T., et al., *Ion beam-treated silicon probes operated in transmission and cross-polarized reflection mode near-infrared scanning near-field optical microscopy (NIR-SNOM).* Surface and Interface Analysis, 1999. **27**(5): p. 486.

12. Minh, P.N., T. Ono, and M. Esashi, *Microfabrication of miniature aperture at the apex of SiO sub 2 tip on silicon cantilever for near-field scanning optical microscopy.* Sensors and Actuators, A: Physical, 2000. **80**(2): p. 163.

13. Weeber, J.C., Krenn, J. R., Dereux, A., Lamprecht, B., Lacroute, Y., and Goudonnet, J. P., *Near-field observation of surface plasmon polariton propagation on thin metal stripes.* Physical Review B, 2001. **64**: p. 045411.

14. Stoney, G.J., *The Tension of Metallic Films Deposited by Electrolysis.* Proceedings of the Royal Society of London, 1909. **A82**: p. 172.

15. Kim, M.T., *Influence of substrates on the elastic reaction of films for the microindentation tests.* Thin Solid Films, 1996. **283**: p. 15.

16. Schneider, D.a.T., M. D., *Non-destructive characterization and evaluation of thin films by laser-induced ultrasonic surface waves.* Thin Solid Films, 1996. **290-291**: p. 305.

17. Rizzo, A., Tagliente, M. A., Alvisi, M. and Scaglione, S., *Structural and Optical Properties of Silver Thin Films Deposited by RF Magnetron Sputtering.* Thin Solid Films, 2001. **396**: p. 29.

Chapter 5: Conclusion

5.1 Conclusion

This thesis presents a study on the modeling and fabrication of near-field nano devices. The numerical modeling of two near-field optical devices, a nano-metallic array, and a hybrid solid immersion near-field probe, as well as the novel fabrication technique of a near-field optical probe are presented and described in detail. For the theoretical treatment, several custom additions are integrated into the basic finite-different difference time-domain algorithm to provide a more comprehensive description of the performance of the above-mentioned devices. The model represents a full numerical solution to Maxwell's equations with the ability to model dielectric and metallic materials alike as both of these factors play an important role in describing the propagation and interaction of light with structures at the nanometer scale.

The model is used to simulate the interaction of ultrashort, pulsed light with three nano-metallic slit arrays having periodicities of 660, 750 and 870 nm. The arrays are found to have greater than 90% transmission efficiency for wavelengths of 1100, 1050 and 800 nm for the 660, 750 and 870 nm respectively. Similar to previously reported results, the arrays exhibit pulse broadening characteristics as well as delayed temporal transmission when pulses in excess of 100 fs are employed; however, the arrays are also found to have unique pulse-shaping characteristics when pulses with durations on the order of the SP (~30 fs) lifetime are utilized. In this case, a train of pulses is reradiated from the array following an initial high-amplitude pulse. The high amplitude pulse is attributed to the coherent SP transmission of the incident pulse, whereas the subsequent pulse train is attributed to the collective re-radiation of the SP as it propagates along the

array. This passive pulse shaping system could have potential uses in a variety of applications as it simultaneously allows high transmission and pulse reshaping.

The second device modeled using the FDTD software suite entails a pair of hybrid solid immersion near-field optical probes. Two probe designs are studied revealing a significant improvement in the transmission properties when compared to conventional cantilevered tips. The enhanced transmission of the high-index layer design is largely attributed to the existence of a high index material layer that essentially decreases the wavelength of the light passing through the cone shaped tip; whereas the solid immersion lens design leads to further increases in transmission not only because of the high index medium, but also because of the focusing properties of the embedded spherical lens. Moreover, surface plasmon effects are also taken into account when computing the transmitted intensities, and play a significant role in the observed results. The physical realization of these probes will certainly help in ushering in the use of cantilevered near-field optical probes as their fiber-based counterparts continue to suffer from the drawbacks of high fragility and low optical throughput.

Following the presentation of a detailed discussion of near-field optics and the model mentioned above, the design and fabrication of a novel monolithic hybrid AFM/NSOM probe is presented. The fabrication procedure consists of a double sided micromachining process capable of mass-producing >100 probes per single wafer. In contrast to previously reported fabrication results [1-5], the newly developed process does not rely on anodic bonding between several wafers, nor does it heavily rely on the time-consuming wet anisotropic etching processes for the holder formation. Consisting of 5 lithographic steps, the fabrication procedure entails: formation of a pyramidal silicon

tip; definition of the cantilever structure; a back-side via and alignment groove etch; and finally the holder formation and cantilever release. As AFM sensors require a reflective coating, and NSOM sensors require a metallic layer for light confinement, the process is completed by sputtering a dual-layer film of chromium and silver. The chromium is necessary for its mechanical properties, whereas the silver is present due to its capacity to support SP waves.

SEM images taken during the fabrication process have initially revealed pyramidal tips with sub-100 nm diameters. Following the cantilever release, images were taken that portray well defined cantilevers 450 μm in length, 60 μm in width and between 2 and 5 μm in thickness in accordance with specifications. Finally, upon closer inspection, tip apertures ranging from 200 – 400 nm in diameter are demonstrated thereby allowing for high-resolution combined near-field/atomic force microscopy.

5.2 Future Directions

The production of the near-field optical probe outlined in this thesis forms the basis for an experimental framework that will lead to many new studies in the nano-scale regime. In terms of the actual probe production, however, the key components to any near-field optical microscope are resolution and optical throughput. The latter is well addressed in this thesis as improvements to optical throughput can be garnered through the use of surface plasmons, high index thin-film layers, and solid immersion lens systems. These additions to the near-field probe outlined herein will likely foster any future experiments.

Moreover, resolution in near-field optical microscopy is correlated directly to aperture diameter. As previously mentioned, the aperture diameter of the near-field probe

described is tailorable to the application at hand. By varying the silicon dioxide etch time, or the metallic layer thickness it should be possible to achieve aperture diameters below 50 nm. At this resolution near-field optical studies of single molecules are indeed possible.

Finally, the ability to integrate the newly fabricated near-field optical probe into existing atomic force microscope systems is extremely convenient as it will provide a direct competitor to the commonly employed fiber based systems. Furthermore, due to the dispersion and pulse broadening associated with pulse propagation through a fiber, the cantilevered near-field probe will facilitate the use of ultrashort pulsed systems in near-field microscopy. Exploring this avenue will provide an entirely new field of research encompassing the realm of femtosecond time-resolved studies in the nanometer regime.

5.3 References

1. Akamine, S., H. Kuwano, and H. Yamada, *Scanning near-field optical microscope using an atomic force microscope cantilever with integrated photodiode.* Applied Physics Letters, 1996. **68**(5): p. 579.

2. Minh, P.N., T. Ono, and M. Esashi, *Microfabrication of miniature aperture at the apex of SiO sub 2 tip on silicon cantilever for near-field scanning optical microscopy.* Sensors and Actuators, A: Physical, 2000. **80**(2): p. 163.

3. Jung, M.Y., S.S. Choi, and I.W. Lyo, *Micromachined Si sub 3N sub 4-tip on cantilever for parallel SFM and NSOM applications.* Microelectronic Engineering, 1999. **46**(1): p. 427.

4. Oesterschulze, E., et al., *Cantilever probes for SNOM applications with single and double aperture tips.* Ultramicroscopy, 1998. **71**(1-4): p. 85.

5. Schurmann, G., et al., *Microfabrication of a combined AFM-SNOM sensor.* Ultramicroscopy, 2000. **82**(1): p. 33.

Appendix A: FDTD Source Code

The source code for the FDTD software suite was written in an object oriented style in C++. The following is a complete listing of the source files for the software developed with classes listed in alphabetical order.

```
////////////////////////////////////////////////////////////////////////////////
//BitmapFieldWriter.cpp – used to output field data to graphical bitmap files
////////////////////////////////////////////////////////////////////////////////
#include "BitmapFieldWriter.h"
#include "stdheader.h"
#include "Bitmap.h"
#include "Options.h"
#include <stdio.h>
#include <math.h>
#include <iostream.h>

int BitmapFieldWriter::write( char*  filename, double** field, int lengthI, int lengthJ, Options* options, bool scaleH )
{
 int rc = SUCCESS;
 int i,j;
 char textfilename[512];
 char bmpfilename[512];
 int mode = options->getOutputMode();
 double scale = !scaleH ? 1.0 : options->getMaxOutputField()*sqrt( EPSILON_NOT * INVERSE_MU_NOT);


 sprintf(textfilename,"%s.txt",filename);
 sprintf(bmpfilename,"%s.bmp",filename);

 BitmapInfoHeader bih;
 BitmapFileHeader bfh;

 bfh.bfOffBits = sizeof(bfh) + sizeof(bih);
 bfh.bfReserved1 = 0;
 bfh.bfReserved2 = 0;
 bfh.bfType = 19778;
 bfh.bfSize = bfh.bfOffBits + (lengthI*lengthJ)*3;

 bih.biSize = sizeof(bih);
 bih.biBitCount=24;
 bih.biHeight=lengthJ;
 bih.biWidth=lengthI;
 bih.biCompression=0;
 bih.biPlanes=1;
 bih.biXPelsPerMeter=3780;
 bih.biYPelsPerMeter=3780;
 bih.biSizeImage = (lengthJ*lengthI)*3;
 bih.biClrUsed = 0;
 bih.biClrImportant =0;
 FILE *bfp = NULL;
 FILE *fp = NULL;

 if( mode == BMP_OUTPUT_MODE ||
  mode == BMP_TXT_OUTPUT_MODE )
 {
  bfp = fopen( bmpfilename, "wb" );
  if( bfp == NULL)
  {
   rc = INVALID_FILENAME;
   return rc;
  }
 }


 if( mode == TXT_OUTPUT_MODE ||
  mode == BMP_TXT_OUTPUT_MODE )
 {
  fp = fopen( textfilename, "wt" );
  if( fp == NULL )
```

```
 {
 rc = INVALID_FILENAME;
 return rc;
 }
}

if( mode == BMP_OUTPUT_MODE ||
 mode == BMP_TXT_OUTPUT_MODE )
{
#ifdef ITANIUM
        fwrite(&bfh.bfType,sizeof(WORD),1,bfp);
        fwrite(&bfh.bfSize,sizeof(DWORD),1,bfp);
        fwrite(&bfh.bfReserved1,sizeof(WORD),1,bfp);
        fwrite(&bfh.bfReserved2,sizeof(WORD),1,bfp);
  fwrite(&bfh.bfOffBits,sizeof(DWORD),1,bfp);
#else
  fwrite(&bfh,sizeof(bfh),1,bfp);
#endif
  fwrite(&bih,sizeof(bih),1,bfp);
}

double min = options->getMinOutputField();
double max = options->getMaxOutputField();

for( i=0;i<lengthJ;i++)
{
 for( j=0;j<lengthI;j++ )
 {
  if( mode == BMP_OUTPUT_MODE ||
   mode == BMP_TXT_OUTPUT_MODE )
   {
   /*unsigned char bgr[3];
   int colorRange = (5*256 + 256/2);

   double percent = ( fabs(min) + field[j][i] )/( fabs(max) + fabs(min) );
   int color = percent*colorRange;

   bgr[0] = color < 2*256 ? 0 : ( color < 3*256 ? color - 2*256 : (color < 5*256 ? 255 : color-5*256 ) );
   bgr[1] = color < 256 ? color : ( color < 3*256 ? 255 : ( color < 4*256 ? color-3*256 : 0 ) );
   bgr[2] =color < 256 ? 255 : ( color < 2*256 ? color-256 : ( color > 5*256 ? color-5*256 : 0 ) );


   bgr[0] = color < 256 ? color : ( color < 2*256 ? 255     : ( color < 3*256 ? 255   : ( color < 4*256 ? color-3*256 : ( color < 5*256 ?
0   : color-5*256 ) ) ) );
   bgr[1] = color < 256 ? 0  : ( color < 2*256 ? color - 256 : ( color < 3*256 ? 255   : ( color < 4*256 ? 255   : ( color < 5*256 ? color -
4*256 : 0 ) ) ) );
   bgr[2] = color < 256 ? 0  : ( color < 2*256 ? 0    : ( color < 3*256 ? color-2*256 : ( color < 4*256 ? 255   : ( color < 5*256 ? 255   :
255 ) ) ) );
   */

   unsigned char bgr[3];
   int colorRange = (5*256);
   double percent = ( fabs(min) + field[j][i]/scale )/( fabs(max) + fabs(min) );
   int color = percent*colorRange;

   bgr[0] = color <= 128 ? 255     : ( color <= 3*256/2 ? 255          : ( color <= 5*256/2 ? 5*256/2 - color : ( color < 7*256/2 ? 0    :
( color < 9*256/2 ? 0      : color-9*256/2 ) ) ) );
   bgr[1] = color <= 128 ? 255     : ( color <= 3*256/2 ? 3*256/2 - color  : ( color <= 5*256/2 ? 0       : ( color < 7*256/2 ? 0    : (
color < 9*256/2 ? color - 7*256/2 : 255 ) ) ) );
   bgr[2] = color <= 128 ? 128-color : ( color <= 3*256/2 ? 0        : ( color <= 5*256/2 ? 0       : ( color < 7*256/2 ? color-5*256/2  :
( color < 9*256/2 ? 255     : 255 ) ) ) );

   fwrite(bgr,sizeof(unsigned char),3,bfp);
   }

  if( mode == TXT_OUTPUT_MODE ||
   mode == BMP_TXT_OUTPUT_MODE )
   {
   fprintf(fp," %e",field[j][i]);
   }
```

```
 }

 if( j % 4 != 0 && ( mode == BMP_OUTPUT_MODE ||
  mode == BMP_TXT_OUTPUT_MODE ) )
  {
  unsigned char space = 0;
  int n = 0;
  for( n=0; n<j%4 ; n ++ )
   fwrite(&space,sizeof(unsigned char),1,bfp);
  }

 if( mode == TXT_OUTPUT_MODE ||
  mode == BMP_TXT_OUTPUT_MODE)
  {
   fprintf(fp,"\n");
  }
 }

 if( fp != NULL )
  fclose(fp);
 if( bfp != NULL )
  fclose(bfp);
 return rc;
}

////////////////////////////////////////////////////////////////////////////////////////////////
//BitmapGeometryReader.cpp -- used to reading geometry files stored as bitmaps
////////////////////////////////////////////////////////////////////////////////////////////////
#include "BitmapGeometryReader.h"
#include "PerfectlyMatchedLayerBoundaryCondition.h"
#include "GaussianSource.h"
#include "Bitmap.h"
#include <cstdio>
#include <iostream.h>
#include "Geometry.h"
#include <assert.h>

/** Reads in the geometry from a 24 bit bitmap file
 *  Returns 0 on success, >0 on failure.
 */
int BitmapGeometryReader::readGeometry( char *geomFile, Geometry* geom)
{
 FILE* geomFP;
 BitmapFileHeader bfh;
 BitmapInfoHeader bih;
 int bytesRead = 0;
 BYTE* bytesReversed = NULL;
 long imageSize;
 int i,j,n;
 int rc = SUCCESS;
 geometry = geom;
 PerfectlyMatchedLayerBoundaryCondition *bc =
  PerfectlyMatchedLayerBoundaryCondition::getBoundaryCondition();

 //make sure we have a boundary condition!!
 assert( bc != NULL );

 cerr << "Size of BYTE, WORD, DWORD, LONG are :"
     << sizeof(BYTE) << " "
     << sizeof(WORD) << " "
     << sizeof(DWORD) << " "
     << sizeof(LONG) << "\n";

 cerr << "Size of BitmapFileHeader is: " << sizeof(BitmapFileHeader) << "\n";
 cerr << "Size of BitmapInfoHeader is: " << sizeof(BitmapInfoHeader) << "\n";

 //Open the geometry files for reading
 geomFP = fopen( geomFile, "rb" );
 if( geomFP == NULL )
```

```
{
 cerr << "Geometry file not found! \n";
 return FILE_NOT_FOUND;
}

//Read in the geometry from the bitmap file
bytesRead = fread(&bfh.bfType,1,sizeof(WORD),geomFP );
bytesRead += fread(&bfh.bfSize,1,sizeof(DWORD),geomFP );
bytesRead += fread(&bfh.bfReserved1,1,sizeof(WORD),geomFP );
bytesRead += fread(&bfh.bfReserved2,1,sizeof(WORD),geomFP );
bytesRead += fread(&bfh.bfOffBits,1,sizeof(DWORD),geomFP );

#ifdef ITANIUM
 bytesRead += bytesRead%4 ;
#endif

 if( bytesRead != sizeof(BitmapFileHeader) )
 {
  //cerr << "Bytes Read: " << bytesRead << "\n";
  //cerr << "Size of header: " << sizeof(BitmapFileHeader) << "\n";
  cerr << "Incorrect geometry file header format!\n";
  fclose( geomFP );
  return INVALID_FILE_FORMAT;
 }

 bytesRead = fread(&bih,1,sizeof(BitmapInfoHeader),geomFP );
 if( bytesRead != sizeof(BitmapInfoHeader ) )
 {
  cerr << "Incorrect geometry info header format!\n";
  fclose( geomFP );
  return INVALID_FILE_FORMAT;
 }

 //Output bitmap statistics
 cerr << "Bit depth: " << bih.biBitCount << "\n";
 cerr << "Image width: " << bih.biWidth << " pixels\n";
 cerr << "Image height: " << bih.biHeight << " pixels\n";

 //must be using a 24 bit bitmap
 if( bih.biBitCount != BIT_DEPTH )
 {
  cerr << "Incorrect geometry format. Bit detph != 24...\n";
  fclose( geomFP );
  return INVALID_FILE_FORMAT;
 }

 //read in the bytes from the file
 geometry->m_iWidth = bih.biWidth;
 geometry->m_iHeight = bih.biHeight;
 imageSize = BIT_DEPTH*geometry->m_iHeight*geometry->m_iWidth / BITS_PER_BYTE + (geometry->m_iWidth*3) %
4*geometry->m_iHeight;

 bytesReversed = new BYTE[imageSize];
 bytesRead = fread(bytesReversed,sizeof(BYTE),imageSize,geomFP);
 if( bytesRead != imageSize )
 {
  cerr << "Error reading file information! \n";
  fclose( geomFP );
  return INVALID_FILE_FORMAT;
 }

 //populate the geometry points
 geometry->points = new GeometryPoint*[geometry->m_iHeight];
 n=0;
 for(j=0;j<geometry->m_iHeight&&rc==SUCCESS;j++)
 {
  geometry->points[ j] = new GeometryPoint[geometry->m_iWidth];
  for(i=0;i<geometry->m_iWidth&&rc==SUCCESS;i++)
  {
   long index = Material::getIndexForColor(bytesReversed[n],
```

```
        bytesReversed[n+1],
        bytesReversed[n+2]);
 n+=3;

 if( j < bc->getWidth() ||
  j >= geometry->m_iHeight - bc->getWidth() ||
  i < bc->getWidth() ||
  i >= geometry->m_iWidth - bc->getWidth() )
 {
  geometry->points[j][i].setMaterial( bc );
 }
 else
 {
 Material* mat = Material::materialExists( index );
  if( mat != NULL )
  {
  geometry->points[j][i].setMaterial( mat );
  if( mat->isSource() )
  {
   // do something here!!!!
   Source* source = (Source*) mat;
   source->setPoint( i, j );
  }
  }
  else
  {
   cerr << "Error: Material for color bgr " << (unsigned int)bytesReversed[n] << " " << (unsigned int)bytesReversed[n+1] << " " <<
(unsigned int)bytesReversed[n+2] << " was null at position x=" << i << " y=" << j << "\n";

   rc = INVALID_FILE_FORMAT;
   break;
  }
 }
 }
 n+=(n%4);
}

delete [] bytesReversed;
fclose( geomFP );
return rc;
}

////////////////////////////////////////////////////////////////////////////////////////////////
//FDTDSimulation.cpp – main control loop for running the software
////////////////////////////////////////////////////////////////////////////////////////////////
#include "Geometry.h"
#include "Options.h"
#include "BitmapGeometryReader.h"
#include "PropertyReader.h"
#include "FiniteDifferenceTimeDomainSolver.h"
#include "SecondOrderSolver.h"
#include "PulsedSecondOrderSolver.h"
#include "GaussianSource.h"
#include <time.h>
#include <math.h>
#include <stdio.h>
#include <iostream.h>

int main( char* argc, char* argv[] )
{
 int rc;

 time_t startTime;
 time_t endTime;
 time_t runTime;
 startTime = time (NULL);
 Geometry *g = new Geometry();
 Options *options = new Options();

 PropertyReader *pr = new PropertyReader();
```

```
rc = pr->readProperties( "properties.prop", g, options );
if( rc != 0 )
{
 cout << "Error reading in properties! Exiting...\n";
 delete pr;
 delete g;
 Material::deleteMaterials();
 return 1;
}
else
 delete pr;

BitmapGeometryReader *bgr = new BitmapGeometryReader();
rc = bgr->readGeometry( "geometry.bmp", g );
if( rc != 0 )
{
 cout << "Error reading in geometry! Exiting...\n";
 delete bgr;
 delete g;
 Material::deleteMaterials();
 return 1;
}
else
 delete bgr;

FiniteDifferenceTimeDomainSolver *sos = NULL;


    sos = new PulsedSecondOrderSolver( g, options );
    if( sos->initialize() != SUCCESS )
    {
     delete sos;
     delete g;
     delete options;
     return rc;
    }

sos->solve();

endTime = time (NULL);
runTime = endTime - startTime;
FILE* fp = fopen("RunningTime.tim","wt");
if( fp != NULL )
{
 long hrs = (long)(runTime/3600);
 long minutes= (long)((runTime - hrs*3600)/60);
 long seconds= (runTime - hrs*3600 - minutes*60);
 cerr << "Run time: " << hrs << " hrs " << minutes << " minutes " << seconds << " seconds\n";
 fprintf(fp, "Run time for %s: %d hrs %d minutes %d seconds\n",options->getSimulationName(),hrs,minutes,seconds);
 fclose(fp);
}
delete sos;
delete g;
delete options;
Material::deleteMaterials();
return SUCCESS;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//FieldIntegrator.cpp -- Used to integrate the given field data over a set period of time
////////////////////////////////////////////////////////////////////////////////////////////////////
#include "FieldIntegrator.h"
#include "stdheader.h"
#include "Bitmap.h"
#include "Options.h"
#include "Geometry.h"
#include <stdio.h>
#include <math.h>
#include <iostream.h>
```

```
FieldIntegrator::FieldIntegrator( Geometry* geom, Options* opt )
{
 int i;
 options = opt;
 geometry = geom;
 m_DFieldX = new double[options->getXIntegrationLineCount()];
 m_DFieldY = new double[options->getYIntegrationLineCount()];
 if( options->getSimulationMode() == TM_MODE )
 {
 sprintf(m_sFilename, "%s_%s.int", options->getSimulationName(),TM_MODE_TOKEN );
 }
 else if( options->getSimulationMode() == TE_MODE )
 {
 sprintf(m_sFilename, "%s_%s.int", options->getSimulationName(),TE_MODE_TOKEN );
 }
 for(i=0;i<options->getXIntegrationLineCount();i++)
  m_DFieldX[i] = 0;
 for(i=0;i<options->getYIntegrationLineCount();i++)
  m_DFieldY[i] = 0;

}

int FieldIntegrator::reset( int oldTime )
{
 int rc = SUCCESS;
 return rc;
}

int FieldIntegrator::addPoint(  double** field, int timeStep )
{
 int rc = SUCCESS;
 int x,y;
 int i;
 for( i=0;i<options->getXIntegrationLineCount();i++)
 {
  x = options->getXIntegrationLine(i);
  if( x >= geometry->getGridWidth() )
   m_DFieldX[i] += 0;
  else if( timeStep >= geometry->getGridTim e() )
   m_DFieldX[i] += 0;
  else
   for( y=0; y<geometry->getGridHeight(); y++ )
    m_DFieldX[i] += field[ x ][ y ]*field[ x ][ y ];
 }
 for( i=0;i<options->getYIntegrationLineCount();i++)
 {
  y = options->getYIntegrationLine(i);
  if( y >= geometry->getGridWidth() )
   m_DFieldY[i] += 0;
  else if( timeStep >= geometry->getGridTime() )
   m_DFieldY[i] += 0;
  else
   for( x=0; x<geometry->getGridWidth(); x++ )
    m_DFieldY[i] += field[ x ][ y ]*field[ x ][ y ];
 }
 return rc;
}

FieldIntegrator::~FieldIntegrator()
{
 delete [] m_DFieldX;
 delete [] m_DFieldY;
}

int FieldIntegrator::write()
{
 int rc = SUCCESS;
 int i,j;

 FILE *fp = NULL;
```

```
fp = fopen( m_sFilename, "wt" );
if( fp == NULL )
{
 rc = INVALID_FILENAME;
 return rc;
}

for(i=0;i<options->getXIntegrationLineCount();i++)
{
 fprintf(fp,"For Line X=%d Field Integration=%le\n",options->getXIntegrationLine(i),m_DFieldX[i] );
}
fprintf(fp,"\n");
for(i=0;i<options->getYIntegrationLineCount();i++)
{
 fprintf(fp,"For Line Y=%d Field Integration=%le\n",options->getYIntegrationLine(i)
  ,m_DFieldY[i] );
}


if( fp != NULL )
 fclose(fp);
return rc;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//FiniteDifferenceTimeDomainSolver.cpp – superclass for the solver algorithm
////////////////////////////////////////////////////////////////////////////////////////////////////
#include "FiniteDifferenceTimeDomainSolver.h"
#include "Geometry.h"
#include "Options.h"
#include "Material.h"
#include <iostream.h>

FiniteDifferenceTimeDomainSolver::FiniteDifferenceTimeDomainSolver( Geometry *geom, Options* opt )
{
 cout << "Creating: FiniteDifferenceTimeDomainSolver\n";
 geometry = geom;
 options = opt;
}

FiniteDifferenceTimeDomainSolver::~FiniteDifferenceTimeDomainSolver()
{
 cout << "Deleting: FiniteDifferenceTimeDomainSolver\n";
 geometry = NULL;
}

int FiniteDifferenceTimeDomainSolver::initialize()
{
 int rc = Material::initializeMaterials( geometry );
 cerr << "Message: Initializing finite difference time domain solver\n";
 return rc;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//GaussianSource.cpp – used to instatiate a CW gaussian source
////////////////////////////////////////////////////////////////////////////////////////////////////
#include "GaussianSource.h"
#include <iostream.h>
#include <math.h>
#include "stdheader.h"
#include "Geometry.h"
#include <stdio .h>

GaussianSource::GaussianSource( long index, char *name, Material *lastMaterial,
      double maxEField, int startTime, double lambda,
      double FWHM )
:Source( index, name, lastMaterial, maxEField, startTime, lambda )
{
 m_dFullWidthHalfMax = FWHM;
```

```
m_dSigma = FWHM / ( 2 * sqrt( 2 * log(2) ) );
ramp = 100;
cerr << "Message: Creating gaussian source with lambda= "<< lambda<< "\n";
}

//TE functions

double GaussianSource::getElectricFieldZx( int i, int n )
{
//if we're a vertical source then all Z Electric field components
//in the x direction should not exist
if( m_iMinX == m_iMaxX )
 return 0;

double offset = m_dDeltaSpace * sqrt( ( i - m_iHalfX ) * ( i - m_iHalfX ) );

double EField =
 m_dMaxElectricField*exp(-0.5*offset*offset/m_ dSigma/m_dSigma);
double time =  (n-m_iStartTime) >= 0 ? ((double)(n-m_iStartTime))*m_dDeltaTime : 0;
double Escale = (n-m_iStartTime) >= ramp ? 1.0 : exp(10.0*(n-m_iStartTime-ramp)/ramp );

EField*= Escale*sin( m_dAngularFrequency*time -
m_dAngularFrequency*INVERSE_LIGHT_SPEED*getRefractiveIndex()*0.5*m_dDeltaSpace);

return EField;
}

double GaussianSource::getElectricFieldZy( int j, int n )
{
//if we're a horizontal source then all Z Electric field components
//in the Y direction should not exist
if( m_iMinY == m_iMaxY )
 return 0;

double offset = m_dDeltaSpace * sqrt( ( j - m_iHalfY ) * ( j - m_iHalfY ) );

double EField =
 m_dMaxElectricField*exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double time =  (n-m_iStartTime) >= 0 ? ((double)(n-m_iStartTime))*m_dDeltaTime : 0;
double Escale = (n-m_iStartTime) >= ramp ? 1.0 : exp(10.0*(n-m_iStartTime-ramp)/ramp );

EField*= Escale*sin( m_dAngularFrequency*time -
m_dAngularFrequency*INVERSE_LIGHT_SPEED*getRefractiveIndex()*0.5*m_dDeltaSpace );

return EField;
}


double GaussianSource::getMagneticFieldX( int i, int n )
{
//if we're a vertical source then all X Magnetic field components
//should not exist
if( m_iMinX == m_iMaxX )
 return 0;

double offset = m_dDeltaSpace * sqrt( ( i - m_iHalfX ) * ( i - m_iHalfX ) );
double HField =
 m_dMaxMagneticField*exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double Hscale = (n-m_iStartTime) >= ramp ? 1.0 : exp(10.0*(n-m_iStartTime-ramp)/ramp );

double time =  (n-m_iStartTime) > 0 ? (0.5 + (double)(n-m_iStartTime))*m_dDeltaTime : 0;

HField*= Hscale*sin( m_dAngularFrequency*time );

return HField;
}

double GaussianSource::getMagneticFieldY( int j, int n )
```

```
{
//if we're a horizontal source then all Y Magnetic field components
//should not exist
if( m_iMinY == m_iMaxY )
 return 0;

double offset = m_dDeltaSpace * sqrt( ( j - m_iHalfY ) * ( j - m_iHalfY ) );
double HField =
 m_dMaxMagneticField*exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double Hscale = (n-m_iStartTime) >= ramp ? 1.0 : exp(10.0*(n-m_iStartTime-ramp)/ramp );

double time =  (n-m_iStartTime) > 0 ? (0.5 + (double)(n-m_iStartTime))*m_dDeltaTime : 0;

HField*= Hscale*sin( m_dAngularFrequency*time );

return HField;
}


//TM functions


double GaussianSource::getMagneticFieldZx( int i, int n )
{
 //if we're a vertical source then all Z Magnetic field components
 //in the x direction should not exist
 if( m_iMinX == m_iMaxX )
  return 0;

 double offset = m_dDeltaSpace * sqrt( ( i - m_iHalfX ) * ( i - m_iHalfX ) );

 double HField =
  m_dMaxMagneticField*exp(-0.5*offset*offset/m_dSigma/m_dSigma);
 double Hscale = (n-m_iStartTime) >= ramp ? 1.0 : exp(10.0*(n-m_iStartTime-ramp)/ramp );

 double time =  (n-m_iStartTime) >= 0 ? (n -m_iStartTime)*m_dDeltaTime : 0;

 HField*= Hscale*sin( -2*PI*LIGHT_SPEED/m_dLambda*time + 2.0*PI*getRefractiveIndex()*0.5*m_dDeltaSpace/m_dLambda );

 return HField;
}

double GaussianSource::getMagneticFieldZy( int j, int n )
{
 //if we're a horizontal source then all Z Magnetic field components
 //in the Y direction should not exist
 if( m_iMinY == m_iMaxY )
  return 0;

 double offset = m_dDeltaSpace * sqrt( ( j - m_iHalfY ) * ( j - m_iHalfY ) );

 double HField =
  m_dMaxMagneticField*exp(-0.5*offset*offset/m_dSigma/m_dSigma);
 double Escale = (n-m_iStartTime) >= ramp ? 1.0 : exp(10.0*(n-m_iStartTime-ramp)/ramp );

 double time =  (n-m_iStartTime) >= 0 ? (n -m_iStartTime)*m_dDeltaTime : 0;

 HField*= sin( -2*PI*LIGHT_SPEED/m_dLambda*time + 2.0*PI*getRefractiveIndex()*0.5*m_dDeltaSpace/m_dLambda );

 return HField;
}


double GaussianSource::getElectricFieldX( int i, int n )
{
 //if we're a vertical source then all X Electric field components
 //should not exist
 if( m_iMinX == m_iMaxX )
  return 0;
```

```
double offset = m_dDeltaSpace * sqrt( ( i - m_iHalfX ) * ( i - m_iHalfX ) );
double EField =
 m_dMaxElectricField*exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double Escale = (n-m_iStartTime) >= ramp ? 1.0 : exp(10.0*(n-m_iStartTime-ramp)/ramp );

double time =  (n-m_iStartTime) > 0 ? (n-m_iStartTime+0.5)*m_dDeltaTime : 0;

EField*= Escale*sin( 2.0*PI*LIGHT_SPEED/m_dLambda*time );

return EField;
}

double GaussianSource::getElectricFieldY( int j, int n )
{
//if we're a horizontal source then all Y Electric field components
//should not exist
if( m_iMinY == m_iMaxY )
 return 0;

double offset = m_dDeltaSpace * sqrt( ( j - m_iHalfY ) * ( j - m_iHalfY ) );
double EField =
 m_dMaxElectricField*exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double Escale = (n-m_iStartTime) >= ramp ? 1.0 : exp(10.0*(n-m_iStartTime-ramp)/ramp );

double time =  (n-m_iStartTime) > 0 ? (n-m_iStartTime+0.5)*m_dDeltaTime : 0;

EField*= Escale*sin( 2.0*PI*LIGHT_SPEED/m_dLambda*time );

return EField;
}


GaussianSource::~GaussianSource()
{
cerr << "Message: Deleting gaussian source\n";
}

////////////////////////////////////////////////////////////////////////////////////////////////
//Geometry.cpp -- stores the geometry and material properties
////////////////////////////////////////////////////////////////////////////////////////////////
#include "Geometry.h"
#include "Options.h"
#include "Material.h"
#include "GeometryPoint.h"
#include <cstdio>
#include <iostream.h>

Geometry::Geometry()
{
 points = NULL;
 m_dDelta = 0;
 m_dDeltaTime = 0;
 m_dMinimumLambda = 1E80;
 m_iWidth = 0;
 m_iHeight = 0;
 m_dGridMode = 0;
}

int Geometry::setDeltaSpace( double delta )
{
 m_dDelta = delta;
 return SUCCESS;
}

int Geometry::setDeltaTime( double deltaTime )
{
 m_dDeltaTime = deltaTime;
 return SUCCESS;
}
```

```
int Geometry::setGridTime( int time )
{
 m_iTime = time;
 return SUCCESS;
}

int Geometry::setMinimumLambda( double lambda)
{
 if( m_dMinimumLambda > lambda )
  m_dMinimumLambda = lambda;
 return SUCCESS;
}

Geometry::~Geometry()
{
 if( points !=NULL )
 {
  for(int i=0;i<m_iHeight;i++)
  {
   if( points[i] != NULL )
    delete [] points[i];
  }
  delete [] points;
  cerr << "Deleting: geometry points\n";
 }
}


Source* Geometry::isPointLeftOfSource( int i, int j)
{
 if( i >= (m_iWidth-1) || j >= m_iHeight )
  return NULL;
 else if( getMaterial(i, j)->isBoundary() )
  return NULL;
 else if( getMaterial(i, j)->isSource() )
  return NULL;
 else if( getMaterial(i+1, j)->isSource() )
  return (Source*)getMaterial(i+1,j);
 else
  return NULL;
}

Source* Geometry::isPointBelowSource( int i, int j)
{
 if( i >= m_iWidth || j >= (m_iHeight-1) )
  return NULL;
 else if( getMaterial(i, j)->isBoundary() )
  return NULL;
 else if( getMaterial(i, j)->isSource() )
  return NULL;
 else if( getMaterial(i, j+1)->isSo urce() )
  return (Source*)getMaterial(i,j+1);
 else
  return NULL;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//GeometryPoint.cpp -- defines a single point in the computational space
////////////////////////////////////////////////////////////////////////////////////////////////////
#include "GeometryPoint.h"
#include <assert.h>
#include <string.h>

GeometryPoint::GeometryPoint()
{
 material = NULL;
}
```

```
void GeometryPoint::setMaterial(Material *mat)
{
 assert( mat != NULL );
 material = mat;
}

////////////////////////////////////////////////////////////////////////////////////////////////
//Material.cpp -- stores all material properties
////////////////////////////////////////////////////////////////////////////////////////////////
#include <string.h>
#include <iostream.h>
#include "Material.h"
#include "Geometry.h"

Material* Material::firstMaterial=NULL;
Material* Material::lastMaterial=NULL;
int Material::m_iDispersiveExists=0;

//member functions
Material::Material( long index, char* name, double EC, double MC, double EP, double MP,
      double b1, double b2, double b3, double c1, double c2,
    double c3, double d1, double d2, double d3 )
{
 m_lIndex = index;
 m_bIsBoundary = false;
 m_bIsSource = false;
 m_bIsDispersive = true;
 m_iDispersiveExists = true;
 m_bIsMetal = false;
 m_dElectricConductivity = EC;
 m_dMagneticConductivity = MC;
 m_dElectricPermittivity = EP * EPSILON_NOT;
 m_dMagneticPermeability = MP * MU_NOT;
 m_dB1 = b1;
 m_dB2 = b2;
 m_dB3 = b3;
 m_dC1 = c1==0?0:2*PI*LIGHT_SPEED/c1;
 m_dC2 = c2==0?0:2*PI*LIGHT_SPEED/c2;
 m_dC3 = c3==0?0:2*PI*LIGHT_SPEED/c3;
 m_dD1 = d1;
 m_dD2 = d2;
 m_dD3 = d3;
 m_dWp = 0;
 m_dSF = 0;
 m_sMaterialName = new char[strlen(name)+1];
 strcpy(m_sMaterialName, name);
 nextMaterial = NULL;
 prevMaterial = NULL;

 //add material to list
 if( firstMaterial == NULL )
 {
  firstMaterial = lastMaterial = this;
 }
 else
 {
  lastMaterial->nextMaterial = this;
  this->prevMaterial = lastMaterial;
  lastMaterial = this;
 }
 cerr << "Creating: " << m_sMaterialName << "\n";
}

Material::Material( long index, char* name, double EC, double MC, double EP, double MP,
      double wp, double sf, bool isMetal = false )
{
 m_lIndex = index;
 m_bIsBoundary = false;
 m_bIsSource = false;
 m_bIsMetal = isMetal;
```

```
m_bIsConductive = !isMetal;
m_bIsDispersive = true;
m_iDispersiveExists = true;
m_dElectricConductivity = EC;
m_dMagneticConductivity = MC;
m_dElectricPermittivity = EP * EPSILON_NOT;
m_dMagneticPermeability = MP * MU_NOT;
m_dB1 = 0;
m_dB2 = 0;
m_dB3 = 0;
m_dC1 = 0;
m_dC2 = 0;
m_dC3 = 0;
m_dD1 = 0;
m_dD2 = 0;
m_dD3 = 0;
m_dWp = wp;
m_dSF = sf;
m_sMaterialName = new char[strlen(name)+1];
strcpy(m_sMaterialName, name);
nextMaterial = NULL;
prevMaterial = NULL;

//add material to list
if( firstMaterial == NULL )
{
 firstMaterial = lastMaterial = this;
}
else
{
 lastMaterial->nextMaterial = this;
 this->prevMaterial = lastMaterial;
 lastMaterial = this;
}
cerr << "Creating: " << m_sMaterialName << "\n";
}

Material::Material( long index, char* name, double EC, double MC, double EP, double MP )
{
m_lIndex = index;
m_bIsBoundary = false;
m_bIsSource = false;
m_bIsDispersive = false;
m_bIsMetal = false;
m_bIsConductive = false;
m_dElectricConductivity = EC;
m_dMagneticConductivity = MC;
m_dElectricPermittivity = EP * EPSILON_NOT;
m_dMagneticPermeability = MP * MU_NOT;
m_dB1 = 0;
m_dB2 = 0;
m_dB3 = 0;
m_dC1 = 0;
m_dC2 = 0;
m_dC3 = 0;
m_dD1 = 0;
m_dD2 = 0;
m_dD3 = 0;
m_dWp = 0;
m_dSF = 0;
m_sMaterialName = new char[strlen(name)+1];
strcpy(m_sMaterialName, name);
nextMaterial = NULL;
prevMaterial = NULL;

//add material to list
if( firstMaterial == NULL )
{
 firstMaterial = lastMaterial = this;
}
```

```
else
{
 lastMaterial->nextMaterial = this;
 this->prevMaterial = lastMaterial;
 lastMaterial = this;
}
cerr << "Creating: " << m_sMaterialName << "\n";
}

Material::Material( long index, char* name, Material *m )
{
 m_lIndex = index;
 m_bIsBoundary = false;
 m_bIsSource = false;
 m_dElectricConductivity = m->getElectricConductivity();
 m_dMagneticConductivity = m->getMagneticConductivity();
 m_dElectricPermittivity = m->getElectricPermitt ivity();
 m_dMagneticPermeability = m->getMagneticPermeability();
 m_dB1 = m->m_dB1;
 m_dB2 = m->m_dB2;
 m_dB3 = m->m_dB3;
 m_dC1 = m->m_dC1;
 m_dC2 = m->m_dC2;
 m_dC3 = m->m_dC3;
 m_dD1 = m->m_dD1;
 m_dD2 = m->m_dD2;
 m_dD3 = m->m_dD3;
 m_dWp = m->m_dWp;
 m_dSF = m->m_dSF;
 m_bIsDispersive = m->m_bIsDispersive;
 m_bIsMetal = m->m_bIsMetal;
 m_bIsConductive = m->m_bIsConductive;
 m_sMaterialName = new char[strlen(name)+1];
 strcpy(m_sMaterialName, name);
 nextMaterial = NULL;
 prevMaterial = NULL;

 //add material to list
 if( firstMaterial == NULL )
 {
  firstMaterial = lastMaterial = this;
 }
 else
 {
  lastMaterial->nextMaterial = this;
  this->prevMaterial = lastMaterial;
  lastMaterial = this;
 }
 cerr << "Message: Creating " << m_sMaterialName << " \n";
}

Material::~Material()
{
 if( this->prevMaterial != NULL )
  this->prevMaterial->nextMaterial = this->nextMaterial;
 if( this->nextMaterial != NULL )
 this->nextMaterial->prevMaterial = this->prevMaterial;

 cerr << "Deleting: " << m_sMaterialName << "\n";
 delete m_sMaterialName;
 m_sMaterialName = NULL;
}

Material* Material::materialExists( long index )
{
 //cerr << "Checking index: " << index << "\n";
 Material *mat = firstMaterial;

 while( mat != NULL )
 {
```

```
 //cerr << "Index is: " << mat->getIndex() << " for material " << mat->getName() << "\n";
 if( mat->getIndex() == index )
  return mat;
 mat=mat->nextMaterial;
 }
 cerr << "Checking index: " << index << "\n";
 mat = firstMaterial;
 while( mat != NULL )
 {
  cerr << "Index is: " << mat->getIndex() << " for material " << mat->getName() << "\n";
  if( mat->getIndex() == index )
   return mat;
  mat=mat->nextMaterial;
 }
 return NULL;
}


long Material::getIndexForColor( BYTE blue, BYTE green, BYTE red )
{
 long index = 0;
 index |= blue<<16;
 index |= green<<8;
 index |= red;
 return index;
}


int Material::initializeMaterials( Geometry *geometry )
{
 Material *mat = firstMaterial;
 int rc = SUCCESS;

 while( mat != NULL )
 {
  rc |= mat->initialize( geometry );
  mat = mat->nextMaterial;
 }

 return rc;
}

int Material::initialize( Geometry *geometry )
{
 int rc = SUCCESS;
 double deltaTime = geometry->getDeltaTime();
 double deltaSpace = geometry->getDeltaSpace();
 cerr << "Message: Initializing material " << this->getName() << "\n";

 m_dElectricCoefficientA = ( 1 - m_dElectricConductivity*deltaTime/( 2*m_dElectricPermittivity ) ) / ( 1 +
m_dElectricConductivity*deltaTime/( 2*m_dElectricPermittivity ) );
 m_dElectricCoefficientB = ( deltaTime/( deltaSpace*m_dElectricPermittivity ) ) / ( 1 + m_dElectricConductivity*deltaTime/(
2*m_dElectricPermittivity ) );
 m_dMagneticCoefficientA = ( 1 - m_dMagneticConductivity*deltaTime/( 2*m_dMagneticPermeability ) ) / ( 1 +
m_dMagneticConductivity*deltaTime/( 2*m_dMagneticPermeability ) );
 m_dMagneticCoefficientB = ( deltaTime/( deltaSpace*m_dMagneticPermeability ) ) / ( 1 + m_dMagneticConductivity*deltaTime/(
2*m_dMagneticPermeability ) );


 m_dA1 = 2 + getSellmeierCoefficientD1()*geometry->getDeltaTime() +
   getSellmeierCoefficientC1() * getSellmeierCoefficientC1() *
   geometry->getDeltaTime() * geometry->getDeltaTime()
   * (1 + getSellmeierCoefficientB1() );

 m_dA2 = 2 + getSellmeierCoefficientD2()*geometry->getDeltaTime() +
   getSellmeierCoefficientC2() * getSellmeierCoefficientC2() *
   geometry->getDeltaTime() * geometry->getDeltaTime()
   * (1 + getSellmeierCoefficientB2() );

 m_dA3 = 2 + getSellmeierCoefficientD3()*geometry->getDeltaTime() +
```

```
  getSellmeierCoefficientC3() * getSellmeierCoefficientC3() *
  geometry->getDeltaTime() * geometry->getDeltaTime()
  * (1 + getSellmeierCoefficientB3() );

m_dW1 = getSellmeierCoefficientB1() * geometry->getDeltaTime() *
  geometry->getDeltaTime() * getSellmeierCoefficientC1() *
  getSellmeierCoefficientC1();

m_dW2 = getSellmeierCoefficientB2() * geometry->getDeltaTime() *
  geometry->getDeltaTime() * getSellmeierCoefficientC2() *
  getSellmeierCoefficientC2();

m_dW3 = getSellmeierCoefficientB3() * geometry->getDeltaTime() *
  geometry->getDeltaTime() * getSellmeierCoefficientC3() *
  getSellmeierCoefficientC3();

m_dG1 = -2 + getSellmeierCoefficientD1()*geometry->getDeltaTime() -
  getSellmeierCoefficientC1() * getSellmeierCoefficientC1() *
  geometry->getDeltaTime() * geometry->getDeltaTime()
  * (1 + getSellmeierCoefficientB1() );

m_dG2 = -2 + getSellmeierCoefficientD2()*geometry->getDeltaTime() -
  getSellmeierCoefficientC2() * getSellmeierCoefficientC2() *
  geometry->getDeltaTime() * geometry->getDeltaTime()
  * (1 + getSellmeierCoefficientB2() );

m_dG3 = -2 + getSellmeierCoefficientD3()*geometry->getDeltaTime() -
  getSellmeierCoefficientC3() * getSellmeierCoefficientC3() *
  geometry->getDeltaTime() * geometry->getDeltaTime()
  * (1 + getSellmeierCoefficientB3() );

m_dH1 = (2.0 + getScatteringFrequency()* geometry->getDeltaTime() );

m_dH2 = (2.0 - getScatteringFrequency()*geometry->getDeltaTime() );

m_dH3 = ( getPlasmaFrequency()*getPlasmaFrequency()*geometry->getDeltaTime()*geometry->getDeltaTime()*EPSILON_NOT
 - getScatteringFrequency()*geometry->getDeltaTime()*getElectricPermittivity()
 + 2*getElectricPermittivity() );

m_dH4 = ( getPlasmaFrequency()*getPlasmaFrequency()*geometry->getDeltaTime()*geometry->getDeltaTime()*EPSILON_NOT
 + getScatteringFrequency()*geometry->getDeltaTime()*getElectricPermittivity()
 + 2*getElectricPermittivity() );

return rc;
}

void Material::deleteMaterials()
{
 Material *mat = firstMaterial;
 while( mat != NULL )
 {
  Material *nextMat = mat->nextMaterial;
  delete mat;
  mat = nextMat;
 }
 firstMaterial = lastMaterial = NULL;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//Options.cpp – stores the option parameters for a given simulation
////////////////////////////////////////////////////////////////////////////////////////////////////
#include "Options.h"
#include "stdheader.h"
#include <iostream.h>
#include <math.h>
#include <string.h>

Options::Options()
{
 m_iFrameSpacing = 0;
```

```
m_sSimulationName = NULL;
m_bPulsedMode = false;
cerr << "Message: Creating options object \n";
}

Options::~Options()
{
cerr << "Message: Deleting options object \n";
delete [] m_sSimulationName;
if( m_IXSamplingPoints != NULL )
 delete [] m_IXSamplingPoints;
if( m_IYSamplingPo ints != NULL )
 delete [] m_IYSamplingPoints;
m_iFrameSpacing = 0;
}


void Options::setFrameSpacing( int spacing )
{
m_iFrameSpacing = abs(spacing);
}

void Options::setSimulationName( char* simulationName )
{
if( simulationName == NULL )
 {
 m_sSimulationName = new char[strlen("fdtd")+1];
 strcpy(m_sSimulationName,"fdtd");
 }
 else if( strlen(simulationName) == 0 )
 {
 m_sSimulationName = new char[strlen("fdtd")+1];
 strcpy(m_sSimulationName,"fdtd");
 }
 else
 {
 m_sSimulationName = new char[strlen(simulationName)+1];
 strcpy(m_sSimulationName, simulationName);
 }
}


void Options::setSimulationMode( char* mode )
{
//cerr << "Simulation mode " << mode;
if( strcmp(mode, TM_TE_MODE_TOKEN) == 0 )
 m_iSimulationMode = TM_TE_MODE;
else if( strcmp(mode, TM_MODE_TOKEN) == 0 )
 m_iSimulationMode = TM_MODE;
else if( strcmp(mode, TE_MODE_TOKEN) == 0 )
 m_iSimulationMode = TE_MODE;
else
 m_iSimulationMode = TM_TE_MODE;
}

void Options::setPulsedMode( bool mode )
{
m_bPulsedMode = mode;
}


void Options::setOutputMode( char* mode, double min, double max )
{
//cerr << "Output mode " << mode;
if( strcmp(mode, BMP_TXT_OUTPUT_MODE_TOKEN) == 0 )
 m_iOutputMode = BMP_TXT_OUTPUT_MODE;
else if( strcmp(mode, BMP_OUTPUT_MODE_TOKEN) == 0 )
 m_iOutputMode = BMP_OUTPUT_MODE;
else if( strcmp(mode, TXT_OUTPUT_MODE_TOKEN) == 0 )
 m_iOutputMode = TXT_OUTPUT_MODE;
```

```
else
 m_iOutputMode = BMP_OUTPUT_MODE;
m_dMaxOutputFieldStrength = max;
m_dMinOutputFieldStrength = min;
}

void Options::setTimeDomainFieldSamplingPointCount( int count )
{
 m_iSamplingPointCount = count;
 if( count > 0 )
 {
  m_IXSamplingPoints = new int[count];
  m_IYSamplingPoints = new int[count];
 }
 else
 {
  m_IXSamplingPoints = NULL;
  m_IYSamplingPoints = NULL;
 }
}

void Options::setTimeDomainFieldSamplingPoint( int index, int x, int y )
{
 m_IXSamplingPoints[index] = x;
 m_IYSamplingPoints[index] = y;
}

void Options::setXIntegrationLineCount( int count )
{
 m_iXIntegrationLineCount = count;
 if( count > 0 )
 {
  m_IXIntegrationLine = new int[count];
 }
 else
 {
  m_IXIntegrationLine = NULL;
 }
}

void Options::setYIntegrationLineCount( int count )
{
 m_iYIntegrationLineCount = count;
 if( count > 0 )
 {
  m_IYIntegrationLine = new int[count];
 }
 else
 {
  m_IYIntegrationLine = NULL;
 }
}

void Options::setXIntegrationLine( int index, int x )
{
 m_IXIntegrationLine[index] = x;
}

void Options::setYIntegrationLine( int index, int y )
{
 m_IYIntegrationLine[index] = y;
}

void Options::setQueryContinue( bool cont )
{
 m_bQueryContinue = cont ;
}

////////////////////////////////////////////////////////////////////////////////////////////////
//PMLBoundaryCondision.cpp -- specifies the boundary condition parameters
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////
#include <string.h>
#include <iostream.h>
#include "PerfectlyMatchedLayerBoundaryCondition.h"
#include "stdheader.h"
#include "Geometry.h"
#include <math.h>

PerfectlyMatchedLayerBoundaryCondition*
 PerfectlyMatchedLayerBoundaryCondition::boundaryCondition = NULL;

//member functions
PerfectlyMatchedLayerBoundaryCondition::PerfectlyMatchedLayerBoundaryCondition( long index, char* name, int width, double ec
)
:Material( index, name, 0, 0, 1.0, 1.0 )
{
 cerr << "Creating: Perfectly matched boundary layer with conductivity " << ec << " and width " << width << "\n";
 m_dElectricConductivity = ec;
 m_dMagneticConductivity = m_dElectricConductivity * MU_NOT / EPSILON_NOT;
 m_iWidth = width;
 m_bIsBoundary = true;
 boundaryCondition = this;
 m_DMagneticExpCoefficientsA = new double[m_iWidth];
 m_DElectricExpCoefficientsA = new double[m_iWidth];
 m_DMagneticExpCoefficientsB = new double[m_iWidth];
 m_DElectricExpCoefficientsB = new double[m_iWidth];
}

int PerfectlyMatchedLayerBoundaryCondition::initialize( Geometry *geometry )
{
 Material::initialize( geometry );
 int rc = SUCCESS;
 cerr << "Message: Initializing boundary condition " << this->getName() << "\n";
 double deltaTime = geometry->getDeltaTime();
 double deltaSpace = geometry->getDeltaSpace();

 /*for( int i = 0; i<m_iWidth; i++ )
 {
  //convention is that sigma increases as index increases
  double sigmaE = m_dElectricConductivity;//*(double)(i)/(double)m_iWidth;
  double sigmaM = m_dMagneticConductivity;//*(double)(i)/(double)m_iWidth;

  m_DElectricExpCoefficientsB[i] = ( deltaTime/( deltaSpace*m_dElectricPermittivity ) ) / ( 1 + sigmaE*deltaTime/(
2*m_dElectricPermittivity ) ) ;

  m_DMagneticExpCoefficientsB[i] = ( deltaTime/( deltaSpace*m_dMagneticPermeability ) ) / ( 1 + sigmaM*deltaTime/(
2*m_dMagneticPermeability ) ) ;

  m_DElectricExpCoefficientsA[i] = ( 1 - m_dElectricConductivity*deltaTime/( 2*m_dElectricPermittivity ) ) / ( 1 +
sigmaE*deltaTime/( 2*m_dElectricPermittivity ) ) ;

  m_DMagneticExpCoefficientsA[i] = ( 1 - m_dMagneticConductivity*deltaTime/( 2*m_dMagneticPermeability ) ) / ( 1 +
sigmaM*deltaTime/( 2*m_dMagneticPermeability ) ) ;

 }

 double sigmaE = 0;
 double sigmaM = 0;
 m_dElectricCoefficientB = ( deltaTime/( deltaSpace*m_dElectricPermittivity ) ) / ( 1 + sigmaE*deltaTime/(
2*m_dElectricPermittivity ) ) ;
 m_dMagneticCoefficientB = ( deltaTime/( deltaSpace*m_dMagneticPermeability ) ) / ( 1 + sigmaM*deltaTime/(
2*m_dMagneticPermeability ) ) ;
 m_dElectricCoefficientA = ( 1 - m_dElectricConductivity*deltaTime/( 2*m_dElectricPermittivity ) ) / ( 1 + sigmaE*deltaTime/(
2*m_dElectricPermittivity ) ) ;
 m_dMagneticCoefficientA = ( 1 - m_dMagneticConductivity*deltaTime/( 2*m_dMagneticPermeability ) ) / ( 1 +
sigmaM*deltaTime/( 2*m_dMagneticPermeability ) ) ;
 */

 for( int i = 0; i<m_iWidth; i++ )
 {
```

```
//convention is that sigma increases as index increases
double sigmaE = m_dElectricConductivity*(double)(i+0.5)/(double)m_iWidth;
double sigmaM = m_dMagneticConductivity*(double)i/(double)m_iWidth;

m_DElectricExpCoefficientsB[i] =
 (1-exp(-sigmaE*deltaTime/EPSILON_NOT))/sigmaE/deltaSpace;

m_DMagneticExpCoefficientsB[i] =
 (1-exp(-sigmaM*deltaTime/MU_NOT))/sigmaM/deltaSpace;

m_DElectricExpCoefficientsA[i] =
 exp(-sigmaE*deltaTime/EPSILON_NOT);

m_DMagneticExpCoefficientsA[i] =
 exp(-sigmaM*deltaTime/MU_NOT);

}

m_DMagneticExpCoefficientsB[0] = deltaTime*INVERSE_MU_NOT/deltaSpace;
//m_DElectricExpCoefficientsB[0] = deltaTime*INVERSE_EPSILON_NOT/deltaSpace;

m_dElectricCoefficientB = deltaTime*INVERSE_EPSILON_NOT/deltaSpace;
m_dMagneticCoefficientB = deltaTime*INVERSE_MU_NOT/deltaSpace;
m_dElectricCoefficientA = 1.0;
m_dMagneticCoefficientA = 1.0;

return rc;

return rc;
}

PerfectlyMatchedLayerBoundaryCondition::~PerfectlyMatchedLayerBoundaryCondition()
{
 if( m_DElectricExpCoefficientsA != NULL )
  delete m_DElectricExpCoefficientsA;
 if( m_DMagneticExpCoefficientsA != NULL )
  delete m_DMagneticExpCoefficientsA;
 if( m_DElectricExpCoefficientsB != NULL )
  delete m_DElectricExpCoefficientsB;
 if( m_DMagneticExpCoefficientsB != NULL )
  delete m_DMagneticExpCoefficientsB;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//PropertyReader.cpp – reads in the options and parameters for the simulation
////////////////////////////////////////////////////////////////////////////////////////////////////
#include "PropertyReader.h"
#include <cstdio>
#include <cstring>
#include <iostream.h>
#include "Geometry.h"
#include "Options.h"
#include <assert.h>
#include "Material.h"
#include <math.h>
#include "GaussianSource.h"
#include "TerahertzSource.h"
#include "PulsedGaussianSource.h"
#include "PerfectlyMatchedLayerBoundaryCondition.h"

PropertyReader::PropertyReader()
{
 readState = -1;
 geometry = NULL;
 cerr << "Creating: PropertyReader \n";
}

/** Read the property file */
int PropertyReader::readProperties( char *propFile, Geometry *geom, Options *opt )
{
```

```
FILE* propertyFP;
int rc = SUCCESS;
char line[2048];
geometry = geom;
options = opt;
lastMaterial = NULL;

propertyFP = fopen( propFile, "rt");
if( propertyFP == NULL )
{
 cerr << "Property file not found!\n";
 return FILE_NOT_FOUND;
}

while( (fgets(line, 2048, propertyFP ) != NULL) && (rc == SUCCESS) )
{
 rc = parseLine( line );
}

fclose( propertyFP );
return rc;
}

int PropertyReader::parseLine( char* line )
{
 assert( line != NULL );
 int rc = SUCCESS;
 if( strstr(line, GRID_TOKEN ) != NULL )
  readState = GRID_STATE;
 else if( strstr(line, MATERIAL_TOKEN ) != NULL )
  readState = MATERIAL_STATE;
 else if( strstr(line, OPTIONS_TOKEN ) != NULL )
  readState = OPTIONS_STATE;
 else if( line[0] == '%' )
  cerr << "Comment: " << &line[1];
 else if( readState == MATERIAL_STATE )
  rc = parseMaterial( line );
 else if( readState == GRID_STATE )
  rc = parseGrid( line );
 else if( readState == OPTIONS_STATE )
  rc = parseOptions( line );
 else
  cerr << "Warning: Invalid property! Skipping line...\n";
 return rc;
}

int PropertyReader::parseGrid( char* line )
{
 int rc = SUCCESS;
 char *position;
 if( (position = strstr(line, DELTA_TOKEN)) != NULL )
 {
  double delta=0;
  position = position + strlen(DELTA_TOKEN);
  if( sscanf( position,"%le",&delta ) != 1 )
   rc = INVALID_FILE_FORMAT;
  geometry->setDeltaSpace( delta );
 }
 else if( (position = strstr(line, DELTA_TIME_TOKEN)) != NULL )
 {
  double time;
  position = position + strlen(DELTA_TIME_TOKEN);
  if( sscanf( position,"%le",&time ) != 1 )
   rc = INVALID_FILE_FORMAT;
  geometry->setDeltaTime( time );
 }
 else if( (position = strstr(line, GRID_TIME_TOKEN)) != NULL )
 {
  int time;
  position = position + strlen(GRID_TIME_TOKEN);
```

```
 if( sscanf( position,"%d",&time ) != 1 )
  rc = INVALID_FILE_FORMAT;
  geometry->setGridTime( time );
 }
 else
 {
  cerr << "Unknown token: " << line;
 }
 return rc;
}

int PropertyReader::parseMaterial( char* line )
{
 int rc = SUCCESS;
 char name[2048];
 char type[2048];
 unsigned char b,g,r;
 if( sscanf(line, "%hhi %hhi %hhi %s %s",&b, &g, &r,name, type) != 5 )
 {
  rc = INVALID_FILE_FORMAT;
 }
 else
 {
  long index = Material::getIndexForColor(b,g,r);
  Material *material = Material::materialExists( index );
  if( material == NULL )
  {
   line = strstr( line, type );
   line += strlen( type );
   rc = parseMaterialType(index, name, type, line);
  }
  else
  {
   cerr << "Error: Attempted to read in an existing material";
   rc = INVALID_FILE_FORMAT;
  }
 }
 return rc;
}

int PropertyReader::parseMaterialType(long index, char* name, char* type, char *line)
{
 int rc = SUCCESS;
 Material *mat = NULL;

 if( strcmp( type, DISPERSIVE_MATERIAL_TYPE_TOKEN ) == 0 )
 {
  double ec, mc, ep, mp;
  double b1,b2,b3,c1,c2,c3,d1,d2,d3;
  if( sscanf(line, "%le %le %le %le %le %le %le %le %le %le %le %le",
   &ec, &mc, &ep, &mp, &b1, &b2, &b3,
   &c1, &c2, &c3,
   &d1, &d2, &d3 ) == 13 )
  {
   mat = new Material( index, name, ec, mc, ep, mp,
    b1, b2, b3,
    c1, c2, c3,
    d1, d2, d3 );
   this->lastMaterial = mat;
  }
  else
  {
   rc = INVALID_FILE_FORMAT;
  }
 }
 else if( strcmp( type, METAL_MATERIAL_TYPE_TOKEN ) == 0 )
 {
  double ec, mc, ep, mp;
  double wp, sf;
  if( sscanf(line, "%le %le %le %le %le %le", &ec, &mc, &ep, &mp, &wp, &sf ) == 6 )
```

```
 {
 mat = new Material( index, name, ec, mc, ep, mp, wp, sf, true );
 this->lastMaterial = mat;
 }
 else
 {
 rc = INVALID_FILE_FORMAT;
 }
}
else if( strcmp( type, CONDUCTIVE_MATERIAL_TYPE_TOKEN ) == 0 )
{
 double ec, mc, ep, mp;
 double wp, sf;
 if( sscanf(line, "%le %le %le %le %le %le", &ec, &mc, &ep, &mp, &wp, &sf) == 6 )
 {
 mat = new Material( index, name, ec, mc, ep, mp, wp, sf, false );
 this->lastMaterial = mat;
 }
 else
 {
 rc = INVALID_FILE_FORMAT;
 }
}
else if( strcmp( type, MATERIAL_TYPE_TOKEN ) == 0 )
{
 double ec, mc, ep, mp;
 if( sscanf(line, "%le %le %le %le", &ec, &mc, &ep, &mp) == 4 )
 {
 mat = new Material( index, name, ec, mc, ep, mp );
 this->lastMaterial = mat;
 }
 else
 {
 rc = INVALID_FILE_FORMAT;
 }
}
else if( strcmp( type, BOUNDARY_TYPE_TOKEN ) == 0 )
{
 char bcType[2048];
 if( sscanf(line, "%s", bcType) != 1 )
 {
 rc = INVALID_FILE_FORMAT;
 }
 else
 {
 line = strstr(line, bcType) + strlen(bcType);
 if( strcmp( bcType, PERFECTLY_MATCHED_LAYER_TOKEN ) == 0 )
 {
  double ec;
  int width;
  if( sscanf(line, "%d %le", &width, &ec ) != 2 )
  {
  cerr << "Error: Invalid perfectly matched boundary condition parameters\n";
  rc = INVALID_FILE_FORMAT;
  }
  else
  {
  mat = new PerfectlyMatchedLayerBoundaryCondition( index, name, width, ec );
  }
 }
 else
 {
  cerr << "Warning: Invalid boundary type\n";
  rc = INVALID_FILE_FORMAT;
 }
 }
}
else if( strcmp( type, SOURCE_TYPE_TOKEN ) == 0 )
{
 char srcType[2048];
```

```
if( sscanf(line, "%s", srcType) != 1 )
{
rc = INVALID_FILE_FORMAT;
}
else
{
line = strstr(line,srcType) + strlen(srcType);
if( strcmp( srcType, GAUSSIAN_SOURCE_TOKEN ) == 0 )
{
if( lastMaterial == NULL )
{
cerr << "Error: Invalid source location\n";
cerr << "Error: Source definition must be placed directly after its corresponding material\n";
rc = INVALID_FILE_FORMAT;
}
else
{
double eField, FWHM, lambda;
int startTime;
if( sscanf(line," %le %d %le %le",&eField, &startTime, &lambda,
&FWHM) != 4 )
{
cerr << "Error: Invalid Source Options\n";
rc = INVALID_FILE_FORMAT;
}
mat = new GaussianSource( index, name, lastMaterial, eField,
startTime, lambda, FWHM );
}
}
else if( strcmp( srcType, PULSED_GAUSSIAN_SOURCE_TOKEN ) == 0 )
{
if( lastMaterial == NULL )
{
cerr << "Error: Invalid source location\n";
cerr << "Error: Source definition must be placed directly after its corresponding material\n";
rc = INVALID_FILE_FORMAT;
}
else
{
double eField, FWHM, pFWHM, phi, lambda;
int startTime;
if( sscanf(line," %le %d %le %le %le %le",&eField, &startTime,
&lambda, &FWHM, &pFWHM, &phi) != 6 )
{
cerr << "Error: Invalid Source Options\n";
rc = INVALID_FILE_FORMAT;
}
mat = new PulsedGaussianSource( index, name, lastMaterial, eField,
startTime, lambda, FWHM, pFWHM, phi);
}
}
else if( strcmp( srcType, TERAHERTZ_SOURCE_TOKEN ) == 0 )
{
if( lastMaterial == NULL )
{
cerr << "Error: Invalid source location\n";
cerr << "Error: Source definition must be placed directly after its corresponding material\n";
rc = INVALID_FILE_FORMAT;
}
else
{
double eField, FWHM, pFWHM, phi, lambda;
int startTime;
if( sscanf(line," %le %d %le %le %le %le",&eField, &startTime,
&lambda, &FWHM, &pFWHM, &phi) != 6 )
{
cerr << "Error: Invalid Source Options\n";
rc = INVALID_FILE_FORMAT;
}
mat = new TerahertzSource( index, name, lastMaterial, eField,
```

```
    startTime, lambda, FWHM, pFWHM, phi);
   }
  }
  else
  {
   cerr << "Warning: Invalid source type - " << srcType <<"\n";
   rc = INVALID_FILE_FORMAT;
  }
 }
}
else
{
 cerr << "Warning: Invalid material type\n";
 rc = INVALID_FILE_FORMAT;
}
return rc;
}

int PropertyReader::parseOptions( char* line )
{
 int rc = SUCCESS;
 char *position;
 if( (position = strstr(line, FRAME_SPACING_TOKEN)) != NULL )
 {
  int frameSpacing=0;
  position = position + strlen(FRAME_SPACING_TOKEN);
  if( sscanf( position,"%d",&frameSpacing) != 1 )
   rc = INVALID_FILE_FORMAT;
  options->setFrameSpacing(frameSpacing);
 }
 else if( (position = strstr(line, SIMULATION_NAME_TOKEN)) != NULL )
 {
  char name[2048];
  position = position + strlen(SIMULATION_NAME_TOKEN);
  if( sscanf( position,"%s",&name ) != 1 )
   rc = INVALID_FILE_FORMAT;
  options->setSimulationName( name );
 }
 else if( (position = strstr(line, SIMULATION_MODE_TOKEN)) != NULL )
 {
  char mode[2048];
  position = position + strlen(SIMULATION_MODE_TOKEN);
  if( sscanf( position,"%s",&mode ) != 1 )
   rc = INVALID_FILE_FORMAT;
  options->setSimulationMode( mode );
 }
 else if( (position = strstr(line, SIMULATION_OUTPUT_TOKEN)) != NULL )
 {
  char mode[2048];
  double min, max;
  position = position + strlen(SIMULATION_OUTPUT_TOKEN);
  if( sscanf( position,"%s %le %le",&mode,&min,&max ) != 3 )
   rc = INVALID_FILE_FORMAT;
  options->setOutputMode( mode, min, max );
 }
 else if( (position = strstr(line, QUERY_CONTINUE_TOKEN)) != NULL )
 {
  char mode[2048];
  int cont;
  position = position + strlen(QUERY_CONTINUE_TOKEN);
  if( sscanf( position,"%d",&cont ) != 1 )
   rc = INVALID_FILE_FORMAT;
  options->setQueryContinue( cont!=0 );
  if ( cont )
   cerr << "Message: Option to continue is active.\n";
  else
   cerr << "Message: Option to continue is dormant.\n";
 }
 else if( (position = strstr(line, TIME_DOMAIN_FIELD_OUTPUT_TOKEN)) != NULL )
 {
```

```
int count;
int x,y;
char test[2048];
position = position + strlen(TIME_DOMAIN_FIELD_OUTPUT_TOKEN);
if( sscanf( position,"%d ", &count ) != 1 )
 rc = INVALID_FILE_FORMAT;
options->setTimeDomainFieldSamplingPointCount( count );
count--;
sprintf(test,"%d ",count);
position = position + strlen( test );
for( ; count>=0&&rc==SUCCESS ; count-- )
{
 if( sscanf( position,"%d %d ", &x, &y ) != 2 )
  rc = INVALID_FILE_FORMAT;
 options->setTimeDomainFieldSamplingPoint( count, x, y );
 sprintf(test,"%d %d ",x,y);
 position = position + strlen( test );
}
cerr << "Message: Sampling " << options->getSamplingPointCount() << " points for output!\n";
}
else if( (position = strstr(line, X_FIELD_INTEGRATOR_TOKEN)) != NULL )
{
 int count;
 int x;
 char test[2048];
 position = position + strlen(X_FIELD_INTEGRATOR_TOKEN);
 if( sscanf( position,"%d ", &count ) != 1 )
  rc = INVALID_FILE_FORMAT;
 options->setXIntegrationLineCount( count );
 count--;
 sprintf(test,"%d ",count);
 position = position + strlen( test );
 for( ; count>=0&&rc==SUCCESS ; count-- )
 {
  if( sscanf( position,"%d ", &x) != 1 )
   rc = INVALID_FILE_FORMAT;
  options->setXIntegrationLine( count, x );
  sprintf(test,"%d ",x);
  position = position + strlen( test );
 }
 cerr << "Message: Sampling " << options->getXIntegrationLineCount() << " x integration lines for output!\n";
}
else if( (position = strstr(line, Y_FIELD_INTEGRATOR_TOKEN)) != NULL )
{
 int count;
 int y;
 char test[2048];
 position = position + strlen(Y_FIELD_INTEGRATOR_TOKEN);
 if( sscanf( position,"%d ", &count ) != 1 )
  rc = INVALID_FILE_FORMAT;
 options->setYIntegrationLineCount( count );
 count--;
 sprintf(test,"%d ",count);
 position = position + strlen( test );
 for( ; count>=0&&rc==SUCCESS ; count-- )
 {
  if( sscanf( position,"%d ", &y) != 1 )
   rc = INVALID_FILE_FORMAT;
  options->setYIntegrationLine( count, y );
  sprintf(test,"%d ",y);
  position = position + strlen( test );
 }
 cerr << "Message: Sampling " << options->getYIntegrationLineCount() << " y integration lines for output!\n";
}
else
{
 cerr << "What the hell!\n";
}
return rc;
}
```

```
PropertyReader::~PropertyReader()
{
cerr << "Deleting: PropertyReader \n";
}

////////////////////////////////////////////////////////////////////////////////////////////////
//Pulsed Gaussian Source.cpp – instantiates a pulsed gaussian source
////////////////////////////////////////////////////////////////////////////////////////////////
#include "PulsedGaussianSource.h"
#include "GaussianSource.h"
#include <iostream.h>
#include <math.h>
#include "Geometry.h"


PulsedGaussianSource::PulsedGaussianSource(
      long index, char *name, Material *lastMaterial,
      double maxEField, int startTime, double lambda,
      double FWHM, double pulseFWHM, double phi)
:GaussianSource( index, name, lastMaterial, maxEField, startTime, lambda, FWHM )
{
 m_dPhi = phi/10.0*m_dAngularFrequency*m_dAngularFrequency;
 m_dPulseOffset = 2*pulseFWHM;
 m_dPulseWidth = pulseFWHM;
 m_dPulseSigma = m_dPulseWidth / ( 2 * sqrt( 2*log( 2 ) ) );
 cerr << "Message: Creating "<<m_dPulseWidth << "s FWHM pulsed gaussian source\n";
}

int PulsedGaussianSource::initialize( Geometry *geometry )
{
 int rc = GaussianSource::initialize( geometry );
 cerr << "Message: Pulse max occurs at time " << m_dPulseOffset << " seconds\n";
 if( m_dPulseOffset >= geometry->getRunningTime() )
 {
  cerr << "Warning: Pulse maximum occurs after the last time step\n";
 }
 return rc;

}

//TE functions

double PulsedGaussianSource::getElectricFieldZx( int i, int n )
{
 //if we're a vertical source then all Z Electric field components
 //in the x direction should not exist
 if( m_iMinX == m_iMaxX )
  return 0;

 double offset = m_dDeltaSpace * sqrt( ( i - m_iHalfX ) * ( i - m_iHalfX ) );

 double EField = m_dMaxElectricField
   *exp(-0.5*offset*offset/m_dSigma/m_dSigma);
 double time =  (n-m_iStartTime) >= 0 ? (n -m_iStartTime)*m_dDeltaTime : 0;

 EField*= cos( (m_dAngularFrequency + m_dPhi*(time - m_dPulseOffset) )*(time - m_dPulseOffset) );

 EField*= exp( - (time-m_dPulseOffset)*(time-m_dPulseOffset)
  / ( 2 * m_dPulseSigma * m_dPulseSigma ) );

 return EField;
}

double PulsedGaussianSource::getElectricFieldZy( int j, int n )
{
 //if we're a horizontal source then all Z Electric field components
 //in the Y direction should not exist
 if( m_iMinY == m_iMaxY )
  return 0;
```

```
double offset = m_dDeltaSpace * sqrt( ( j - m_iHalfY ) * ( j - m_iHalfY ) );

double EField = m_dMaxElectricField
  *exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double time =  (n-m_iStartTime) >= 0 ? (n -m_iStartTime)*m_dDeltaTime : 0;

EField*= cos(  (m_dAngularFrequency + m_dPhi*(time - m_dPulseOffset) )*(time - m_dPulseOffset) );

EField*= exp( - (time-m_dPulseOffset)*(time-m_dPulseOffset)
 / ( 2 * m_dPulseSigma * m_dPulseSigma ) );

return EField;
}

double PulsedGaussianSource::getMagneticFieldX( int i, int n )
{
//if we're a vertical source then all X Magnetic field components
//should not exist
if( m_iMinX == m_iMaxX )
 return 0;

double offset = m_dDeltaSpace * sqrt( ( i - m_iHalfX ) * ( i - m_iHalfX ) );
double HField = m_dMaxMagneticField
  *exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double time =  (n-m_iStartTime) > 0 ? (n-m_iStartTime+0.5)*m_dDeltaTime : 0;

HField*= cos(  (m_dAngularFrequency + m_dPhi*(time - m_dPulseOffset) )*(time - m_dPulseOffset)
 - m_dAngularFrequency*INVERSE_LIGHT_SPEED*getRefractiveIndex()*0.5*m_dDeltaSpace );

HField*=  exp( - ( (time-m_dPulseOffset+INVERSE_LIGHT_SPEED*getRefractiveIndex()*0.5*m_dDeltaSpace)*(time-
m_dPulseOffset+INVERSE_LIGHT_SPEED*getRefractiveIndex()*0.5*m_dDeltaSpace) )
 / ( 2 * m_dPulseSigma * m_dPulseSigma ) );

return HField;
}

double PulsedGaussianSource::getMagneticFieldY( int j, int n )
{
//if we're a horizontal source then all Y Magnetic field components
//should not exist
if( m_iMinY == m_iMaxY )
 return 0;

double offset = m_dDeltaSpace * sqrt( ( j - m_iHalfY ) * ( j - m_iHalfY ) );
double HField = m_dMaxMagneticField
  *exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double time =  (n-m_iStartTime) > 0 ? (n-m_iStartTime+0.5)*m_dDeltaTime : 0;

HField*= cos(  (m_dAngularFrequency + m_dPhi*(time - m_dPulseOffset) )*(time - m_dPulseOffset)
 - m_dAngularFrequency*INVERSE_LIGHT_SPEED*getRefractiveIndex()*0.5*m_dDeltaSpace );

HField*=  exp( - ( (time-m_dPulseOffset+INVERSE_LIGHT_SPEED*getRefractiveIndex()*0.5*m_dDeltaSpace)*(time-
m_dPulseOffset+INVERSE_LIGHT_SPEED*getRefractiveIndex()*0.5*m_dDeltaSpace) )
 / ( 2 * m_dPulseSigma * m_dPulseSigma ) );


return HField;
}

//TM Functions
double PulsedGaussianSource::getMagneticFieldZx( int i, int n )
{
//if we're a vertical source then all Z Magnetic field components
//in the x direction should not exist
if( m_iMinX == m_iMaxX )
 return 0;

double offset = m_dDeltaSpace * sqrt( ( i - m_iHalfX ) * ( i - m_iHalfX ) );
```

```
double HField = m_dMaxMagneticField
  *exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double time = (n-m_iStartTime) >= 0 ? (n-m_iStartTime)*m_dDeltaTim e : 0;

HField*= cos( (m_dAngularFrequency + m_dPhi*(time - m_dPulseOffset) )*(time - m_dPulseOffset)
 - m_dAngularFrequency*INVERSE_LIGHT_SPEED*getRefractiveIndex()*0.5*m_dDeltaSpace);

HField*= exp( - (time-m_dPulseOffset)*(time-m_dPulseOffset)
 / ( 2 * m_dPulseSigma * m_dPulseSigma ) );

return -HField;
}

double PulsedGaussianSource::getMagneticFieldZy( int j, int n )
{
//if we're a horizontal source then all Z Magnetic field components
//in the Y direction should not exist
if( m_iMinY == m_iMaxY )
 return 0;

double offset = m_dDeltaSpace * sqrt( ( j - m_iHalfY ) * ( j - m_iHalfY ) );

double HField = m_dMaxMagneticField
  *exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double time = (n-m_iStartTime) >= 0 ? (n-m_iStartTime)*m_dDeltaTime : 0;

HField*= cos( (m_dAngularFrequency + m_dPhi*(time - m_dPulseOffset) )*(time - m_dPulseOffset)
 - m_dAngularFrequency*INVERSE_LIGHT_SPEED*getRefractiveIndex()*0.5*m_dDeltaSpace);

HField*= exp( - (time-m_dPulseOffset)*(time-m_dPulseOffset)
 / ( 2 * m_dPulseSigma * m_dPulseSigma ) );

return -HField;
}

double PulsedGaussianSource::getElectricFieldX( int i, int n )
{
//if we're a vertical source then all X Electric field components
//should not exist
if( m_iMinX == m_iMaxX )
 return 0;

double offset = m_dDeltaSpace * sqrt( ( i - m_iHalfX ) * ( i - m_iHalfX ) );
double EField = m_dMaxElectricField
  *exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double time = (n-m_iStartTime) > 0 ? (n-m_iStartTime+0.5)*m_dDeltaTime : 0;

EField*= cos( (m_dAngularFrequency + m_dPhi*(time - m_dPulseOffset) )*(time - m_dPulseOffset) );

EField*= exp( - (time-m_dPulseOffset+INVERSE_LIGHT_SPEED*0.5*m_dDeltaSpace)*(time-
m_dPulseOffset+INVERSE_LIGHT_SPEED*0.5*m_dDeltaSpace)
 / ( 2 * m_dPulseSigma * m_dPulseSigma ) );


return EField;
}

double PulsedGaussianSource::getElectricFieldY( int j, int n )
{
//if we're a horizontal source then all Y Electric field components
//should not exist
if( m_iMinY == m_iMaxY )
 return 0;

double offset = m_dDeltaSpace * sqrt( ( j - m_iHalfY ) * ( j - m_iHalfY ) );
double EField = m_dMaxElectricField
  *exp(-0.5*offset*offset/m_dSigma/m_dSigma);
double time = (n-m_iStartTime) > 0 ? (n-m_iStartTime+0.5)*m_dDeltaTime : 0;

EField*= cos( (m_dAngularFrequency + m_dPhi*(t ime - m_dPulseOffset) )*(time - m_dPulseOffset) );
```

```
EField*= exp( - (time-m_dPulseOffset+INVERSE_LIGHT_SPEED*0.5*m_dDeltaSpace)*(time-
m_dPulseOffset+INVERSE_LIGHT_SPEED*0.5*m_dDeltaSpace)
 / ( 2 * m_dPulseSigma * m_dPulseSigma ) );


return EField;
}


double PulsedGaussianSource::getElectricFieldZ( int i, int k, int n )
{
return 0;
}

double PulsedGaussianSource::getElectricFieldTheta( int i, int k, int n )
{
return 0;
}

double PulsedGaussianSource::getElectricFieldR( int i, int k, int n )
{
return 0;
}

double PulsedGaussianSource::getMagneticFieldZ( int i, int k, int n )
{
return 0;
}

double PulsedGaussianSource::getMagneticFieldTheta( int i, int k, int n )
{
return 0;
}

double PulsedGaussianSource::getMagneticFieldR( int i, int k, int n )
{
return 0;
}

PulsedGaussianSource::~PulsedGaussianSource()
{
cerr << "Message: Deleting pulsed gaussian source\n";
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//PulsedSecondOrderSolver.cpp – second order accurate implementation of the FDTD algorithm
////////////////////////////////////////////////////////////////////////////////////////////////////
#include "PulsedSecondOrderSolver.h"
#include "SecondOrderSolver.h"
#include "Geometry.h"
#include "Options.h"
#include "iostream.h"
#include "SecondOrderSolver.h"
#include "PerfectlyMatchedLayerBoundaryCondition.h"
#include "GaussianSource.h"
#include "BitmapFieldWriter.h"
#include "TimeDomainFieldWriter.h"
#include "FieldIntegrator.h"
#include <math.h>
#include <cstdio>

PulsedSecondOrderSolver::PulsedSecondOrderSolver( Geometry *geom, Options *options )
:SecondOrderSolver( geom, options )
{
cout <<"Creating: PulsedSecondOrderSolver\n";
//TM Mode
m_DElectricFieldX = NULL;
m_DElectricFieldY = NULL;
m_DMagneticFieldZ = NULL;
```

```
//TE Mode
m_DMagneticFieldX = NULL;
m_DMagneticFieldY = NULL;
m_DElectricFieldZ = NULL;
}

PulsedSecondOrderSolver::~PulsedSecondOrderSolver()
{
cout <<"Deleting: PulsedSecondOrderSolver\n";
geometry = NULL;
}

int PulsedSecondOrderSolver::solve()
{
int rc = SUCCESS;

if( options->getSimulationMode() == TE_MODE )
{
cerr << "Message: Solving geometry for TE Mode\n";
rc = solveTEMode();
if ( rc != SUCCESS ){ return rc; }
}
if( options->getSimulationMode() == TM_MODE )
{
cerr << "Message: Solving geometry for TM Mode\n";
rc = solveTMMode();
if ( rc != SUCCESS ){ return rc; }
}
if( options->getSimulationMode() == TM_TE_MODE )
{
cerr << "Message: Solving geometry for TE and TM Mode\n";
rc |= solveTEMode();
rc |= solveTMMode();
if ( rc != SUCCESS ){ return rc; }
}
return rc;
}

int PulsedSecondOrderSolver::initialize()
{
int rc = FiniteDifferenceTimeDomainSolver::initialize();
if( rc != SUCCESS )
 return rc;

//check the m_iTime and space deltas...
cerr << "Message: Initializing Second Order Solver\n";
if( geometry->getDeltaTime() > geometry->getDeltaSpace()/( LIGHT_SPEED )/sqrt(2.0) )
{
cerr << "Warning: Upper bound on lattice spacing is not met!\n";
cerr << "Message: m_dDeltaSpace = " << geometry->getDeltaSpace() << "\n";
cerr << "Message: m_dDeltaTime = " << geometry->getDeltaTime() << "\n";
rc = NUMERICAL_STABILITY_WARNING;
}
return rc;
}


int PulsedSecondOrderSolver::solveTEMode()
{
int rc = SUCCESS;
int i,j,n,k;
m_iWidth = geometry->getGridWidth();
m_iHeight = geometry->getGridHeight();
m_iTime = geometry->getGridTime();
TimeDomainFieldWriter *tdfw = new TimeDomainFieldWriter( geometry, options, "Ez" );
FieldIntegrator *fi = new FieldI ntegrator( geometry, options );

m_dDeltaTime = geometry->getDeltaTime();
m_dDeltaSpace = geometry->getDeltaSpace();
```

```
bc = PerfectlyMatchedLayerBoundaryCondition::getBoundaryCondition();

PML_width = bc->getWidth();
PML_top = m_iHeight - PML_width;
PML_bottom = PML_width;
PML_right = m_iWidth - PML_width;
PML_left = PML_width;


//TM according to Taflove
cerr << "\nMessage: Solving for TE Mode!\n";

cerr << "Message: Grid size = " << geometry->getGridWidth() << " x " <<
 geometry->getGridHeight() << "\n";
cerr << "Message: Space step = " << geometry->getDeltaSpace() << " meters\n";
cerr << "Message: m_iTime step = " << geometry->getDeltaTime() << " seconds\n";
cerr << "Message: Simulation m_iTime = " << geometry->getRunningTime() << " seconds\n";
cerr << "Message: Cells per minimum lambda = " << geometry->getMinimumLambda()/geometry->getDeltaSpace() << "\n";

m_DElectricFieldZ = new double*[ m_iWidth ];
m_DMagneticFieldX = new double*[ m_iWidth ];
m_DMagneticFieldY = new double*[ m_iWidth ];
m_DElectricFieldPrevZ  = new double*[ m_iWidth ];

cerr << "Dispersive Materials exist. Creating extra arrays...\n";
m_DElectricFieldDisplacementZPrevPrev = new double*[ m_iWidth ];
m_DElectricFieldDisplacementZPrev = new double*[ m_iWidth ];
m_DElectricFieldDisplacementZ = new double*[ m_iWidth ];

m_DLinearPolarization1PrevPrev = new double*[ m_iWidth ];
m_DLinearPolarization1Prev = new double*[ m_iWidth ];

m_DLinearPolarization2PrevPrev = new double*[ m_iWidth ];
m_DLinearPolarization2Prev = new double*[ m_iWidth ];

m_DLinearPolarization3PrevPrev = new double*[ m_iWidth ];
m_DLinearPolarization3Prev = new double*[ m_iWidth ];

//extra arrays for boundary condition
m_DElectricFieldZX = new double*[ m_iWidth ];
m_DElectricFieldZY = new double*[ m_iWidth ];

//initialize the actual fields
for( i=0; i<m_iWidth; i++ )
{
 m_DElectricFieldDisplacementZPrevPrev[i] = new double[ m_iHeight ];
 m_DElectricFieldDisplacementZPrev[i] = new double[ m_iHeight ];
 m_DElectricFieldDisplacementZ[i] = new double[ m_iHeight ];

 m_DLinearPolarization1PrevPrev[i] = new double[ m_iHeight ];
 m_DLinearPolarization1Prev[i] = new double[ m_iHeight ];

 m_DLinearPolarization2PrevPrev[i] = new double[ m_iHeight ];
 m_DLinearPolarization2Prev[i] = new double[ m_iHeight ];

 m_DLinearPolarization3PrevPrev[i] = new double[ m_iHeight ];
 m_DLinearPolarization3Prev[i] = new double[ m_iHeight ];


 m_DElectricFieldZ[i] = new double [ m_iHeight ];
 m_DMagneticFieldX[i] = new double [ m_iHeight ];
 m_DMagneticFieldY[i] = new double [ m_iHeight ];
 m_DElectricFieldPrevZ[i] = new double [ m_iHeight ];

 m_DElectricFieldZX[i] = new double[ m_iHeight ];
 m_DElectricFieldZY[i] = new double[ m_iHeight ];

 for( j=0; j<m_iHeight; j++ )
 {
```

```
 m_DElectricFieldZ[i][j] = 0;
 m_DMagneticFieldX[i][j] = 0;
 m_DMagneticFieldY[i][j] = 0;
 m_DElectricFieldPrevZ[i][j] = 0;

 m_DElectricFieldDisplacementZPrevPrev[i][j] = 0;
 m_DElectricFieldDisplacementZPrev[i][j] = 0;
 m_DElectricFieldD isplacementZ[i][j] = 0;

 m_DLinearPolarization1PrevPrev[i][j] = 0;
 m_DLinearPolarization1Prev[i][j] = 0;

 m_DLinearPolarization2PrevPrev[i][j] = 0;
 m_DLinearPolarization2Prev[i][j] = 0;

 m_DLinearPolarization3PrevPrev[i][j] = 0;
 m_DLinearPolarization3Prev[i][j] = 0;


 m_DElectricFieldZX[i][j] = 0;
 m_DElectricFieldZY[i][j] = 0;
 }
}

//loop through the m_iTime steps...
for( n=0; n<m_iTime; n++ )
{
 //loop through the space region solving for H
 cerr << "Message: " << double(n)/double(m_iTime)*100 << " percent complete\n";

 for( k=0; k<m_iWidth*m_iHeight; k++ )
 {
  processPointTE(i,j,n);

  i++;
  if(i%m_iWidth == 0)
  {
   i=0;
   j++;
   if( j%m_iHeight == 0 )
    j=0;

  }
 } // index loop

 if( options->getSamplingPointCount() > 0 )
 {
  tdfw->addPoint( m_DElectricFieldZ, n );
  fi->addPoint( m_DElectricFieldZ, n );
 }

 //output to a file if necessary
 if( options->getFrameSpacing() != 0 && (n+1)%options->getFrameSpacing() == 0 )
 {
  char filename[512];
  //output the E field at this m_iTime step if required
  sprintf(filename,"%s_%s_%s_%4d",options->getSimulationName(), TE_MODE_TOKEN, "Ez",(n+1)/options-
>getFrameSpacing());

  rc = BitmapFieldWriter::write( filename, m_DElectricFieldZ, m_iWidth, m_iHeight, options );
 }


 if( n == m_iTime-1 && options->isQueryContinue() )
 {
  char option;
  cerr << "Message: Would you like to continue processing (y/n)?";
  cin >> option;
  int exfTime=0;
  while( option != 'y' && option != 'n' )
```

```
 {
 cerr << "Please input y/n!";
 cin >> option;
 }
 if( option == 'y' )
 {
 cerr << "Please specify the number of extended m_iTime steps: ";
 cin >> extTime;
 int oldTime = m_iTime;
 m_iTime +=extTime;
 geometry->setGridTime( m_iTime );
 tdfw->reset( oldTime );
 fi->reset( oldTime );
 }
 }
} // n loop

if( options->getFrameSpacing() == 0 )
{
 char filename[512];
 //output the E field at this m_iTime step if required
 sprintf(filename,"%s_%s_%s_%4d",options->getSimulationName(), TE_MODE_TOKEN, "Ez", 0 );
 rc = BitmapFieldWriter::write( filename, m_DElectricFieldZ, m_iWidth, m_iHeight, options );
}

if( options->getSamplingPointCount() > 0 )
{
 tdfw->write();
 fi->write();
}

//delete the arrays
for( i=0; i<m_iWidth; i++ )
{
 delete [] m_DElectricFieldZ[i];
 delete [] m_DMagneticFieldX[i];
 delete [] m_DMagneticFieldY[i];
 delete [] m_DElectricFieldZX[i];
 delete [] m_DElectricFieldZY[i];
}
delete [] m_DElectricFieldZ;
delete [] m_DMagneticFieldX;
delete [] m_DMagneticFieldY;
delete [] m_DElectricFieldZX;
delete [] m_DElectricFieldZY;
delete tdfw;
delete fi;

cerr << "Message: Finished TE Mode\n\n";
return rc;
}


int PulsedSecondOrderSolver::solveTMMode()
{
int rc = SUCCESS;
int i,j,n,k;
m_iWidth = geometry->getGridWidth();
m_iHeight = geometry->getGridHeight();
m_iTime = geometry->getGridTime();

m_dDeltaTime = geometry->getDeltaTime();
m_dDeltaSpace = geometry->getDeltaSpace();

TimeDomainFieldWriter *t dfwx = new TimeDomainFieldWriter( geometry, options, "Ex" );
TimeDomainFieldWriter *tdfwy = new TimeDomainFieldWriter( geometry, options, "Ey" );
FieldIntegrator *fi = new FieldIntegrator( geometry, options );

bc = PerfectlyMatchedLayerBoundaryCondition::getBoundaryCondition();
```

```
PML_width=bc->getWidth();
PML_top = m_iHeight - PML_width;
PML_bottom = PML_width-1;
PML_right = m_iWidth - PML_width;
PML_left = PML_width-1;


//TE according to Taflove
cerr << "\nMessage: Solving for TM Mode!\n";

cerr << "Message: Grid size = " << geometry->getGridWidth() << " x " <<
 geometry->getGridHeight() << "\n";
cerr << "Message: Space step = " << geometry->getDeltaSpace() << " meters\n";
cerr << "Message: m_iTime step = " << geometry->getDeltaTime() << " seconds\n";
cerr << "Message: Simulation m_iTime = " << geometry->getRunningTime() << " seconds\n";
cerr << "Message: Cells per minimum lambda = " << geometry->getMinimumLambda()/geometry->getDeltaSpace() << "\n";


m_DMagneticFieldZ = new double*[ m_iWidth ];
m_DElectricFieldX = new double*[ m_iWidth ];
m_DElectricFieldY = new double*[ m_iWidth ];
m_DTotalElectricField = new double*[m_iWidth];
m_DCurrentDensityX = new double*[ m_iWidth ];
m_DCurrentDensityY = new double*[ m_iWidth ];


m_DElectricFieldPrevX = new double*[ m_iWidth ];
m_DElectricFieldPrevY = new double*[ m_iWidth ];

m_DElectricFieldDisplacementXPrevPrev = new double*[ m_iWidth ];
m_DElectricFieldDisplacementXPrev = new double*[ m_iWidth ];
m_DElectricFieldDisplacementX = new double*[ m_iWidth ];

m_DElectricFieldDisplacementYPrevPrev = new double*[ m_iWidth ];
m_DElectricFieldDisplacementYPrev = new double*[ m_iWidth ];
m_DElectricFieldDisplacementY = new double*[ m_iWidth ];

m_DLinearPolarization1XPrevPrev = new double*[ m_iWidth ];
m_DLinearPolarization1XPrev = new double*[ m_iWidth ];

m_DLinearPolarization1YPrevPrev = new double*[ m_iWidth ];
m_DLinearPolarization1YPrev = new double*[ m_iWidth ];

m_DLinearPolarization2XPrevPrev = new double*[ m_iWidth ];
m_DLinearPolarization2XPrev = new double*[ m_iWidth ];

m_DLinearPolarization2YPrevPrev = new double*[ m_iWidth ];
m_DLinearPolarization2YPrev = new double*[ m_iWidth ];

m_DLinearPolarization3XPrevPrev = new double*[ m_iWidth ];
m_DLinearPolarization3XPrev = new double*[ m_iWidth ];

m_DLinearPolarization3YPrevPrev = new double*[ m_iWidth ];
m_DLinearPolarization3YPrev = new double*[ m_iWidth ];


//extra arrays for boundary condition
m_DMagneticFieldZX = new double*[ m_iWidth ];
m_DMagneticFieldZY = new double*[ m_iWidth ];

//initialize the actual fields
for( i=0; i<m_iWidth; i++ )
{
 m_DElectricFieldDisplacementXPrevPrev[i] = new double[ m_iHeight ];
 m_DElectricFieldDisplacementXPrev[i] = new double[ m_iHeight ];
 m_DElectricFieldDisplacementX[i] = new double[ m_iHeight ];

 m_DElectricFieldDisplacementYPrevPrev[i] = new double[ m_iHeight ];
 m_DElectricFieldDisplacementYPrev[i] = new double[ m_iHeight ];
 m_DElectricFieldDisplacementY[i] = new double[ m_iHeight ];
```

```
m_DLinearPolarization1XPrevPrev[i] = new double[ m_iHeight ];
m_DLinearPolarization1XPrev[i] = new double[ m_iHeight ];

m_DLinearPolarization1YPrevPrev[i] = new double[ m_iHeight ];
m_DLinearPolarization1YPrev[i] = new double[ m_iHeight ];

m_DLinearPolarization2XPrevPrev[i] = new double[ m_iHeight ];
m_DLinearPolarization2XPrev[i] = new double[ m_iHeight ];

m_DLinearPolarization2YPrevPrev[i] = new double[ m_iHeight ];
m_DLinearPolarization2YPrev[i] = new double[ m_iHeight ];

m_DLinearPolarization3XPrevPrev[i] = new double[ m_iHeight ];
m_DLinearPolarization3XPrev[i] = new double[ m_iHeight ];

m_DLinearPolarization3YPrevPrev[i] = new double[ m_iHeight ];
m_DLinearPolarization3YPrev[i] = new double[ m_iHeight ];

m_DElectricFieldPrevX[i] = new double [ m_iHeight ];
m_DElectricFieldPrevY[i] = new double [ m_iHeight ];


m_DMagneticFieldZ[i] = new double [ m_iHeight ];
m_DElectricFieldX[i] = new double [ m_iHeight ];
m_DElectricFieldY[i] = new double [ m_iHeight ];
m_DTotalElectricField[i] = new double [m_iHeight];

m_DCurrentDensityX[i] = new double[ m_iHeight ];
m_DCurrentDensityY[i] = new double[ m_iHeight ];

m_DMagneticFieldZX[i] = new double[ m_iHeight ];
m_DMagneticFieldZY[i] = new double[ m_iHeight ];

for( j=0; j<m_iHeight; j++ )
{
m_DMagneticFieldZ[i][j] = 0; //add on the incident beam at t=0
m_DElectricFieldX[i][j] = 0;
m_DElectricFieldY[i][j] = 0;
m_DTotalElectricField [i][j] = 0;
m_DCurrentDensityX[i][j] = 0;
m_DCurrentDensityY[i][j] = 0;

m_DElectricFieldPrevX[i][j] = 0;
m_DElectricFieldPrevY[i][j] = 0;

m_DElectricFieldDisplacementXPrevPrev[i][j] = 0;
m_DElectricFieldDisplacementXPrev[i][j] = 0;
m_DElectricFieldDisplacementX[i][j] = 0;

m_DElectricFieldDisplacementYPrevPrev[i][j] = 0;
m_DElectricFieldDisplacementYPrev[i][j] = 0;
m_DElectricFieldDisplacementY[i][j] = 0;

m_DLinearPolarization1XPrevPrev[i][j] = 0;
m_DLinearPolarization1XPrev[i][j] = 0;

m_DLinearPolarization1YPrevPrev[i][j] = 0;
m_DLinearPolarization1YPrev[i][j] = 0;

m_DLinearPolarization2XPrevPrev[i][j] = 0;
m_DLinearPolarization2XPrev[i][j] = 0;

m_DLinearPolarization2YPrevPrev[i][j] = 0;
m_DLinearPolarization2YPrev[i][j] = 0;

m_DLinearPolarization3XPrevPrev[i][j] = 0;
m_DLinearPolarization3XPrev[i][j] = 0;

m_DLinearPolarization3YPrevPrev[i][j] = 0;
```

```
 m_DLinearPolarization3YPrev[i][j] = 0;


 m_DMagneticFieldZX[i][j] = 0;
 m_DMagneticFieldZY[i][j] = 0;
 }
}

//loop through the m_iTime steps...
for( n=0; n<m_iTime; n++ )
{

 //loop through the space region
 cerr << "Message: " << double(n)/double(m_iTime)*100 << " percent complete\n";
 i=j=0;

 for( i=0; i<m_iWidth; i++ )
  for( j=0; j<m_iHeight; j++ )
   processPointTM(i,j,n);
   // index loop

 //output to a file if necessary
 if( options->getFrameSpacing() != 0 && (n+1)%options->getFrameSpacing() == 0 )
 {
  char filename[512];

  //output the E field at this m_iTime step if required
  sprintf(filename,"%s_%s_%s_%4d",options->getSimulationName(), TM_MODE_TOKEN,"Et",(n+1)/options-
>getFrameSpacing());

  rc = BitmapFieldWriter::write( filename, m_DTotalElectricField, m_iWidth, m_iHeight, options );

  //output the E field at this m_iTime step if required
  sprintf(filename,"%s_%s_%s_%4d",options->getSimulationName(), TM_MODE_TOKEN,"Hz",(n+1)/options-
>getFrameSpacing());

  rc = BitmapFieldWriter::write( filename, m_DMagneticFieldZ, m_iWidth, m_iHeight, options, true );

 }

 if( options->getSamplingPointCount() > 0 )
 {
  tdfwx->addPoint( m_DElectricFieldX, n );
  tdfwy->addPoint( m_DElectricFieldY, n );
  fi->addPoint( m_DTotalElectricField, n );
 }

 if( n == m_iTime-1  && options->isQueryContinue() )
 {
  char option;
  cerr << "Message: Would you like to continue processing (y/n)?";
  cin >> option;
  int extTime=0;
  while( option != 'y' && option != 'n' )
  {
   cerr << "Please input y/n!";
   cin >> option;
  }
  if( option == 'y' )
  {
   cerr << "Please specify the number of extended m_iTime steps: ";
   cin >> extTime;
   int oldTime = m_iTime;
   m_iTime +=extTime;
   geometry->setGridTime( m_iTime );
   tdfwx->reset( oldTime );
   fi->reset( oldTime );
  }
 }
```

```
} // n loop


if( options->getFrameSpacing() == 0 )
{
 char filename[512];
 //output the E field at this m_iTime step if required
 sprintf(filename,"%s_%s_%s_%4d",options->getSimulationName(),TM_MODE_TOKEN,"Ez",0 );
 rc = BitmapFieldWriter::write( filename, m_DTotalElectricField, m_iWidth, m_iHeight, options );
}

if( options->getSamplingPointCount() > 0 )
{
 tdfwy->write();
 tdfwx->write();
 fi->write();
}

//delete the arrays
for( i=0; i<m_iWidth; i++ )
{
 delete [] m_DTotalElectricField[i];
 delete [] m_DMagneticFieldZ[i];
 delete [] m_DElectricFieldX[i];
 delete [] m_DElectricFieldY[i];
 delete [] m_DMagneticFieldZX [i];
 delete [] m_DMagneticFieldZY [i];

}
delete [] m_DTotalElectricField;
delete [] m_DMagneticFieldZ;
delete [] m_DElectricFieldX;
delete [] m_DElectricFieldY;
delete [] m_DMagneticFieldZX;
delete [] m_DMagneticFieldZY;
delete tdfwx;
delete tdfwy;
delete fi;
cerr << "Message: Finished TM Mode\n\n";
return rc;
}

int PulsedSecondOrderSolver::processPointTM(int i, int j, intn)
{
 Material* mat = geometry->getMaterial(i,j);

 double linearPolarization1X;
 double linearPolarization1Y;
 double linearPolarization2X;
 double linearPolarization2Y;
 double linearPolarization3X;
 double linearPolarization3Y;
 double denominator;
 double Z1X,Z1Y,Z2X,Z2Y,Z3X,Z3Y;
 double a1,a2,a3,c1,c2,c3,g1,g2,g3;
 double h1,h2,h3,h4;

 double tempEFieldX = m_DElectricFieldX[i][j];
 double tempEFieldY = m_DElectricFieldY[i][j];

 if( mat->isBoundary() )
 {
  linearPolarization1X = 0;
  linearPolarization1Y = 0;
  linearPolarization2X = 0;
  linearPolarization2Y = 0;
  linearPolarization3X = 0;
  linearPolarization3Y = 0;
  double mAx = bc->getMagneticCoefficientA(),
```

```
mBx = bc->getMagneticCoefficientB(),
mAy = bc->getMagneticCoefficientA(),
mBy = bc->getMagneticCoefficientB();
double eAx = bc->getElectricCoefficientA(),
eBx = bc->getElectricCoefficientB(),
eAy = bc->getElectricCoefficientA(),
eBy = bc->getElectricCoefficientB();

//were in the left layer
if( i < PML_width )
{
//were in the bottom left corner
if( j < PML_width )
{
mAy = bc->getMagneticFieldExponentialCoefficientA( PML_width - j - 1 );
mBy = bc->getMagneticFieldExponentialCoefficientB( PML_width- j - 1 );
mAx = bc->getMagneticFieldExponentialCoefficientA( PML_width - i - 1 );
mBx = bc->getMagneticFieldExponentialCoefficientB( PML_width- i - 1 );

eAx = bc->getElectricFieldExponentialCoefficientA( PML_width - j - 1 );
eBx = bc->getElectricFieldExponentialCoefficientB( PML_width - j - 1 );
eAy = bc->getElectricFieldExponentialCoefficientA( PML_width - i - 1 );
eBy = bc->getElectricFieldExponentialCoefficientB( PML_width - i - 1 );
}
//were in the top left corner
else if( j >= m_iHeight- PML_width )
{
mAx = bc->getMagneticFieldExponentialCoefficientA( PML_width - i - 1 );
mBx = bc->getMagneticFieldExponentialCoefficientB( PML_width- i - 1 );
mAy = bc->getMagneticFieldExponentialCoefficientA( j-m_iHeight+PML_width ) ;
mBy = bc->getMagneticFieldExponentialCoefficientB( j-m_iHeight+PML_width );

eAy = bc->getElectricFieldExponentialCoefficientA( PML_width - i - 1 );
eBy = bc->getElectricFieldExponentialCoefficientB( PML_width - i - 1 );
eAx = bc->getElectricFieldExponentialCoefficientA( j-m_iHeight+PML_width );
eBx = bc->getElectricFieldExponentialCoefficientB( j-m_iHeight+PML_width );
}
//were in the left layer only
else
{
mAx = bc->getMagneticFieldExponentialCoefficientA( PML_width - i - 1 );
mBx = bc->getMagneticFieldExponentialCoefficientB( PML_width- i - 1 );
mAy = bc->getMagneticCoefficientA();
mBy = bc->getMagneticCoefficientB();

eAy = bc->getElectricFieldExponentialCoefficientA( PML_width - i - 1 );
eBy = bc->getElectricFieldExponentialCoefficientB( PML_width - i - 1 );
eAx = bc->getElectricCoefficientA();
eBx = bc->getElectricCoefficientB();
}
}
else if( i >= m_iWidth - PML_width )
{
//were in the bottom right corner
if( j < PML_width )
{
mAx = bc->getMagneticFieldExponentialCoefficientA( i-m_iWidth+PML_width );
mBx = bc->getMagneticFieldExponentialCoefficientB( i-m_iWidth+PML_width );
mAy = bc->getMagneticFieldExponentialCoefficientA( PML_width - j - 1 );
mBy = bc->getMagneticFieldExponentialCoefficientB( PML_width- j - 1 );

eAy = bc->getElectricFieldExponentialCoefficientA( i-m_iWidth+PML_width );
eBy = bc->getElectricFieldExponentialCoefficientB( i-m_iWidth+PML_width );
eAx = bc->getElectricFieldExponentialCoefficientA( PML_width - j - 1 );
eBx = bc->getElectricFieldExponentialCoefficientB( PML_width - j - 1 );
}
//were in the top right corner
else if( j >= m_iHeight - PML_width )
{
mAx = bc->getMagneticFieldExponentialCoefficientA( i-m_iWidth+PML_width );
```

```
 mBx = bc->getMagneticFieldExponentialCoefficientB( i-m_iWidth+PML_width );
 mAy = bc->getMagneticFieldExponentialCoefficientA( j-m_iHeight+PML_width );
 mBy = bc->getMagneticFieldExponentialCoefficientB( j-m_iHeight+PML_width );

 eAy = bc->getElectricFieldExponentialCoefficientA( i-m_iWidth+PML_width );
 eBy = bc->getElectricFieldExponentialCoefficientB( i-m_iWidth+PML_width );
 eAx = bc->getElectricFieldExponentialCoefficientA( j-m_iHeight+PML_width );
 eBx = bc->getElectricFieldExponentialCoefficientB( j-m_iHeight+PML_width );
 }
//were in the right layer only
 else
 {
 mAx = bc->getMagneticFieldExponentialCoefficientA( i-m_iWidth+PML_width );
 mBx = bc->getMagneticFieldExponentialCoefficientB( i-m_iWidth+PML_width );
 mAy = bc->getMagneticCoefficientA();
 mBy = bc->getMagneticCoefficientB();

 eAy = bc->getElectricFieldExponentialCoefficientA( i-m_iWidth+PML_width );
 eBy = bc->getElectricFieldExponentialCoefficientB( i-m_iWidth+PML_width );
 eAx = bc->getElectricCoefficientA();
 eBx = bc->getElectricCoefficientB();
 }
}
//were in the bottom layer only
else if( j < PML_width )
{
 mAx = bc->getMagneticCoefficientA();
 mBx = bc->getMagneticCoefficientB();
 mAy = bc->getMagneticFieldExponentialCoefficientA( PML_width - j - 1 );
 mBy = bc->getMagneticFieldExponentialCoefficientB( PML_width - j - 1 );

 eAy = bc->getElectricCoefficientA();
 eBy = bc->getElectricCoefficientB();
 eAx = bc->getElectricFieldExponentialCoefficientA( PML_width - j - 1 );
 eBx = bc->getElectricFieldExponentialCoefficientB( PML_width - j - 1 );
}
//were in the top layer only
else if( j >= m_iHeight - PML_width )
{
 mAx = bc->getMagneticCoefficientA();
 mBx = bc->getMagneticCoefficientB();
 mAy = bc->getMagneticFieldExponentialCoefficientA( j-m_iHeight+PML_width );
 mBy = bc->getMagneticFieldExponentialCoefficientB( j-m_iHeight+PML_width );

 eAy = bc->getElectricCoefficientA();
 eBy = bc->getElectricCoefficientB();
 eAx = bc->getElectricFieldExponentialCoefficientA( j-m_iHeight+PML_width );
 eBx = bc->getElectricFieldExponentialCoefficientB( j-m_iHeight+PML_width );
}
else
{
 cerr << "Error: Enterened erroneous region!\n";
}

double Hzx = j==m_iHeight-1?0.0:m_DMagneticFieldZ[i][j+1];
double Hzy = i==m_iWidth-1?0.0:m_DMagneticFieldZ[i+1][j];

m_DElectricFieldX[i][j] = eAx * m_DElectricFieldX[i][j]
 - eBx * ( m_DMagneticFieldZ[i][j]
 - Hzx );

m_DElectricFieldY[i][j] = eAy * m_DElectricFieldY[i][j]
 - eBy * ( Hzy
 - m_DMagneticFieldZ[i][j] );

double Ey = i==0?0.0:m_DElectricFieldY[i-1][j];
double Ex = j==0?0.0:m_DElectricFieldX[i][j-1];

m_DMagneticFieldZX[i][j] =
 mAx * m_DMagneticFieldZX[i][j]
```

```
 - mBx * ( m_DElectricFieldY[i][j]
 - Ey );

m_DMagneticFieldZY[i][j] =
 mAy * m_DMagneticFieldZY[i][j]
 - mBy * ( - m_DElectricFieldX[i][j]
 + Ex );

m_DMagneticFieldZ[i][j] = m_DMagneticFieldZX[i][j] + m_DMagneticFieldZY[i][j];
}
else
{
double A =mat->getElectricCoefficientA();
double B = mat->getElectricCoefficientB();
double Bo = geometry->getDeltaTime()/geometry->getDeltaSpace();

double Hzx = j==m_iHeight-1?0.0:m_DMagneticFieldZ[i][j+1];
double Hzy = i==m_iWidth-1?0.0:m_DMagneticFieldZ[i+1][j];

double HzIncX = 0;
double HzIncY = 0;

Source *source = NULL;

if( ( source = geometry->isPointLeftOfSource( i, j ) ) != NULL )
{
 HzIncY = source->getMagneticFieldZy( j, n );

}
else if( ( source = geometry->isPointBelowSource( i, j ) ) != NULL )
{
 HzIncX = source->getMagneticFieldZx( i, n );
}

m_DElectricFieldDisplacementX[i][j] = m_DElectricFieldDisplacementX[i][j]
 + Bo * ( Hzx
 - m_DMagneticFieldZ[i][j]
 - HzIncX);

m_DElectricFieldDisplacementY[i][j] = m_DElectricFieldDisplacementY[i][j]
 + Bo * ( m_DMagneticFieldZ[i][j]
 - Hzy
 + HzIncY);

if( mat->isDispersive() )
{
 if( mat->isMetal() )
 {
 h1 = mat->getCoefficientH1();
 h2 = mat->getCoefficientH2();
 h3 = mat->getCoefficientH3();
 h4 = mat->getCoefficientH4();

 m_DElectricFieldX[i][j] = ( h1*m_DElectricFieldDisplacementX[i][j] - 4*m_DElectricFieldDisplacementXPrev[i][j]
  + h2*m_DElectricFieldDisplacementXPrevPrev[i][j] + 4*mat->getElectricPermittivity()*m_DElectricFieldX[i][j]
  - h3*m_DElectricFieldPrevX[i][j] )
  / ( h4 );

 m_DElectricFieldY[i][j] = ( h1*m_DElectricFieldDisplacementY[i][j] - 4*m_DElectricFieldDisplacementYPrev[i][j]
  + h2*m_DElectricFieldDisplacementYPrevPrev[i][j] + 4*mat->getElectricPermittivity()*m_DElectricFieldY[i][j]
  - h3*m_DElectricFieldPrevY[i][j] )
  / ( h4 );
 }
 else if( mat->isConductive() )
 {
  double delta=geometry->getDeltaSpace();
  double delta_time=geometry->getDeltaTime();

  m_DElectricFieldX[i][j] = m_DElectricFieldX[i][j]
  + B * ( Hzx - m_DMagneticFieldZ[i][j]
```

```
    - delta*m_DCurrentDensityX[i][j]
    );

   m_DElectricFieldY[i][j] = m_DElectricFieldY[i][j]
    + B * ( m_DMagneticFieldZ[i][j] -  Hzy
    - delta*m_DCurrentDensityY[i][j]
    );


    double epsilon_eff=mat->getElectricPermittivity();
    double alpha=mat->getScatteringFrequency();
    double beta=EPSILON_NOT*mat->getPlasmaFrequency()*mat->getPlasmaFrequency();

   m_DCurrentDensityX[i][j]=(2+alpha*delta_time)/(2-alpha*delta_time)*m_DCurrentDensityX[i][j]+delta_time*beta/(2-
alpha*delta_time)*m_DElectricFieldX[i][j];
   m_DCurrentDensityY[i][j]=(2+alpha*delta_time)/(2-alpha*delta_time)*m_DCurrentDensityY[i][j]+delta_time*beta/(2-
alpha*delta_time)*m_DElectricFieldY[i][j];
   }
   else
   {
   a1 = mat->getCoefficientA1();
   a2 = mat->getCoefficientA2();
   a3 = mat->getCoefficientA3();
   c1 = mat->getCoefficientW1();
   c2 = mat->getCoefficientW2();
   c3 = mat->getCoefficientW3();
   g1 = mat->getCoefficientG1();
   g2 = mat->getCoefficientG2();
   g3 = mat->getCoefficientG3();
   denominator = -a1*a2*a3 + a1*c2*c3 + c2*c1*a3 - 2*c1*c2*c3 + c3*c1*a2;

   //X direction
   Z1X = c1*( m_DElectricFieldDisplacementX[i][j] + m_DElectricFieldDisplacementXPrevPrev[i][j] -
    m_DLinearPolarization2XPrevPrev[i][j] - m_DLinearPolarization3XPrevPrev[i][j] ) +
    4*m_DLinearPolarization1XPrev[i][j] + g1*m_DLinearPolarization1XPrevPrev[i][j];

   Z2X = c2*( m_DElectricFieldDisplacementX[i][j] + m_DElectricFieldDisplacementXPrevPrev[i][j] -
    m_DLinearPolarization1XPrevPrev[i][j] - m_DLinearPolarization3XPrevPrev[i][j] ) +
    4*m_DLinearPolarization2XPrev[i][j] + g2*m_DLinearPolarization2XPrevPrev[i][j];

   Z3X = c3*( m_DElectricFieldDisplacementX[i][j] + m_DElectricFieldDisplacementXPrevPrev[i][j] -
    m_DLinearPolarization1XPrevPrev[i][j] - m_DLinearPolarization2XPrevPrev[i][j] ) +
    4*m_DLinearPolarization3XPrev[i][j] + g3*m_DLinearPolarization3XPrevPrev[i][j];
   //Y direction
   Z1Y = c1*( m_DElectricFieldDisplacementY[i][j] + m_DElectricFieldDisplacementYPrevPrev[i][j] -
    m_DLinearPolarization2YPrevPrev[i][j] - m_DLinearPolarization3YPrevPrev[i][j] ) +
    4*m_DLinearPolarization1YPrev[i][j] + g1*m_DLinearPolarization1YPrevPrev[i][j];

   Z2Y = c2*( m_DElectricFieldDisplacementY[i][j] + m_DElectricFieldDisplacementYPrevPrev[i][j] -
    m_DLinearPolarization1YPrevPrev[i][j] - m_DLinearPolarization3YPrevPrev[i][j] ) +
    4*m_DLinearPolarization2YPrev[i][j] + g2*m_DLinearPolarization2YPrevPrev[i][j];

   Z3Y = c3*( m_DElectricFieldDisplacementY[i][j] + m_DElectricFieldDisplacementYPrevPrev[i][j] -
    m_DLinearPolarization1YPrevPrev[i][j] - m_DLinearPolarization2YPrevPrev[i][j] ) +
    4*m_DLinearPolarization3YPrev[i][j] + g3*m_DLinearPolarization3YPrevPrev[i][j];
   //X direction
   linearPolarization1X = Z1X*(-a2*a3+c2*c3)/denominator-c1*(-a3+c3)/denominator*Z2X-c1*(c2-a2)/denominator*Z3X;

   linearPolarization2X = -c2*(-a3+c3)/denominator*Z1X-Z2X*(a1*a3-c1*c3)/denominator+c2*(a1-c1)/denominator*Z3X;

   linearPolarization3X = -c3*(c2-a2)/denominator*Z1X+c3*(a1-c1)/denominator*Z2X-Z3X*(a1*a2-c1*c2)/denominator;
   m_DElectricFieldX[i][j] = (1.0/EPSILON_NOT) * ( m_DElectricFieldDisplacementX[i][j] - linearPolarization1X -
linearPolarization2X - linearPolarization3X);
   //Y direction
   linearPolarization1Y = Z1Y*(-a2*a3+c2*c3)/denominator-c1*(-a3+c3)/denominator*Z2Y-c1*(c2-a2)/denominator*Z3Y;

   linearPolarization2Y = -c2*(-a3+c3)/denominator*Z1Y-Z2Y*(a1*a3-c1*c3)/denominator+c2*(a1-c1)/denominator*Z3Y;

   linearPolarization3Y = -c3*(c2-a2)/denominator*Z1Y+c3*(a1-c1)/denominator*Z2Y-Z3Y*(a1*a2-c1*c2)/denominator;
```

```
  m_DElectricFieldY[i][j] = (1.0/mat->getElectricPermittivity()) * ( m_DElectricFieldDisplacementY[i][j] - linearPolarization1Y -
linearPolarization2Y - linearPolarization3Y);
 }
 }
 else
 {
 A = mat->getElectricCoefficientA();
 B = mat->getElectricCoefficientB();
 linearPolarization1X = 0;
 linearPolarization2X = 0;
 linearPolarization3X = 0;
 linearPolarization1Y = 0;
 linearPolarization2Y = 0;
 linearPolarization3Y = 0;
 m_DElectricFieldX[i][j] = A * m_DElectricFieldX[i][j]
  + B * ( Hzx
  - m_DMagneticFieldZ[i][j]
  - HzIncX);

 m_DElectricFieldY[i][j] = A * m_DElectricFieldY[i][j]
  + B * ( m_DMagneticFieldZ[i][j]
  - Hzy
  + HzIncY);
 }

 m_DElectricFieldPrevX[i][j] = tempEFieldX;
 m_DElectricFieldPrevY[i][j] = tempEFieldY;

 m_DLinearPolarizat ion1XPrevPrev[i][j]=m_DLinearPolarization1XPrev[i][j];
 m_DLinearPolarization2XPrevPrev[i][j]=m_DLinearPolarization2XPrev[i][j];
 m_DLinearPolarization3XPrevPrev[i][j]=m_DLinearPolarization3XPrev[i][j];

 m_DLinearPolarization1YPrevPrev[i][j]=m_DLinearPolarization1YPrev[i][j];
 m_DLinearPolarization2YPrevPrev[i][j]=m_DLinearPolarization2YPrev[i][j];
 m_DLinearPolarization3YPrevPrev[i][j]=m_DLinearPolarization3YPrev[i][j];

 m_DLinearPolarization1XPrev[i][j] = linearPolarization1X;
 m_DLinearPolarization2XPrev[i][j] = linearPolarization2X;
 m_DLinearPolarization3XPrev[i][j] = linearPolarization3X;
 m_DElectricFieldDisplacementXPrevPrev[i][j] = m_DElectricFieldDisplacementXPrev[i][j];
 m_DElectricFieldDisplacementXPrev[i][j] = m_DElectricFieldDisplacementX[i][j];

 m_DLinearPolarization1YPrev[i][j] = linearPolarization1Y;
 m_DLinearPolarization2YPrev[i][j] = linearPolarization2Y;
 m_DLinearPolarization3YPrev[i][j] = linearPolarization3Y;
 m_DElectricFieldDisplacementYPrevPrev[i][j] = m_DElectricFieldDisplacementYPrev[i][j];
 m_DElectricFieldDisplacementYPrev[i][j] = m_DElectricFieldDisplacementY[i][j];


 Material *mat = geometry->getMaterial(i,j);

 double Ey = i==0?0.0:m_DElectricFieldY[i-1][j];
 double Ex = j==0?0.0:m_DElectricFieldX[i][j-1];

 A = mat->getMagneticCoefficientA();
 B = mat->getMagneticCoefficientB();

 double EIncX = 0;
 double EIncY = 0;

 if( mat->isSource() )
 {
  Source *source = (Source *) mat;
  EIncX = source->getElectricFieldX( i, n );
  EIncY = source->getElectricFieldY( j, n );
 }

 m_DMagneticFieldZ[i][j] =
  A * m_DMagneticFieldZ[i][j]
  + B * ( m_DElectricFieldX[i][j]
```

```
  - Ex
  - m_DElectricFieldY[i][j]
  + Ey
  - EIncX
  + EIncY );
 }
 // calculate the total electric field
 m_DTotalElectricField[i][j] = sqrt(m_DElectricFieldY[i][j]*m_DElectricFieldY[i][j] +
m_DElectricFieldX[i][j]*m_DElectricFieldX[i][j]);

 return SUCCESS;
}

int PulsedSecondOrderSolver::processPointTE(int i, int j, int n)
{
 Material* mat = geometry->getMaterial(i,j);

 double linearPolarization1;
 double linearPolarization2;
 double linearPolarization3;
 double denominator;
 double Z1,Z2,Z3;
 double a1,a2,a3,c1,c2,c3,g1,g2,g3,h1,h2,h3,h4;

 double tempEFieldZ = m_DElectricFieldZ[i][j];

 if( mat->isBoundary() )
 {
  linearPolarization1 = 0;
  linearPolarization2 = 0;
  linearPolarization3 = 0;
  double mAx = bc->getMagneticCoefficientA(),
   mBx = bc->getMagneticCoefficientB(),
   mAy = bc->getMagneticCoefficientA(),
   mBy = bc->getMagneticCoefficientB();
  double eAx = bc->getElectricCoefficientA(),
   eBx = bc->getElectricCoefficientB(),
   eAy = bc->getElectricCoefficientA(),
   eBy = bc->getElectricCoefficientB();

  //were in the left layer
  if( i < PML_width )
  {
   //were in the bottom left corner
   if( j < PML_width )
   {
    mAy = bc->getMagneticFieldExponentialCoefficientA( PML_width - i - 1 );
    mBy = bc->getMagneticFieldExponentialCoefficientB( PML_width- i - 1 );
    mAx = bc->getMagneticFieldExponentialCoefficientA( PML_width - j - 1 );
    mBx = bc->getMagneticFieldExponentialCoefficientB( PML_width- j - 1 );

    eAx = bc->getElectricFieldExponentialCoefficientA( PML_width - i - 1 );
    eBx = bc->getElectricFieldExponentialCoefficientB( PML_width - i - 1 );
    eAy = bc->getElectricFieldExponentialCoefficientA( PML_width - j - 1 );
    eBy = bc->getElectricFieldExponentialCoefficientB( PML_width - j - 1 );
   }
   //were in the top left corner
   else if( j >= m_iHeight- PML_width )
   {
    mAx = bc->getMagneticFieldExponentialCoefficientA( PML_width - i - 1 );
    mBx = bc->getMagneticFieldExponentialCoefficientB( PML_width- i - 1 );
    mAy = bc->getMagneticFieldExponentialCoefficientA( j-m_iHeight+PML_width ) ;
    mBy = bc->getMagneticFieldExponentialCoefficientB( j-m_iHeight+PML_width );

    eAx = bc->getElectricFieldExponentialCoefficientA( PML_width - i - 1 );
    eBx = bc->getElectricFieldExponentialCoefficientB( PML_width - i - 1 );
    eAy = bc->getElectricFieldExponentialCoefficientA( j-m_iHeight+PML_width );
    eBy = bc->getElectricFieldExponentialCoefficientB( j-m_iHeight+PML_width );
   }
   //were in the left layer only
```

```
 else
 {
  mAy = bc->getMagneticFieldExponentialCoefficientA( PML_width - i - 1 );
  mBy = bc->getMagneticFieldExponentialCoefficientB( PML_width- i - 1 );
  mAx = bc->getMagneticCoefficientA();
  mBx = bc->getMagneticCoefficientB();

  eAx = bc->getElectricFieldExponentialCoefficientA( PML_width - i - 1 );
  eBx = bc->getElectricFieldExponentialCoefficientB( PML_width - i - 1 );
  eAy = bc->getElectricCoefficientA();
  eBy = bc->getElectricCoefficientB();
 }
}
else if( i >= m_iWidth - PML_width )
{
 //were in the bottom right corner
 if( j < PML_width )
 {
  mAy = bc->getMagneticFieldExponentialCoefficientA( i-m_iWidth+PML_width );
  mBy = bc->getMagneticFieldExponentialCoefficientB( i-m_iWidth+PML_width );
  mAx = bc->getMagneticFieldExponentialCoefficientA( PML_width - j - 1 );
  mBx = bc->getMagneticFieldExponentialCoefficientB( PML_width- j - 1 );

  eAx = bc->getElectricFieldExponentialCoefficientA( i-m_iWidth+PML_width );
  eBx = bc->getElectricFieldExponentialCoefficientB( i-m_iWidth+PML_width );
  eAy = bc->getElectricFieldExponentialCoefficientA( PML_width - j - 1 );
  eBy = bc->getElectricFieldExponentialCoefficientB( PML_width - j - 1 );
 }
 //were in the top right corner
 else if( j >= m_iHeight - PML_width )
 {
  mAy = bc->getMagneticFieldExponentialCoefficientA( i-m_iWidth+PML_width );
  mBy = bc->getMagneticFieldExponentialCoefficientB( i-m_iWidth+PML_width );
  mAx = bc->getMagneticFieldExponentialCoefficientA( j-m_iHeight+PML_width );
  mBx = bc->getMagneticFieldExponentialCoefficientB( j-m_iHeight+PML_width );

  eAx = bc->getElectricFieldExponentialCoefficientA( i-m_iWidth+PML_width );
  eBx = bc->getElectricFieldExponentialCoefficientB( i-m_iWidth+PML_width );
  eAy = bc->getElectricFieldExponentialCoefficientA( j-m_iHeight+PML_width );
  eBy = bc->getElectricFieldExponentialCoefficientB( j-m_iHeight+PML_width );
 }
 //were in the right layer only
 else
 {
  mAy = bc->getMagneticFieldExponentialCoefficientA( i-m_iWidth+PML_width );
  mBy = bc->getMagneticFieldExponentialCoefficientB( i-m_iWidth+PML_width );
  mAx = bc->getMagneticCoefficientA();
  mBx = bc->getMagneticCoefficientB();

  eAx = bc->getElectricFieldExponentialCoefficientA( i-m_iWidth+PML_width );
  eBx = bc->getElectricFieldExponentialCoefficientB( i-m_iWidth+PML_width );
  eAy = bc->getElectricCoefficientA();
  eBy = bc->getElectricCoefficientB();
 }
}
//were in the bottom layer only
else if( j < PML_width )
{
 mAy = bc->getMagneticCoefficientA();
 mBy = bc->getMagneticCoefficientB();
 mAx = bc->getMagneticFieldExponentialCoefficientA( PML_width - j - 1 );
 mBx = bc->getMagneticFieldExponentialCoefficientB( PML_width - j - 1 );

 eAx = bc->getElectricCoefficientA();
 eBx = bc->getElectricCoefficientB();
 eAy = bc->getElectricFieldExponentialCoefficientA( PML_width - j - 1 );
 eBy = bc->getElectricFieldExponentialCoefficientB( PML_width - j - 1 );
}
//were in the top layer only
else if( j >= m_iHeight - PML_width )
```

```
{
 mAy = bc->getMagneticCoefficientA();
 mBy = bc->getMagneticCoefficientB();
 mAx = bc->getMagneticFieldExponentialCoefficientA( j-m_iHeight+PML_width );
 mBx = bc->getMagneticFieldExponentialCoefficientB( j-m_iHeight+PML_width );

 eAx = bc->getElectricCoefficientA();
 eBx = bc->getElectricCoefficientB();
 eAy = bc->getElectricFieldExponentialCoefficientA( j-m_iHeight+PML_width );
 eBy = bc->getElectricFieldExponentialCoefficientB( j-m_iHeight+PML_width );
}
else
{
 cerr << "Error: Enterened erroneous region!\n";
}

double Ezx = j==m_iHeight-1?0.0:m_DElectricFieldZ[i][j+1];
double Ezy = i==m_iWidth-1?0.0:m_DElectricFieldZ[i+1][j];

m_DMagneticFieldX[i][j] = mAx * m_DMagneticFieldX[i][j]
 + mBx * ( m_DElectricFieldZ[i][j]
 - Ezx );

m_DMagneticFieldY[i][j] = mAy * m_DMagneticFieldY[i][j]
 + mBy * ( Ezy
 - m_DElectricFieldZ[i][j] );

double Hy = i==0?0.0:m_DMagneticFieldY[i-1][j];
double Hx = j==0?0.0:m_DMagneticFieldX[i][j-1];

m_DElectricFieldDisplacementZ[i][j] =
 m_DElectricFieldDisplacementZ[i][j]
 + geometry->getDeltaTime()/geometry->getDeltaSpace()
 * ( m_DMagneticFieldY[i][j]
 - Hy
 - m_DMagneticFieldX[i][j]
 + Hx );

m_DElectricFieldZX[i][j] =
 eAx * m_DElectricFieldZX[i][j]
 + eBx * ( m_DMagneticFieldY[i][j]
 - Hy );

m_DElectricFieldZY[i][j] =
 eAy * m_DElectricFieldZY[i][j]
 + eBy * (- m_DMagneticFieldX[i][j]
 + Hx );

m_DElectricFieldZ[i][j] =
 m_DElectricFieldZX[i][j] + m_DElectricFieldZY[i][j];
}
else
{
 double A = mat->getMagneticCoefficientA();
 double B = mat->getMagneticCoefficientB();

 double Ezx = j==m_iHeight-1?0.0:m_DElectricFieldZ[i][j+1];
 double Ezy = i==m_iWidth-1?0.0:m_DElectricFieldZ[i+1][j];

 double EzIncX = 0;
 double EzIncY = 0;

 Source *source = NULL;

 if( ( source = geometry->isPointLeftOfSource( i, j ) ) != NULL )
 {
  EzIncY = source->getElectricFieldZy( j, n );
 }
 else if( ( source = geometry->isPointBelowSource( i, j ) ) != NULL )
 {
```

```
 EzIncX = source->getElectricFieldZx( i, n );
}

m_DMagneticFieldX[i][j] = A * m_DMagneticFieldX[i][j]
 + B * ( m_DElectricFieldZ[i][j]
 - Ezx
 + EzIncX);

m_DMagneticFieldY[i][j] = A * m_DMagneticFieldY[i][j]
 + B * ( Ezy
 - m_DElectricFieldZ[i][j]
 - EzIncY);

double Hy = i==0?0.0:m_DMagneticFieldY[i-1][j];
double Hx = j==0?0.0:m_DMagneticFieldX[i][j-1];

B = geometry->getDeltaTime()/geometry->getDeltaSpace();
A = (1 - mat->getElectricConductivity()*geometry->getDeltaTime()/2.0) /
 (1 - mat->getElectricConductivity()*geometry->getDeltaTime()/2.0);

double HIncX = 0;
double HIncY = 0;

if( mat->isSource() )
{
 Source *source = (Source *) mat;
 HIncX = source->getMagneticFieldX( i, n );
 HIncY = source->getMagneticFieldY( j, n );
}

m_DElectricFieldDisplacementZ[i][j] =
 m_DElectricFieldDisplacementZ[i][j]
 + B * ( m_DMagneticFieldY[i][j]
 - Hy
 - m_DMagneticFieldX[i][j]
 + Hx
 + HIncX
 - HIncY );

if( mat->isDispersive() )
{
 if( mat->isMetal() )
 {
 h1 = mat->getCoefficientH1();
 h2 = mat->getCoefficientH2();
 h3 = mat->getCoefficientH3();
 h4 = mat->getCoefficientH4();

  m_DElectricFieldZ[i][j] = ( h1*m_DElectricFieldDisplacementZ[i][j] - 4*m_DElectricFieldDisplacementZPrev[i][j]
  + h2*m_DElectricFieldDisplacementZPrevPrev[i][j] + 4*mat->getElectricPermittivity()*m_DElectricFieldZ[i][j]
  - h3*m_DElectricFieldPrevZ[i][j] )
  / ( h4 );
 }
 else
 {
 a1 = mat->getCoefficientA1();
 a2 = mat->getCoefficientA2();
 a3 = mat->getCoefficientA3();
 c1 = mat->getCoefficientW1();
 c2 = mat->getCoefficientW2();
 c3 = mat->getCoefficientW3();
 g1 = mat->getCoefficientG1();
 g2 = mat->getCoefficientG2();
 g3 = mat->getCoefficientG3();
  denominator = -a1*a2*a3 + a1*c2*c3 + c2*c1*a3 - 2*c1*c2*c3 + c3*c1*a2;


  Z1 = c1*( m_DElectricFieldDisplacementZ[i][j] + m_DElectricFieldDisplacementZPrevPrev[i][j] -
  m_DLinearPolarization2PrevPrev[i][j] - m_DLinearPolarization3PrevPrev[i][j] ) +
  4*m_DLinearPolarization1Prev[i][j] + g1*m_DLinearPolarization1PrevPrev[i][j];
```

```
   Z2 = c2*( m_DElectricFieldDisplacementZ[i][j] + m_DElectricFieldDisplacementZPrevPrev[i][j] -
    m_DLinearPolarization1PrevPrev[i][j] - m_DLinearPolarization3PrevPrev[i][j] ) +
    4*m_DLinearPolarization2Prev[i][j] + g2*m_DLinearPolarization2PrevPrev[i][j];

   Z3 = c3*( m_DElectricFieldDisplacementZ[i][j] + m_DElectricFieldDisplacementZPrevPrev[i][j] -
    m_DLinearPolarization1PrevPrev[i][j] - m_DLinearPolarization2PrevPrev[i][j] ) +
    4*m_DLinearPolarization3Prev[i][j] + g3*m_DLinearPolarization3PrevPrev[i][j];

   linearPolarization1 = Z1*(-a2*a3+c2*c3)/denominator-c1*(-a3+c3)/denominator*Z2-c1*(c2-a2)/denominator*Z3;

   linearPolarization2 = -c2*(-a3+c3)/denominator*Z1-Z2*(a1*a3-c1*c3)/denominator+c2*(a1-c1)/denominator*Z3;

   linearPolarization3 = -c3*(c2-a2)/denominator*Z1+c3*(a1-c1)/denominator*Z2-Z3*(a1*a2-c1*c2)/denominator;
   m_DElectricFieldZ[i][j] = (1.0/mat->getElectricPermittivity()) * ( m_DElectricFieldDisplacementZ[i][j] - linearPolarization1 -
linearPolarization2 - linearPolarization3);
  }
 }
 else
 {
  A = mat->getElectricCoefficientA();
  B = mat->getElectricCoefficientB();
  linearPolarization1 = 0;
  linearPolarization2 = 0;
  linearPolarization3 = 0;
  m_DElectricFieldZ[i][j] =
   A * m_DElectricFieldZ[i][j]
   + B * ( m_DMagneticFieldY[i][j]
   - Hy
   - m_DMagneticFieldX[i][j]
   + Hx
   + HIncX
   - HIncY );
 }
 m_DElectricFieldPrevZ[i][j] = tempEFieldZ;
 m_DLinearPolarization1PrevPrev[i][j]=m_DLinearPolarization1Prev[i][j];
 m_DLinearPolarization2PrevPrev[i][j]=m_DLinearPolarization2Prev[i][j];
 m_DLinearPolarization3PrevPrev[i][j]=m_DLinearPolarization3Prev[i][j];

 m_DLinearPolarization1Prev[i][j] = linearPolarization1;
 m_DLinearPolarization2Prev[i][j] = linearPolarization2;
 m_DLinearPolarization3Prev[i][j] = linearPolarization3;
 m_DElectricFieldDisplacementZPrevPrev[i][j] = m_DElectricFieldDisplacementZPrev[i][j];
 m_DElectricFieldDisplacementZPrev[i][j] = m_DElectricFieldDisplacementZ[i][j];

 }

 return SUCCESS;
}
////////////////////////////////////////////////////////////////////////////////////////////////////
//TimeDomainFieldWriter.cpp – outputs the field into a matlab readable format
////////////////////////////////////////////////////////////////////////////////////////////////////
#include "TimeDomainFieldWriter.h"
#include "stdheader.h"
#include "Bitmap.h"
#include "Options.h"
#include "Geometry.h"
#include <stdio.h>
#include <math.h>
#include <iostream.h>

TimeDomainFieldWriter::TimeDomainFieldWriter( Geometry* geom, Options* opt, char* FieldDir )
{
 options = opt;
 geometry = geom;
 m_DField = new double*[options->getSamplingPointCount()];
 if( options->getSimulationMode() == TM_MODE )
 {
 sprintf(m_sFilename, "%s_%s_%s.tim", options->getSimulationName(),TM_MODE_TOKEN, FieldDir );
 }
```

```
else
{
 sprintf(m_sFilename, "%s_%s_%s.tim", options->getSimulationName(),TE_MODE_TOKEN, FieldDir );
}

for(int i=0;i<options->getSamplingPointCount();i++)
{
 m_DField[i] = new double[ geometry->getGridTime() ];
}
}

int TimeDomainFieldWriter::reset( int oldTime )
{
 int rc = SUCCESS;
 for(int i=0;i<options->getSamplingPointCount();i++)
 {
  double *temp = m_DField[i];
  m_DField[i] = new double[ geometry->getGridTime() ];
  memcpy( m_DField[i], temp, sizeof(double)*oldTime );
  delete temp;
 }
 return rc;
}

int TimeDomainFieldWriter::addPoint( double** field, int timeStep )
{
 int rc = SUCCESS;
 int x,y;
 for(int i=0;i<options->getSamplingPointCount();i++)
 {
  x = options->getXSamplingPoint(i);
  y = options->getYSamplingPoint(i);
  if( x >= geometry->getGridWidth() )
   m_DField[i][timeStep] = 0;
  else if( y >= geometry->getGridHeight() )
   m_DField[i][timeStep] = 0;
  else if( timeStep >= geometry->getGridTime() )
   m_DField[i][timeStep] = 0;
  else
   m_DField[i][timeStep] = field[ x ][ y ];
 }
 return rc;
}

TimeDomainFieldWriter::~TimeDomainFieldWriter()
{
 for(int i=0;i<options->getSamplingPointCount();i++)
 {
  delete [] m_DField[i];
 }
 delete [] m_DField;
}

int TimeDomainFieldWriter::write()
{
 int rc = SUCCESS;
 int i,j;

 FILE *fp = NULL;

 fp = fopen( m_sFilename, "wt" );
 if( fp == NULL )
 {
  rc = INVALID_FILENAME;
  return rc;
 }

 for( j=0;j<geometry->getGridTime(); j++ )
 {
  fprintf(fp," %le", (double)j*geometry->getDeltaTime() );
```

```
}
fprintf(fp,"\n");

for(i=0;i<options->getSamplingPointCount();i++)
{
 for( j=0;j<geometry->getGridTime(); j++ )
 {
  fprintf(fp," %le",m_DField[i][j] );
 }
 fprintf(fp,"\n");
}


if( fp != NULL )
 fclose(fp);
return rc;
}
```