THREE DIMENSIONAL SIMULATIONS OF COMPLEX STRUCTURES

A Thesis Submitted to the

Graduate School of Natural and Applied Sciences of

Dokuz Eylül University

In Partial Fulfillment of the Requirements for

the Degree of Master of Science in Computer Engineering

by

Mustafa KASAP

August, 2003

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We certify that we have read this thesis and **"THREE DIMENSIONAL SIMULATIONS OF COMPLEX STRUCTURES"** completed by MUSTAFA **KASAP** under supervision of **PROF. DR. TATYANA YAKHNO and** VALERY **YAKHNO** and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof Dr. Tatyana Yakhno

(Supervisor)

Prof. Dr. Valery YAKHNO

(Coadvisor)

Prof. Dr. Alp Kut

(Committee Member)

Prof. Dr. İsmail Sökmen

(Committee Member)

Yrd. Doç. Dr. Şen Çakır

(Committee Member)

Approved by the

Graduate School of Natural and Applied Sciences

Prof.Dr. Cahit HELVACI

Director

ACKNOWLEDGEMENTS

I would like to thank my supervisors Prof. Dr. Tatyana Yakhno and Prof. Dr. Valery Yakhno for giving me great ideas and concentration about this thesis.

Mustafa KASAP

ABSTRACT

Simulation of complex structures such as electromagnetic waves, smoke etc. is the technique we use to describe the shape of abstract or concrete objects. In latter studies we use the shapes of these objects to simulate the process for different time intervals.

Geometric modeling provides a description or model that is analytical, mathematical, and abstract rather than concrete. We prefer such methods because it is a convenient and economical substitute for the real object or process. It is easier and more practical to analyze a model than to test or measure or experiment with the real object.

From the point of ideas above simulation process in computer environment requires deep mathematical knowledge and also powerful computers equipped with powerful graphic computation capabilities. In this thesis we studied electromagnetic wave propagation with the support of required equipment mentioned.

Keywords: Geometric modeling, complex structures, simulation

ÖZET

Elektromanyetik dalgalar, duman gibi karmaşık yapıların simülasyonu, soyut yada somut cisimlerin tanımlanmasında kullandığımız tekniktir. İleriki çalışmalarda bu cisimlerin şekillerini kullanarak değişik zaman dilimlerine ait işlem simulasyonlarını elde edeceğiz.

Geometrik modelleme somut bir tanımdan yada modelden öte analitik, matematiksel ve soyut bir yaklaşım sunar. Bu çeşit bir yöntemi seçmiş olmamızın sebebi gerçek nesne ve işlemleri oluşturmaktan daha uygun ve ekonomik oluşudur. Gerçek nesneler üzerinde işlemler, ölçümler yapmaktansa bu tür bir yöntem daha kolay ve pratiktir.

Yukarıda bahsedilen fikirler ışığında bilgisayar ortamında simülasyon işlemi derin matematik bilgisi ile yüksek kapasiteli ve güçlü grafik işleme gücüne sahip bilgisayarlar gerektirmektedir. Bu tez çalışmasında bahsedilen gereçler kullanılarak elektromanyetik dalgaların yayılımları incelenmektedir.

Anahtar sözcükler: Geometrik modelleme, Karmaşık cisimler, Simülasyon

CONTENTS

Chapter Three

THE CAUCHY PROBLEM FOR MAXWELL'S FOR THE ISOTROPIC MEDIUM

Chapter Four

THE CAUCHY PROBLEM FOR MAXWELL'S FOR THE ANISOTROPIC MEDIUM

Chapter Five

CONCLUSION & FUTURE WORKS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Anisotropic electromagnetic theory has been an active research subject for the last years due to its applications to engineering, automatics, computer facilities, measurement technologies, and geophysics[1]. The mathematical models of wave processes inside of electromagnetic bodies are described by the special system of differential equations of electromagnetodynamics[2]. The number of electromagnetic constants appear in this system is big, and as a result of it the solution of this system depends on many electromagnetic constants. This fact is interesting not only in theory but also in applications[1, 3, 4, 5, 6]. For this reason to study the explicit formulas for fundamental solutions of the Cauchy problem for electromagnetodynamic system and computer simulation of the wave propagation using these formulas is very important problem. The aim of this thesis is to study this problem.

1.1 Main Equations of Electromagnetics

The theory developed by Maxwell and Hertz describes electromagnetic waves by means of two vector fields E and H, the electric and magnetic fields. The properties of the isotropic homogeneous medium in which the waves propagate are given by the material constants $\varepsilon > 0$, $\mu > 0$, σ.

The properties of the anisotropic homogeneous medium are given by symmetrical matrices ε, μ, σ, where ε, μ are positive defined.

The complete set of basic Maxwell's equations has the form[7]:

$$curl_x H = \frac{1}{c}\frac{\partial D}{\partial t} + \frac{4\pi}{c} J, \tag{1.1}$$

$$curl_x E = -\frac{1}{c}\frac{\partial B}{\partial t}, \tag{1.2}$$

$$div_x B = 0, \tag{1.3}$$

$$div_x D = 4\pi\rho, \tag{1.4}$$

where vectors D, B and J are electric displacement, magnetic induction and current density, ρ is the density of electric charges. The values ρ and J satisfy the relation

$$\frac{\partial \rho}{\partial t} + div_x J = 0, \tag{1.5}$$

and hence equations (1.1) and (1.2) are related to each other. This relation expresses the law of the conservation of electric charge. In addition there are constitutive relations that express D, B and J in terms of E and H. These equations are

$$D = \varepsilon E, \qquad B = \mu H, \qquad J = \sigma E + j, \tag{1.6}$$

where ε is the dielectric permeability, μ is the magnetic permeability, σ is the conductivity, j is the density of currents, arising from the action of the external electromagnetic forces.

Later we shall assume

$$E = 0, H = 0, \qquad \rho = 0, j = 0, \text{ for } \quad t < 0. \tag{1.7}$$

This means there is no electromagnetic field, currents, or electric charges at the time $t < 0$. We note that equation (1.3) follows immediately from (1.2), (1.6) and

(1.7), and equation (1.4) can be obtained from (1.1), (1.5), and (1.6). Really, applying the operator div_x to (1.2) we have

$$\frac{\partial}{\partial t} div_x B = 0, \qquad \text{where } (div_x curl_x Z = 0)$$

And we can find from (1.6)

$$div_x B\big|_{t<0} = 0.$$

These last two relations imply (1.3). Applying the operator div_x to (1.1) we find

$$div_x \frac{\partial D}{\partial t} + 4\pi div_x J = 0$$

and using (1.5) we have

$$\frac{\partial}{\partial t}\left[div_x D - 4\pi\rho\right] = 0.$$

Equation (1.4) follows from this last relation and (1.6), (1.7).

1.2 Main Problem

Let us consider now the problem of determining vector-functions E, H and the function ρ satisfying (1.1)-(1.7) and j is the known vector function. It is easy to show that a solution of this problem may be found successively. On the first step we determine E, H appearing in equations

$$curl_x H = \frac{1}{c}\frac{\partial}{\partial t}(\varepsilon E) + \frac{4\pi}{c}\sigma E + \frac{4\pi}{c} j, \qquad (1.8)$$

$$curl_x E = -\frac{1}{c}\frac{\partial}{\partial t}(\mu H), \qquad (1.9)$$

subject to conditions

$$E\big|_{t<0} = 0, \qquad H\big|_{t<0} = 0. \qquad (1.10)$$

And then recover a function ρ from relations

$$\frac{\partial \rho}{\partial t} = -div_x\left[\sigma E + \frac{4\pi}{c} j\right], \qquad (1.11)$$

$$\rho\big|_{t<0} = 0. \qquad (1.12)$$

Equations (1.8)-(1.9) may be written as follows

$$curl_x H = \bar{\varepsilon}\left(\frac{\partial E}{\partial t} + \bar{\sigma}E\right) + \bar{j}, \quad curl_x E = -\bar{\mu}\frac{\partial H}{\partial t}, \qquad (1.13)$$

$$E\big|_{t<0} = 0, \qquad H\big|_{t<0} = 0, \text{ where} \qquad (1.14)$$

$$\bar{\varepsilon} = \frac{\varepsilon}{c}, \ \bar{\mu} = \frac{\mu}{c}, \qquad \bar{\sigma} = \frac{4\pi\sigma}{c}, \qquad \bar{j} = \frac{4\pi j}{c}.$$

Further we shall omit bars over letters ε, μ, σ, j for the simplicity of writing.

From the equations (1.13), (1.14), and (1.11) we can write more specific Maxwell equations for our assumptions.

$$curl H = \varepsilon\left(\frac{\partial E}{\partial t} + \sigma E\right) + j, \qquad (1.15)$$

$$curlE = -\mu \frac{\partial H}{\partial t}, \tag{1.16}$$

$$E\big|_{t<0} = 0, \qquad H\big|_{t<0} = 0, \tag{1.17}$$

$$\frac{d\rho}{dt} = -div\left[\frac{c}{4\pi\varepsilon}\sigma E + j\right], \rho\big|_{t<0} = 0, \tag{1.18}$$

1.3 The Goal of the Thesis

One of the goals of this thesis is to solve the Cauchy problem given in equation (1.15)–(1.17). The other goal is to find explicit formulas for the solutions of electromagnetic wave propagation by computer using these solutions.

1.4 Methods and Tools

In this thesis we have used two kinds of tools, one is MatLAB other is Visual Studio Program Development Environment[8, 9, 10].

MatLAB is an environment where the user only writes the function formulas and use predefined functions or commands to simulate the representation of that function under different sized spaces. It is easy to use and doesn't require knowledge about programming. With limited training, every person can easily write his function which's syntax is very close to mathematical notation style. Visual Studio is also a development environment. It supports several programming language to write programs. One of these languages is C++. It is difficult to write programs with this language. Because it is low level language it requires some basic knowledge. In this thesis we have used both of these tools for modeling. MatLAB is used to take a snapshot view of any function under any condition. This is preferred because the code written in MatLAB can be understandable. It is the easiest way to develop models rapidly. Once you write the functions that you want to model, it takes time to do computations. It is the main limitation of the MatLAB. Because of computation time limitation we don't use MatLAB for animations.

We preferred Visual Studio with C++ because the code that you write is compiled and standalone executable generated. Because this executable doesn't contain any debug information and consist of machine level code, it is very fast in computations. Disadvantages are writing code is very difficult so one can easily make syntactic or logical errors. Because graphic output required, some initialization process required. And other disadvantage is compilation time. It takes time to generate a single executable but not much as computation.

1.5 The Structure of This Thesis

Thesis consists of five main chapters. In the first chapter brief description of Maxwell and Hertz theories are given. And electromagnetic wave propagation under different conditions is mentioned. In the second chapter, propagation under isotropic homogeneous medium is mentioned. Also programs and their outputs are given. In the third chapter propagation under anisotropic medium explored. In this chapter medium constants kept as simple as possible. In the forth chapter, propagation under anisotropic medium with complex medium constants are explored. In the last chapter analyses of modeling and future works mentioned. Appendix contains tutorial like documentation. It describes step by step, how to prepare a C++ application with OpenGL support from the beginning.

CHAPTER TWO

SIMULATION METHODS

In this chapter terms, symbols, methods that are used in the former chapters will be described.

2.1 Legend of graphics

2.1.1 az_el:

The azimuth, ***az***, is the horizontal rotation about the z-axis as measured in degrees from the negative y-axis. Positive values indicate counterclockwise rotation of the viewpoint.

el is the vertical elevation of the viewpoint in degrees. Positive values of elevation correspond to moving above the object; negative values correspond to moving below the object.

2.1.2 Eps:

Epsilon parameter used in Dirac Delta function

2.1.3 Frequency:

One of the modeling methods for any two or three dimensional function is to take a sample data interval and generate corresponding values for each sample data. Then for each tupple draw a point. At the end connect these points with lines to generate the final graph.

Let us consider for example the following two functions:

$$y = f(x) = x^2 \tag{2.1}$$

$$z = f(x, y) = x^2 + y^2 \tag{2.2}$$

For the function (2.1) above, we can generate two dimensional graphic, for the second function we can generate three dimensional graphic. In the first case we must take a data interval for x axis and find the corresponding y values for this data interval. Say that data interval for x axis is $[-5,5]$. Defining interval is not enough we must determine a set of data within this interval. I.e. all integers in this interval such as $(-5,-4,-3,-2,-1,0,1,2,3,4,5)$. In this data interval set, we have 11 sample points on x axis. Drawing within this method is taking same steps for three dimensional cases.

From this point, ***frequency*** is the number of points used to model the function. By increasing the frequency we get smoother graphics. In three dimensional cases we must take two distinct data samples. In formula (2.2) we take one interval for x axis and other interval for y axis. Generally these two different axis's data interval is same. So the frequency of these two axes is the same and then only one frequency term is used for these two distinct axes.

To generate some sample graphics and show the differences with different frequency values we use a function of two variables, obtained by translating and scaling Gaussian distributions.

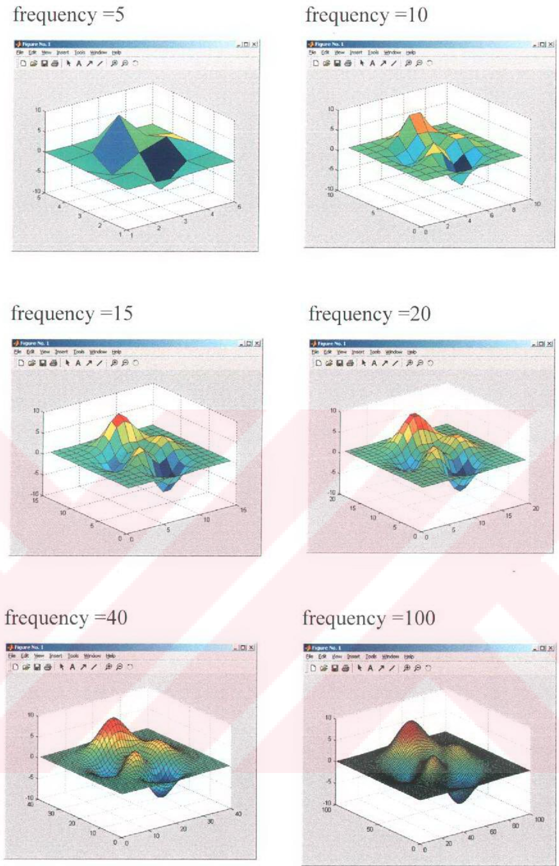

Figure 2.1: Gaussian distributions as sample graphics with different frequency values

As seen in figure 2.1, greater the frequency value, smoother the graphic that we can generate.

2.2 Problem of the frequency

Drawing graph of any function is not as simple as described above. In some generalized functions like Dirac Delta function we encounter with problems. Before making any discussion about these problems lets take a look at Dirac Delta function.

The Dirac Delta function is not a classical function and it is impossible to define this function by point wise manner.

There are many classical functions which are regularizations of the Dirac Delta function. For example,

$$\delta_{\varepsilon}(x) = \frac{1}{2\sqrt{\pi\varepsilon}} \exp\left(-\frac{x^2}{4\varepsilon}\right), \varepsilon \to +0 \qquad (2.3)$$

is the regularization of the Dirac Delta function. This function for sufficiently small $\varepsilon > 0$ may be used instead of the Dirac Delta function as an approximation of it. We will draw this function like the Dirac Delta function. In this function we use a special seed, epsilon, which makes our final graph more realistic, if we choose small epsilon values.

As described above, now we can draw the graph of Dirac Delta regularization defined in formula (2.3) by using the steps explained before.

Below, figure 2.2 is the graph of Dirac Delta regularization with $epsilon = \frac{1}{2^5}$ and within the interval [-5, 5] on x axis. Frequency of the graphic is ~20000.

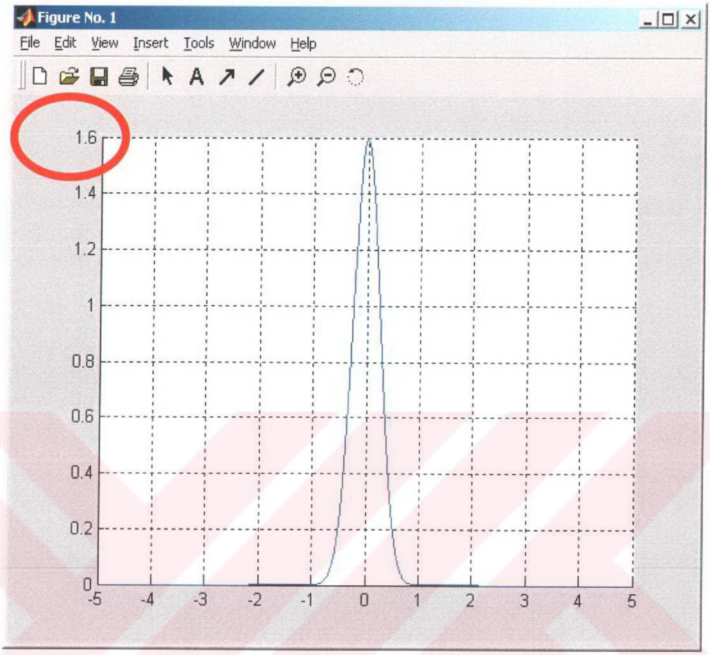

Figure 2.2: Dirac Delta regularization sample with large frequency value equal to ~20000

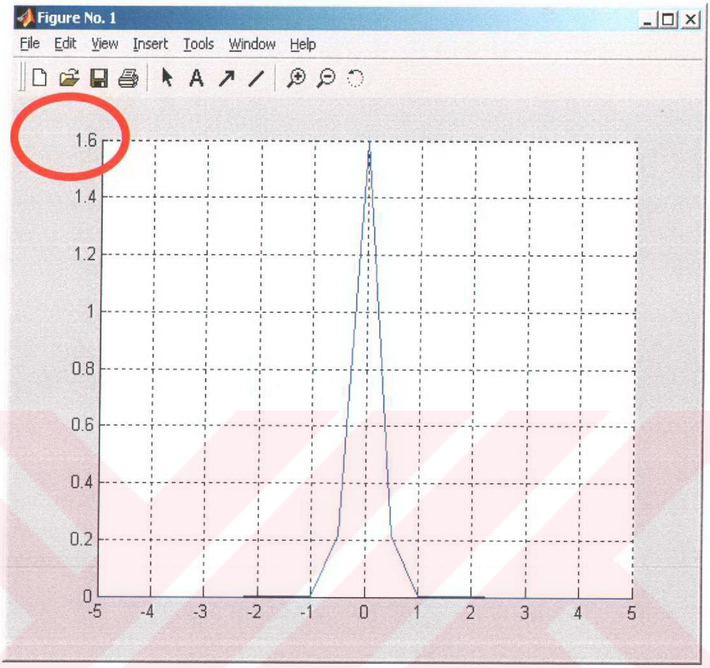

Figure 2.3: Dirac Delta regularization sample with small frequency value equal to 20

In figure 2.3 we select the frequency equal to 20 and resulting graph gets very rough.

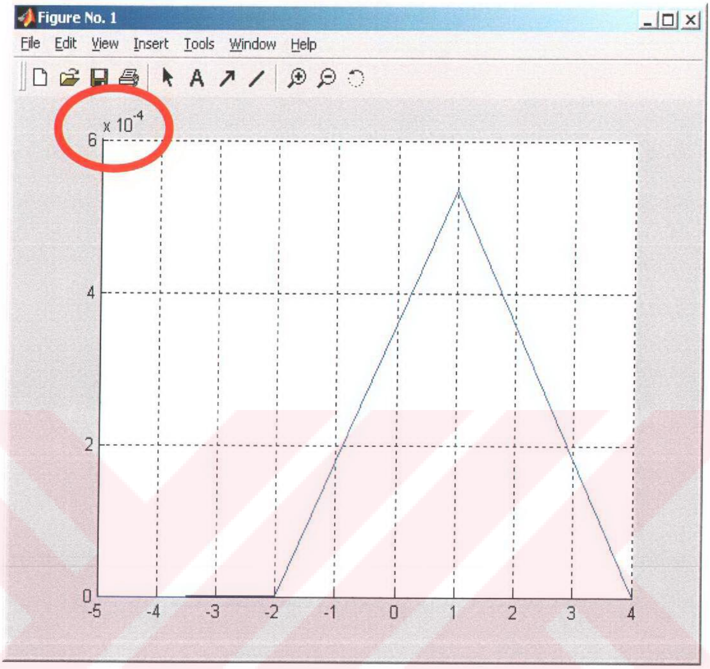

Figure 2.4: Dirac Delta regularization sample with very small frequency value equal to 3

In figure 2.4 we select the frequency equal to 3.

According to the graphs above we must select the correct density value to get the smother and correct graph. Here there is a threshold. Smaller the density value slower the graph generation, and memory consumption. Larger the density value faster the graph generation but rough graph.

2.3 2D modeling of the Dirac Delta regularization for the frequency problem

To show the frequency problem in another perspective we take a part of the function with its parameters that we planned to use in our future experiments.

Dirac delta function defined in formula 2.3 is used, and the related MatLAB code segment is below.

Code Segment 2.1: DiracDelta.m file

```
function y = DiracDelta(eps, x)
y = (1/(2*sqrt(pi*eps))) * exp(-x.^2/(4*eps))
```

$$y=\delta_{\varepsilon}\left(t-\frac{|x|}{a}\right) \tag{2.4}$$

In formula 2.4 we use the same parameter that we planned to use in future experiments. According to these definitions, final code segment that we will use is below. Only we need to set density and epsilon parameters in the code below. Data interval of x axis is in [-20, 20]. This code generates 5 pick sets for 5 distinct time values in the specified properties.

Code Segment 2.2: Program that uses Dirac Delta function

```
X = [0.3, 0.1, 0.7];
a = [2.0, 5.0, 8.0];
x = linspace(-20,20, Frequency);
pTime=linspace(1,5,5);

for i = 1:length(pTime)
  ParameterRange = pTime(i)-(sqrt(x.^2+X(2)^2+X(3)^2)/a(1));
  y = DiracDelta(eps, ParameterRange)
  hold on
```

```
  plot(x,y)
  drawnow;
end
```

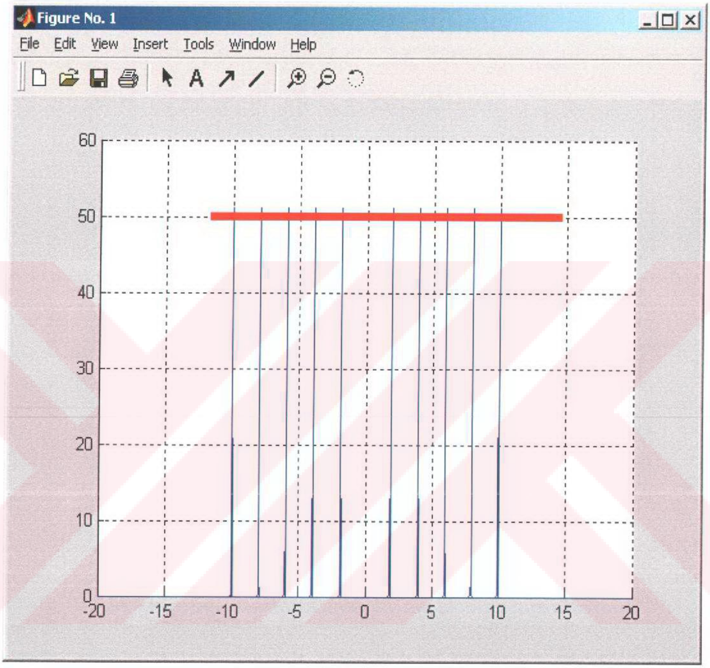

Figure 2.5: Dirac Delta regularization sample with large frequency and epsilon value

Frequency	100000
Epsilon	$\frac{1}{2^{15}}$

In figure 2.5, frequency on X axis is 100000 point in [-20, 20] range and epsilon is 1/2^15. To generate this graphic we set *frequency* parameter of line 4 of *code segment 2* to 100000. With this parameter for 5 distinct time values, we get 5 distinct pick on each side of the origin (Totally 10 picks). When we draw a horizontal line at the highest value of the picks we see that all are same length.

In figure 2.1, density is 100000. This is time consuming process to calculate the y values of 100000 distinct x values. For generating this graph it's not so complex to generate these much number of points. But this Dirac Delta function is used in other formulas such as assumptions. In such formulas if we use these much points, it will take very long time to generate simple time frame of a graph. So we make some optimization and decrease the number of points or decrease the frequency, to make calculation faster then before.

Decreasing frequency or number of points makes the graphic to be unrealistic. In figure 2.5 frequency is 100000. Now we decrease the density to 10000 points. Here is the result:

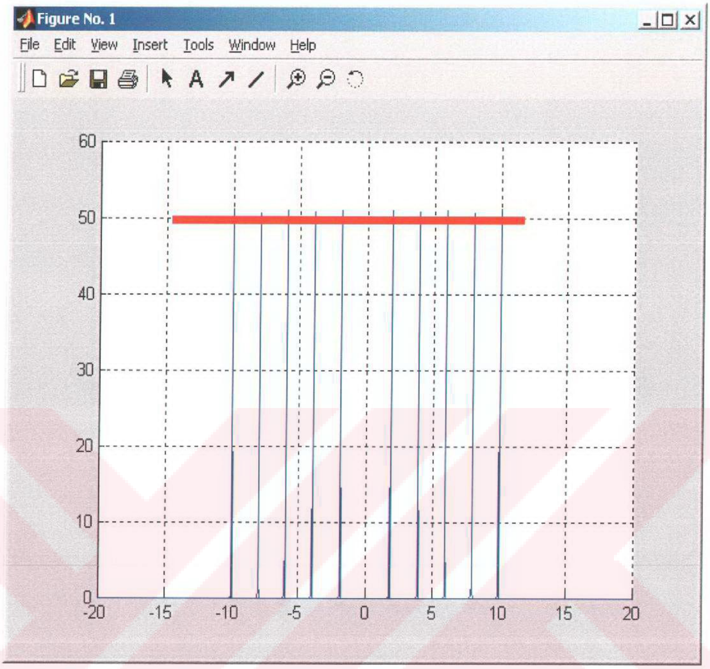

Figure 2.6: Dirac Delta regularization sample with small frequency w.r.t Figure 2.5

Frequency	10000
Epsilon	$\frac{1}{2^{15}}$

When we draw the same line, we see that there are some small length differences between picks.

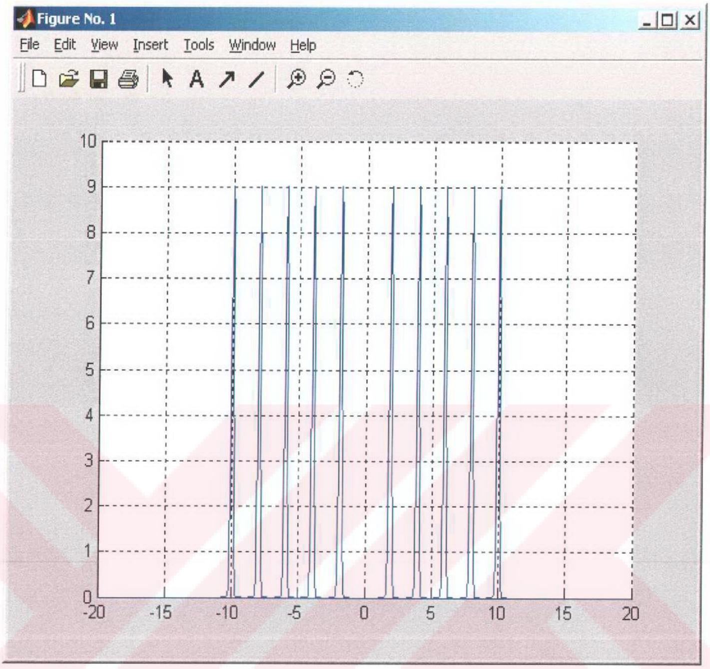

Figure 2.7: Dirac Delta regularization sample with small epsilon w.r.t Figure 2.6

Frequency	10000
Epsilon	$\frac{1}{2^{10}}$

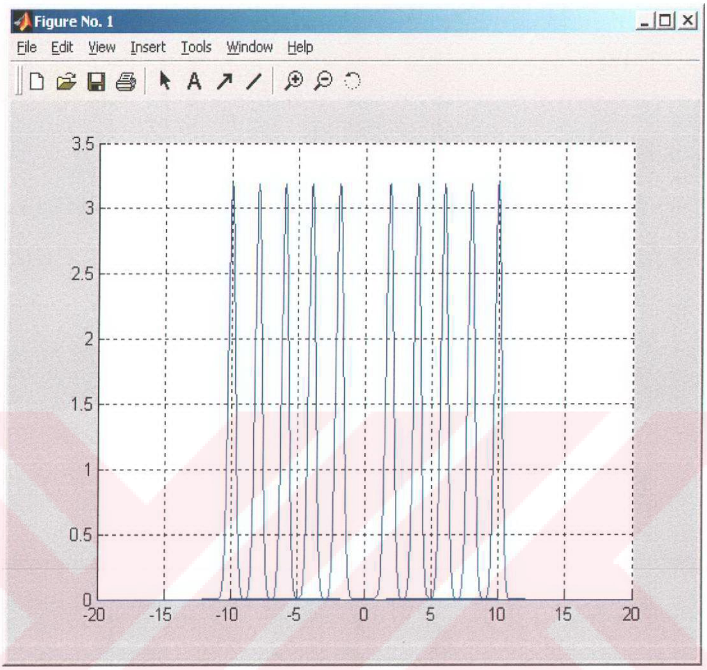

Figure 2.8: Dirac Delta regularization sample with smaller epsilon w.r.t Figure 2.7

Frequency	10000
Epsilon	$\frac{1}{2^7}$

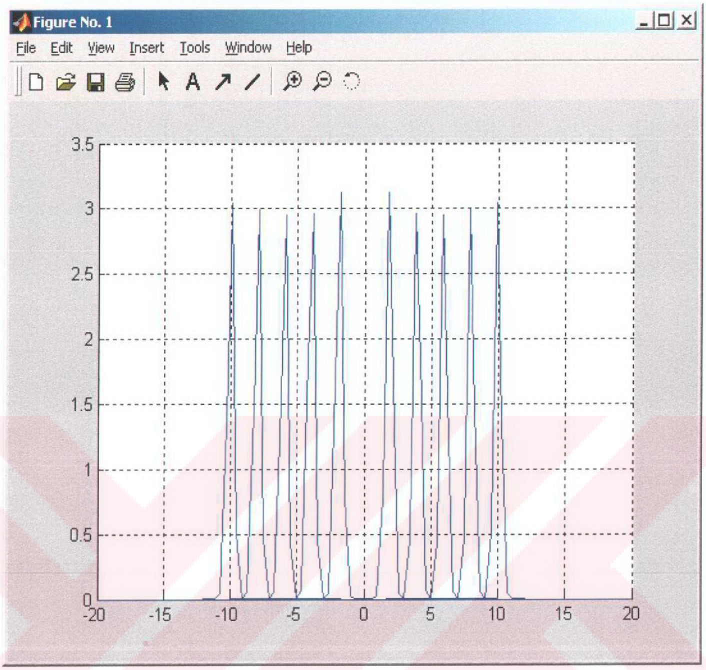

Figure 2.9: Dirac Delta regularization sample with smaller frequency w.r.t Figure 2.8

Frequency	100
Epsilon	$\frac{1}{2^7}$

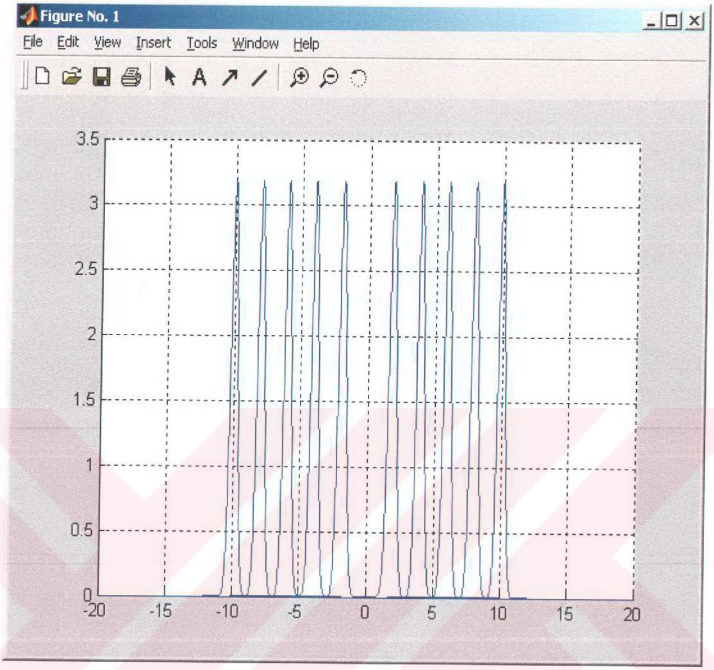

Figure 2.10: Dirac Delta regularization sample with small epsilon w.r.t Figure 2.5

Frequency	100000
Epsilon	$\frac{1}{2^7}$

When we compare the figures between 2.6 and 2.10 we saw that there are big differences between them. When we look at the function or program segment we saw that everything is same only the number of points used in predefined interval is decreasing.

Another point is epsilon value. When we make comparison between figure 2.5 and 2.10 the frequency is same but the epsilon value is different. Resulting graphic also has differences.

To conclude, it's clear that to generate realistic graphics we must use greater density values, and we must work with high resolution data values which can hold $\frac{1}{2^{15}}$ like values.

There is a threshold in such decision point. When we use large density and small data values it takes much time to finish calculation. It requires more powerful computers or special programming techniques such as parallel programming or optimization in the code for its algorithm or for platform specific mathematical processors.

2.4 3D modeling of the Dirac Delta regularization for the frequency problem

Problems explained above is also continuing on three dimensional modeling. In three dimensional model of the Dirac Delta function we expect to see a cylinder with an infinite height. Because the frequency value is not large enough we will encounter with a cylinder which is constructed with different height picks. Below there are three different snapshot of Dirac Delta function modeled under three dimensional space.

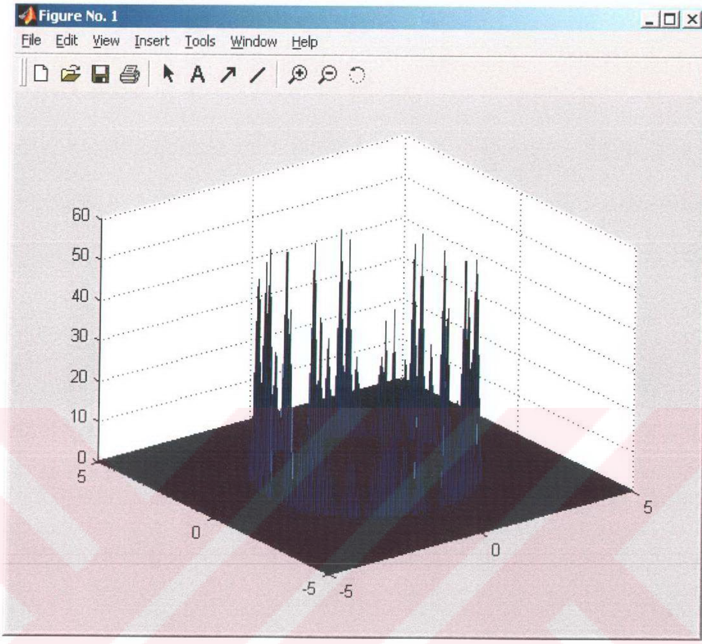

Figure 2.11: Dirac Delta regularization sample in 3D space

Frequency	100 in x and z axis
Epsilon	$\frac{1}{2^{15}}$
Time	3

As explained above, in the figure 2.11 we see a cylinder like object which is constructed by different height picks. There are some holes around the cylinder again because of frequency problem. This makes the model unrealistic. To generate more realistic we need larger frequency values or we will look closer to the model as in figure 2.12. In the former figure the largest value on y axis is ~60, in the latter figure the largest value is 1 because of looking very closer to the same figure.

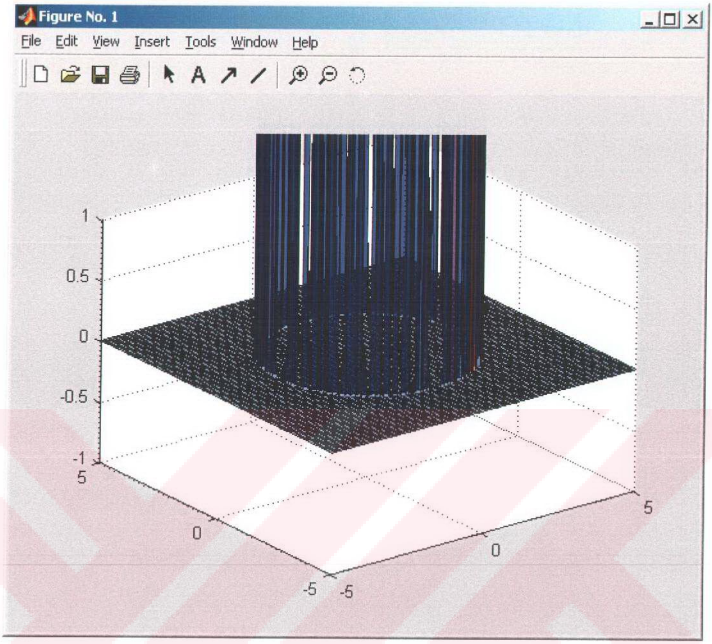

Figure 2.12: Dirac Delta regularization sample in 3D space (with closer look)

In figure 2.12, we see more realistic model because all picks has the same height. This makes the model look like a cylinder. But there are also some holes between picks. To prevent this unrealistic behavior we increase the frequency value and decrease the performance of model generation. In figure 2.13, frequency value is 250; there is no hole between the picks and as a result complete cylinder looking model generated.

In two dimensional modeling, increasing the frequency value linearly decrease the performance linearly too. But in three dimensional modeling, increasing the frequency means there is an increase in both x and z axis so performance will decrease square times the frequency.

So generating more realistic models in three dimensional space costs too much computation time.

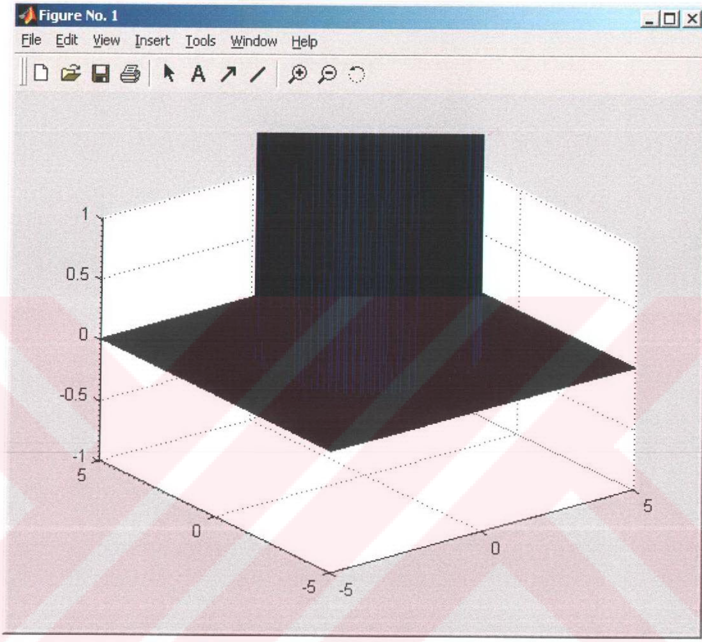

Figure 2.13: Dirac Delta regularization sample in 3D space (with larger frequency)

Frequency	250 in x and z axis
Epsilon	$\frac{1}{2^{15}}$
Time	3

Related MatLAB code segment with the above experiment is as below.

Code Segment 2.3: Dirac Delta regularization sample in 3D

```
clear;

% Define constant vectors
X = [0.3, 0.1, 0.7];

x1 = linspace(-5,5, Frequency);
x2 = linspace(-5,5, Frequency);
y = zeros(length(x1));

%epsilon parameter used in DiracDelta
eps = 1 / 2^15;


pTime=3;

for i = 1:length(x1)
    for j = 1:length(x2)
        X(1) = x1(i);
        X(2) = x2(j);

        Parameter = pTime-norm(X);
        y(i, j) = DiracDelta(eps, Parameter);
    end
end

set(gcf,'Color',[1,0.4,0.6])
surf(x1, x2, y);
axis([min(x1) max(x1) min(x2) max(x2) -1.0 1.0]);
drawnow;
```

2.5 Summary

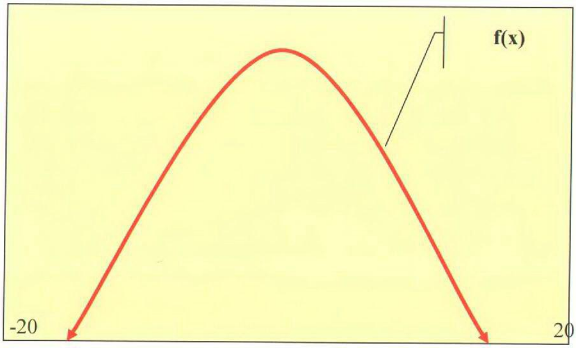

Figure 2.14: Frequency effect

In figure 2.14, expected graphic of the function f(x) is shown with curve. Data interval of the graph is [-20, 20]. Assume that in the first attempt we take the frequency equal to 5. The resulting graph will be similar to the one in figure 2.15.

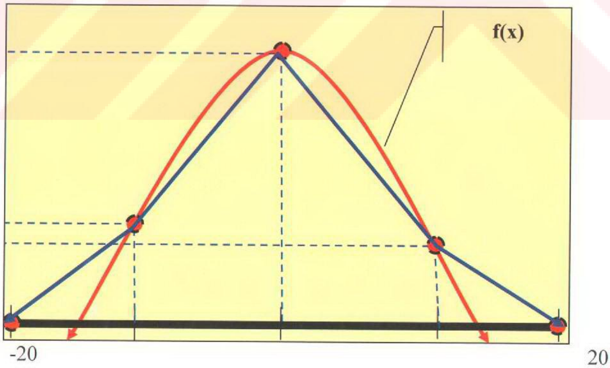

Figure 2.15: Frequency effect

As shown in figure 2.15, the frequency is 5 so we have 5 sample points to draw the function. Problem is by using these 5 points with their corresponding y values to draw the function; we get the curve which is constructed with straight lines. It looks like our expected graph but not same or similar. This is the first problem that why we take greater frequency values.

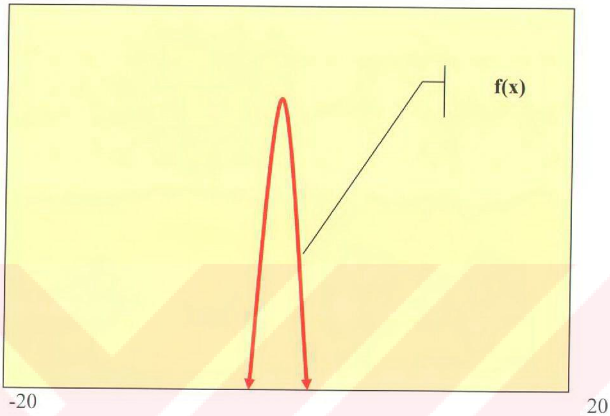

Figure 2.16: Frequency effect

Assume that the expected graph of f(x) is shown in figure 2.16. We take the frequency equal to 4. And start to draw the function with these four sample data point tupples.

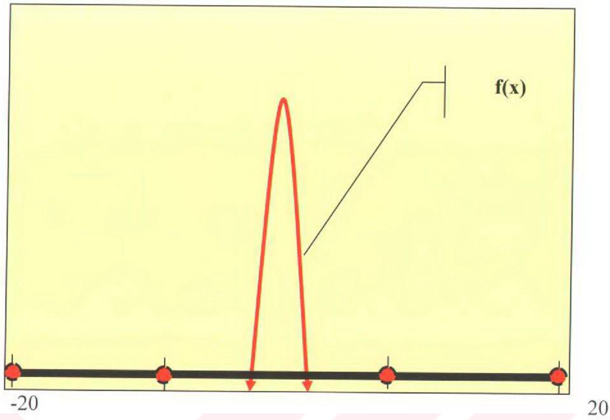

Figure 2.17: Frequency effect

With these four sample data points we find their corresponding y values. The problem is we don't take the frequency value as large enough to fit in the support interval of the original curve. So for four points in figure 2.17, their corresponding values are zero in f(x) function. Then resulting graph is a horizontal line on x axis. But in the reality there is a pick in a very small x axis data interval. This is the second important point that we must take into account while doing further function modeling.

CHAPTER THREE

THE CAUCHY PROBLEM FOR MAXWELL'S FOR THE ISOTROPIC MEDIUM

The explicit formulas for the solution of the Cauchy problem for the Maxwell system in the isotropic case are given in this chapter.

Analysis of these formulas and the simulation of the solution for the Cauchy problem are given in details.

3.1 Assumptions

In this chapter we will assume that ε, μ are positive constant, $\sigma = 0$, j is defined one of the following rules

(A1) $j = \varepsilon \vec{e}\delta(x)\theta_1(t)$,

(A2) $j = \varepsilon \vec{e}\delta(x)\theta(t)$,

(A3) $j = \varepsilon \vec{e}\delta(x)\delta(t)$,

(A4) $j = \dfrac{\vec{e}}{\varepsilon}\delta(x)f(t)$.

Here,

$\delta(x) = \delta(x_1)\delta(x_2)\delta(x_3)$ is the Dirac Delta function with the support at $x_1 = 0, x_2 = 0, x_3 = 0$;

$\delta(t)$ is the Dirac Delta function with the support at $t = 0$;

$\theta(t)$ is the Heaviside function which is defined by

$$\theta(t) = \begin{cases} 1, t \geq 0 \\ 0, t < 0 \end{cases} \tag{3.1a}$$

$$\theta_1(t) = t\theta(t) \tag{3.1b}$$

Remark:

The cases (A1), (A2), (A4) related to the point source with the amplitudes $\theta_1(t)$, $\theta(t)$, $f(t)$ respectively. The case (A3) is connected with the pulse point source. All these sources have the direction e and concentrate at the point $x = 0$.

These four types of the current density will be used as a generator of electromagnetic wave simulator.

We seek the solution of (1.15)-(1.18) in the form

$$H = \frac{1}{\mu} curlA, \; E = -\frac{\partial A}{\partial t} + \nabla\varphi \tag{3.2}$$

Substituting (3.2) into (1.15) we have

$$curl\left(\frac{1}{\mu} curlA\right) = -\varepsilon\frac{\partial^2 A}{\partial t^2} - \sigma\varepsilon\frac{\partial A}{\partial t} + \varepsilon\frac{\partial(\nabla\varphi)}{\partial t} + \varepsilon\sigma(\nabla\varphi) + j \tag{3.3}$$

Using the formula $curlcurlA = \nabla divA - \nabla A$ we find from (3.3)

$$\frac{1}{\mu}\nabla divA - \frac{1}{\mu}\Delta A = -\varepsilon\frac{\partial^2 A}{\partial t^2} - \sigma\varepsilon\frac{\partial A}{\partial t} + \varepsilon\frac{\partial(\nabla\varphi)}{\partial t} + \varepsilon\sigma(\nabla\varphi) + j \tag{3.4}$$

We will find $\nabla\varphi$ such that

$$\varepsilon\frac{\partial(\nabla\varphi)}{\partial t} + \varepsilon\sigma(\nabla\varphi) = \frac{1}{\mu}\nabla divA\,, \qquad \nabla\varphi\big|_{t<0} = 0 \tag{3.5a}$$

We have from (3.4), (3.5)

$$\varepsilon\frac{\partial^2 A}{\partial t^2} - \frac{1}{\mu}\Delta A + \varepsilon\sigma\frac{\partial A}{\partial t} = j \tag{3.5b}$$

Therefore the problem of finding E, H satisfying (1.15)-(3.1) may be reduced to the solution of the Cauchy problem for the vector wave equation.

$$\frac{\partial^2 A}{\partial t^2} = a^2\Delta A - \sigma\frac{\partial A}{\partial t} + \frac{1}{\varepsilon}j\,, \quad a^2 = \frac{1}{\mu\varepsilon} \tag{3.6}$$

$$A\big|_{t<0} = 0 \tag{3.7}$$

and then the solution of (3.5).

Let $G(x,t)$ be a fundamental solution of the Cauchy problem for the vector wave operator, this means that $G(x,t)$ satisfies

$$\frac{\partial^2 G}{\partial t^2} = a^2\Delta G + \mathrm{I}\delta(x)\delta(t)\,, \qquad G\big|_{t<0} = 0\,,$$

and is given by the formula

$$G(x,t) = \mathrm{I}\frac{\theta(t)\delta\left(t^2 - \frac{|x|^2}{a^2}\right)}{2\pi a^3} = \mathrm{I}\theta(t)\frac{\delta\left(t - \frac{|x|}{a}\right)}{4\pi a^2 |x|}, \tag{3.8}$$

where I is the matrix.

The solution of (3.6), (3.7) may be found by

$$A = G * \left(\frac{1}{\varepsilon} j\right). \tag{3.9}$$

The explicit formulas for the solution of the Cauchy problem (1.13), (1.14) for the current density $j = \varepsilon\vec{e}\delta(x)\theta_1(t)$ will be obtained in this section.

3.1.1 Case (A1)

We have for the case (A1) the following formula

$$A = G * \left(\frac{1}{\varepsilon} j\right) = G_{(x,t)} * [\vec{e}\,\delta(x)\theta_1(t)] =$$

$$G_{(t)} * \vec{e}\,\theta_1(t) = \int_{-\infty}^{\infty} G(x, t-\tau)\vec{e}\,\theta_1(\tau)d\tau =$$

$$= \int_{-\infty}^{\infty} \theta(t-\tau)\theta_1(\tau)\vec{e}\,\frac{\delta\left(t - \tau - \frac{|x|}{a}\right)}{4\pi a^2|x|}d\tau$$

$$= \frac{1}{4\pi a^2 |x|}\vec{e}\,\theta_1\left(t - \frac{|x|}{a}\right). \tag{3.10}$$

We find below the expressions for $divA$ and $\nabla divA$. These are the following

$$div_x A = \frac{1}{4\pi a^2} div_x\left(\frac{\vec{e}}{|x|}\right)\theta_1\left(t-\frac{|x|}{a}\right) - \frac{1}{4\pi a^3}\left(\frac{x}{|x|^2}\vec{e}\right)\theta\left(t-\frac{|x|}{a}\right),$$

$$\nabla_x div_x A = \frac{1}{4\pi a^2}\nabla_x div_x\left(\frac{\vec{e}}{|x|}\right)\theta_1\left(t-\frac{|x|}{a}\right)$$

$$-\frac{1}{4\pi a^3} div_x\left(\frac{\vec{e}}{|x|}\right)\frac{x}{|x|}\theta\left(t-\frac{|x|}{a}\right)$$

$$-\frac{1}{4\pi a^2}\nabla_x\left(\frac{x}{|x|^2}\vec{e}\right)\theta\left(t-\frac{|x|}{a}\right) + \frac{\theta(t)}{4\pi a^4}\frac{x}{|x|}\left(\frac{x}{|x|^2}\vec{e}\right)\delta\left(t-\frac{|x|}{a}\right)$$

$$= \frac{1}{4\pi a^2}\nabla_x div_x\left(\frac{\vec{e}}{|x|}\right)\theta_1\left(t-\frac{|x|}{a}\right)$$

$$-\frac{1}{4\pi a^3}\left[div_x\left(\frac{\vec{e}}{|x|}\right)\frac{x}{|x|} + \nabla_x\left(\frac{x}{|x|^2}\vec{e}\right)\right]\theta\left(t-\frac{|x|}{a}\right).$$

$$+\frac{\theta(t)}{4\pi a^4}\frac{x}{|x|}\left(\frac{x}{|x|^2}\vec{e}\right)\delta\left(t-\frac{|x|}{a}\right)$$

The solution $\Phi = \nabla\varphi$ of the problem (22) may be found by

$$\Phi(x,t) = \theta(t)_{(t)} * a^2\nabla_x div_x A =$$

$$\frac{1}{4\pi a^2}\left(\frac{x}{|x|^2}\vec{e}\right)\frac{x}{|x|}\theta\left(t-\frac{|x|}{a}\right)$$

$$-\frac{1}{4\pi a}\left[div_x\left(\frac{\vec{e}}{|x|}\right)\frac{x}{|x|} + \nabla_x\left(\frac{x}{|x|^2}\vec{e}\right)\right]\theta_1\left(t-\frac{|x|}{a}\right).$$

$$+\frac{1}{4\pi}\nabla_x div_x\left(\frac{\vec{e}}{|x|}\right)\theta_2\left(t-\frac{|x|}{a}\right) \qquad (3.11)$$

Remark: Here we used the formulas

$$\theta(t)_{(t)} * \theta(t)\theta_k\left(t-\frac{|x|}{a}\right) = \int_{-\infty}^{\infty}\theta(t-z)\theta(z)\theta_K\left(z-\frac{|x|}{a}\right)dz =$$

$$\theta\left(t-\frac{|x|}{a}\right)\int_{\frac{|x|}{a}}^{t}\frac{\left(z-\frac{|x|}{a}\right)^k}{k!}dz = \theta\left(t-\frac{|x|}{a}\right)\int_0^{t-\frac{|x|}{a}}\frac{\tau^k}{k!}d\tau =$$

$$\theta\left(t-\frac{|x|}{a}\right)\frac{\left(t-\frac{|x|}{a}\right)^{k+1}}{(k+1)!} = \theta_{k+1}\left(t-\frac{|x|}{a}\right), k = 0,1,2,....$$

$$\theta(t)_{(t)} * \theta(t)\delta\left(t-\frac{|x|}{a}\right) = \int_{-\infty}^{\infty}\theta(t-z)\theta(z)\delta\left(z-\frac{|x|}{a}\right)dz$$

$$= \theta\left(t-\frac{|x|}{a}\right)$$

We find from (3.20)

$$\frac{\partial A}{\partial t} = \frac{1}{4\pi a^2}\frac{\vec{e}}{|x|}\theta\left(t-\frac{|x|}{a}\right).$$

Using (3.11), (3.12) we find the electric and magnetic intensity vectors E, H by

$$E = -\frac{1}{4\pi a^2}\frac{\vec{e}}{|x|}\theta\left(t-\frac{|x|}{a}\right) + \frac{1}{4\pi a^2|x|}\left(\frac{x}{|x|}\vec{e}\right)\frac{x}{|x|}\theta\left(t-\frac{|x|}{a}\right)$$
$$-\frac{1}{4\pi a}\left[div_x\left(\frac{\vec{e}}{|x|}\right)\frac{x}{|x|} + \nabla_x\left(\frac{x}{|x|^2}\vec{e}\right)\right]\theta_1\left(t-\frac{|x|}{a}\right)$$
$$+\frac{1}{4\pi}\nabla_x div_x\left(\frac{\vec{e}}{x}\right)\theta_2\left(t-\frac{|x|}{a}\right) \qquad \textbf{(3.12a)}$$

$$A = \frac{1}{4\pi a^2 |x|} e\theta_1\left(t - \frac{|x|}{a}\right), \qquad H = curl(A), \qquad \textit{(3.12b)}$$

and using the similar reasoning (or differentiation w.r.t. t formulas (3.10),(3.11),(3.12) we find for (A2) case (i.e. $j = \varepsilon\vec{e}\,\delta(x)\theta(t)$) formulas.

3.1.2 Case (A2)

$$A = \frac{1}{4\pi a^2 |x|}\vec{e}\,\theta\left(t - \frac{|x|}{a}\right) \qquad \textit{(3.13)}$$

$$\begin{aligned}\Phi(x,t) = {} & \frac{\theta(t)}{4\pi a^2}\left(\frac{x}{|x|^2}\vec{e}\right)\frac{x}{|x|}\delta\left(t - \frac{|x|}{a}\right) \\ & - \frac{1}{4\pi a}\left[div_x\left(\frac{\vec{e}}{|x|}\right)\frac{x}{|x|} + \nabla_x\left(\frac{x}{|x|^2}\vec{e}\right)\right]\theta\left(t - \frac{|x|}{a}\right) \\ & + \frac{1}{4\pi}\nabla_x div_x\left(\frac{\vec{e}}{|x|}\right)\theta_1\left(t - \frac{|x|}{a}\right)\end{aligned}$$

$$\begin{aligned}\vec{E}(x,t) = {} & \frac{\theta(t)}{4\pi a^2 |x|}\left[\left(\frac{x}{|x|}\vec{e}\right)\frac{x}{|x|} - \vec{e}\right]\delta(t - \frac{|x|}{a}) \\ & - \frac{1}{4\pi a}\left[div_x\left(\frac{\vec{e}}{|x|}\right)\frac{x}{|x|} + \nabla_x\left(\frac{x}{|x|^2}\vec{e}\right)\right]\theta\left(t - \frac{|x|}{a}\right) \\ & + \frac{1}{4\pi}\nabla_x div_x\left(\frac{\vec{e}}{x}\right)\theta_1\left(t - \frac{|x|}{a}\right)\end{aligned} \qquad \textit{(3.14a)}$$

$$A = \frac{1}{4\pi a^2 |x|} e\theta\left(t - \frac{|x|}{a}\right), \qquad H = curl(A), \qquad \textit{(3.14b)}$$

Applying the operator $\frac{\partial^2}{\partial t^2}$ to formulas (3.10), (3.11), (3.12) we obtain for the case (A3):

3.1.3 Case (A3)

$$A(x,t) = \frac{\theta(t)}{4\pi a^2 |x|} \vec{e}\,\delta\left(t - \frac{|x|}{a}\right) \qquad (3.15)$$

$$\Phi(x,t) = \frac{\theta(t)}{4\pi a^2}\left(\frac{x}{|x|^2}\vec{e}\right)\frac{x}{|x|}\delta'\left(t-\frac{|x|}{a}\right)$$
$$-\frac{\theta(t)}{4\pi a}\left[div_x\left(\frac{\vec{e}}{|x|}\right)\frac{x}{|x|} + \nabla_x\left(\frac{x}{|x|^2}\vec{e}\right)\right]\delta\left(t-\frac{|x|}{a}\right)$$
$$+\frac{1}{4\pi}\nabla_x div_x\left(\frac{\vec{e}}{|x|}\right)\theta\left(t-\frac{|x|}{a}\right),$$

$$\vec{E}(x,t) = \frac{\theta(t)}{4\pi a^2 |x|}\left[\left(\frac{x}{|x|}\vec{e}\right)\frac{x}{|x|} - \vec{e}\right]\delta'(t - \frac{|x|}{a}) \qquad (3.16)$$
$$-\frac{\theta(t)}{4\pi a}\left[div_x\left(\frac{\vec{e}}{|x|}\right)\frac{x}{|x|} + \nabla_x\left(\frac{x}{|x|^2}\vec{e}\right)\right]\delta\left(t-\frac{|x|}{a}\right)$$
$$+\frac{1}{4\pi}\nabla_x div_x\left(\frac{\vec{e}}{x}\right)\theta\left(t-\frac{|x|}{a}\right)$$

$$A = \frac{1}{4\pi a^2 |x|} e\delta\left(t - \frac{|x|}{a}\right), \qquad H = curl(A), \qquad (3.17)$$

Now we rearrange all the formulas above to generate the explicit form of the first component of vector E. Following are the three distinct explicit formulas for first three assumptions.

3.2 Modeling and Analysis

3.2.1 Assumption 1, E1

In assumption 1 we used the following approach $j = \varepsilon \vec{e} \delta(x) \theta_1(t)$. In the formula (3.12) we have the raw calculations for assumption 1.

In explicit form of the formula (3.12), for the first component of vector E is the following:

$$\begin{aligned} E_1 &= A1E_1P1 + (A1E_1P2 * A1E_1P3) \\ &+ (A1E_1P4 * A1E_1P5) + (A1E_1P6 * A1E_1P7) \end{aligned}, \qquad \textit{(3.18)}$$

where

$$A1E_1P1 = -\frac{1}{4\pi a_1^{\,2}} \frac{\vec{e}_1}{|x|} \theta\left(t - \frac{|x|}{a_1}\right),$$

$$A1E_1P2 = \frac{1}{4\pi a_1^{\,2}|x|} \sum_{j=1}^{3} \left(\frac{x_j \vec{e}_j}{|x|}\right) \frac{x_1}{|x|},$$

$$A1E_1P3 = \theta\left(t - \frac{|x|}{a_1}\right),$$

$$\begin{aligned} A1E_1P4 = -\frac{1}{4\pi a_1}\Bigg[&\frac{-(e_1x_1 + e_2x_2 + e_3x_3)}{|x|^3} \frac{x_1}{|x|} \\ &+ \frac{e_1|x|^2 - 2x_1^{\,2}e_1}{|x|^4} \\ &- \frac{2x_1x_2e_2}{|x|^4} - \frac{2x_1x_3e_3}{|x|^4} \Bigg], \end{aligned}$$

$$A1E_1P5 = \theta_1\left(t - \frac{|x|}{a_1}\right),$$

$$A1E_1P6 = \frac{1}{4\pi}\left(\frac{-e_1|x|^3 + 3x_1|x|(e_1x_1 + e_2x_2 + e_3x_3)}{|x|^6}\right)$$

$$A1E_1P7 = \theta_2\left(t - \frac{|x|}{a_1}\right)$$

Above we separate the formula (3.18) in to seven distinct parts. Each of these parts is examined respectively to generate its graph.

3.2.1.1 Drawing $A1E_1P1$:

Consider the task to draw the graph of the function

$$f(x) = A1E_1P1\_1 * A1E_1P1\_2 * A1E_1P1\_3 \qquad (3.19)$$

$$A1E_1P1\_1 = -\frac{\vec{e}_1}{4\pi a_1^{\,2}},$$

$$A1E_1P1\_2 = \frac{1}{|x|},$$

$$A1E_1P1\_3 = \theta\left(t - \frac{|x|}{a_1}\right).$$

, where

$x = (x_1, x_2, x_3) \in R^3$,

$t = 1$,

$x_3 = 0.7$,

$a_1 = 2.0$,

$e_1 = 1.0, e_2 = 2.0, e_3 = 4.0$

Analysis:

The graph (figure 3.1) below belongs to part 1 of the formula (3.18) above which is the function on (3.19). It's a frame at specific time point where $t = 1$. Because there is a Heaviside function multiplication used in the formula, we observe that every point is zero outside the front of the wave.

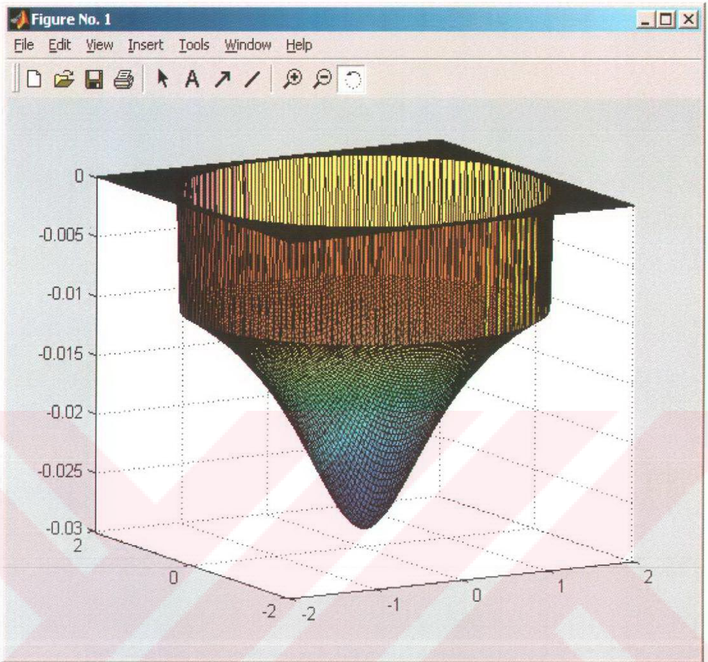

Figure 3.1: Modelling $A1E_1P1$

$x = (x_1, x_2, x_3) \in R^3$,

$t = 1$,

$x_3 = 0.7$,

$a_1 = 2.0$,

$e_1 = 1.0$,
$e_2 = 2.0$,
$e_3 = 4.0$

Conclusion:

In the formula (3.19), $A1E_1P1\_1$ is constant, $A1E_1P1\_3$ is Heaviside function. As explained before outside the front is zero because of $A1E_1P1\_3$. $A1E_1P1\_2$ is variadic, for small values of x we get bigger peaks.

3.2.1.2 Drawing $A1E_1P2$:

Consider the task to draw the graph of the function

$$f(x) = A1E_1P2\_1 * A1E_1P2\_2 * A1E_1P2\_3, \qquad (3.20)$$

$$A1E_1P2\_1 = \frac{1}{4\pi a_1^{\,2}|x|},$$

$$A1E_1P2\_2 = \sum_{j=1}^{3}\left(\frac{x_j\bar{e}_j}{|x|}\right),$$

$$A1E_1P2\_3 = \frac{x_1}{|x|}.$$

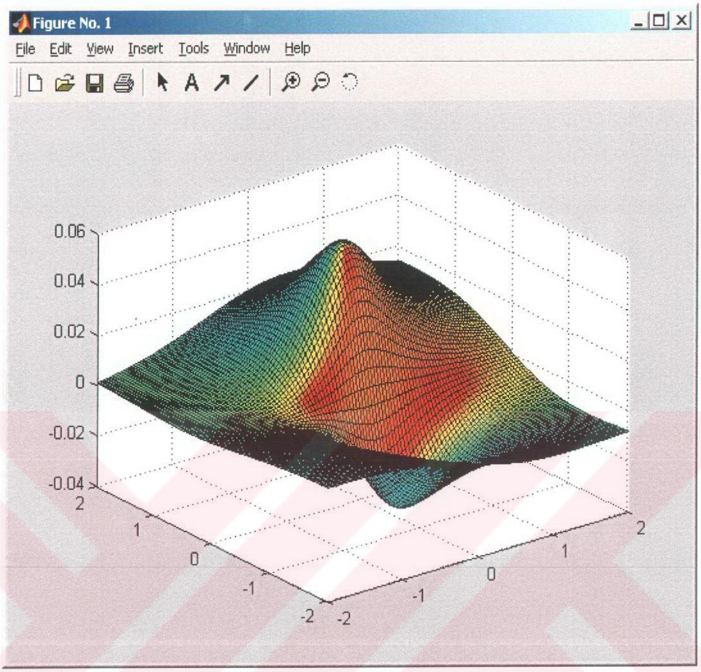

Figure 3.2: Modeling $A1E_1P2$

$x = (x_1, x_2, x_3) \in R^3$,

$x_3 = 0.7$,

$a_1 = 2.0$,

$e_1 = 1.0$,

$e_2 = 2.0$, .

$e_3 = 4.0$

Analysis:

In the formula (3.20) of the function we can simplify the components to the following two distinct representations

$$f(x) = \frac{1}{|x|} \tag{3.21}$$

$$f(x) = \frac{x_1}{|x|} \tag{3.22}$$

In the former formula (3.21) graph is a conical in the positive y axis, But in the latter formula (3.22), nominator is x_1 and this variable is changing in negative to positive range of values. So the resulting graph includes 2 distinct conical one in positive y axis other in negative y axis.

Conclusion:

Without their constants $A1E_1P2\_1$, $A1E_1P2\_2$ and $A1E_1P2\_3$ parts of the formula (3.20) can be represented any one of the formula (3.21) or (3.22). Because these parts are in multiplication we can summarize them in the form of formula (3.22). And the resulting graph will be as in (figure 3.1) as expected.

3.2.1.3 Drawing Part $A1E_1P2$ & $A1E_1P3$ together:

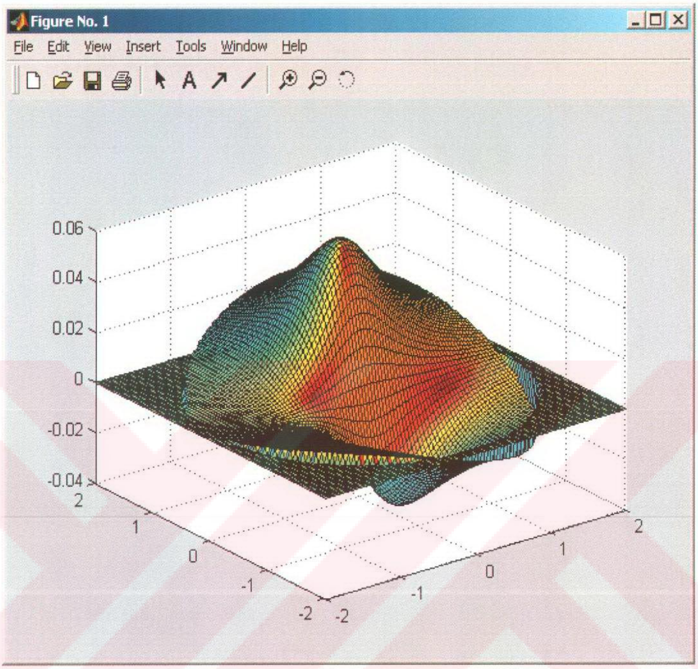

Figure 3.3: Modelling $A1E_1P2$ **&** $A1E_1P3$

Parameters same as in Fig 1 and Fig 2

Conclusion:

Above is the graph of $A1E_1P2$ and $A1E_1P3$ of the formula (3.18) together. In the graph above outside the front is zero. This is because Heaviside function generates 0 values for outside of the front and $A1E_1P2$ and $A1E_1P3$ are multiplied together as a result from the point of the front of the wave everywhere is zero.

3.2.1.4 Drawing $A1E_1P4$:

$$f(x) = -\frac{1}{4\pi a_1}\left[\frac{-(e_1x_1 + e_2x_2 + e_3x_3)}{|x|^3}\frac{x_1}{|x|} + \frac{e_1|x|^2 - 2x_1{}^2e_1}{|x|^4} - \frac{2x_1x_2e_2}{|x|^4} - \frac{2x_1x_3e_3}{|x|^4}\right] \tag{3.23}$$

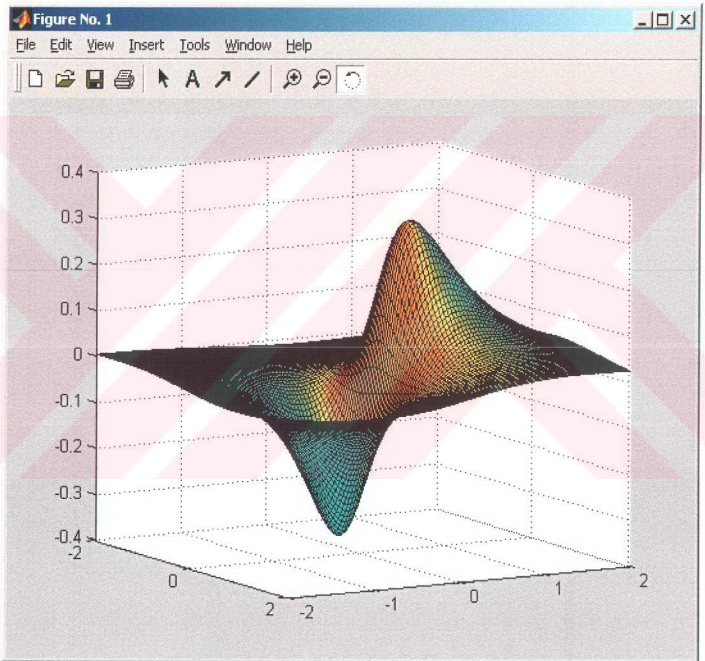

Figure 3.4: Modelling $A1E_1P4$

$x = (x_1, x_2, x_3) \in R^3$,

$x_3 = 0.7$,

$a_1 = 2.0$, $e_1 = 1.0$, $e_2 = 2.0$, $e_3 = 4.0$

Analysis:

In the formula (3.23) above, there is no special function call (like Dirac Delta or Heaviside) made and this formula can be reduced to formula (3.22) where the similar operation applied in Part 2 of formula (3.28).

Conclusion:

As explained in 3.2.2 reduced formula is similar to formula (3.22). Denominator of the reduced formula is constant. Because the absolute value of the denominator is taken, result of the reduction is positive. But nominator is summation of some constant values and some combination of x_1, x_2, x_3. Because x_1, x_2 ranges from positive to negative values the resulting graph has two cones. One's top is in negative region others in positive. As a result figure 3.4 is expected graph that we observed.

3.2.1.5 Drawing $A1E_1P5$:

Consider the task to draw the graph of the function

$\theta_1\left(t - \frac{|x|}{a_1}\right)$, where the definition of the function is defined in formula (3.1a), (3.1b) *(3.24)*

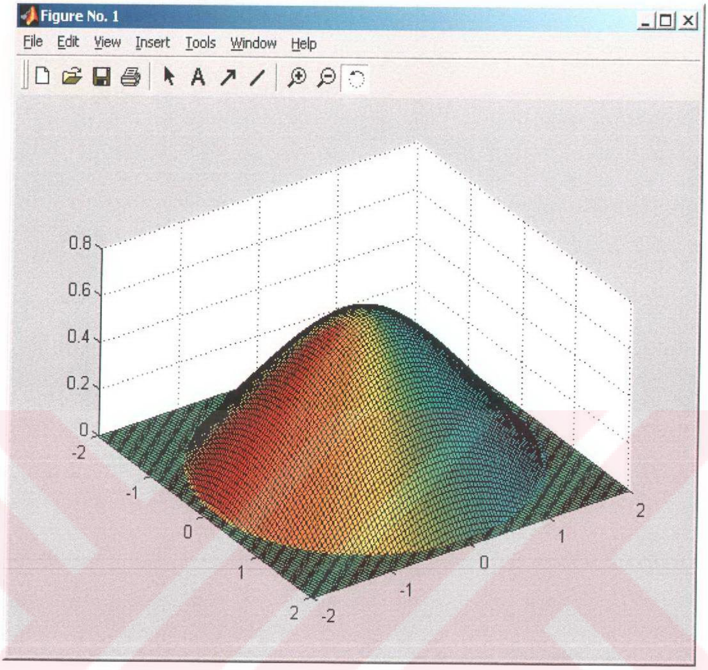

Figure 3.5: Modelling $A1E_1P5$

$x = (x_1, x_2, x_3) \in R^3$,

$t = 1$,

$a_1 = 2.0$

Analysis:

In the formula (3.23) variable t is constant and its value is set to 1. |x| has dynamic value where x1 and x2 are in the range of [-2, 2] but x3 is constant and has the value of 0.7. And also a1 has the value of 2.0. According to these constants and value range resulting value of the parameter of Heaviside function is always positive and for large values of x vector components, result get smaller, for smaller values it gets larger.

Conclusion:

From the perspective of the analysis above, resulting picture figure 3.5 is as expected. There is only one cone in the origin.

3.2.1.6 Drawing Part $A1E_1P4$ & $A1E_1P5$ together:

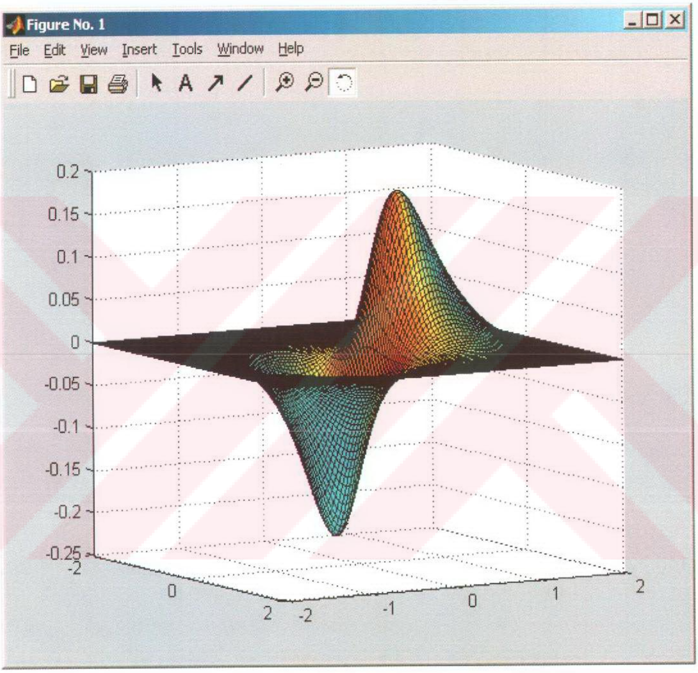

Figure 3.6: Modelling $A1E_1P4$ **&** $A1E_1P5$

Conclusion:

In the figure 3.6 above multiplication of part 4 and part 5 can be observed. In part 5 there is a cone inside the front, elsewhere is zero. Value of largest part of the cone is near to 0.4 so the values inside the front ranges between 0 and 0.4 then the multiplication of part 4 and part 5 produces same type of graphic as in part 4 but the scale at part 4 in y axis becomes smaller then before as we see in the figure above.

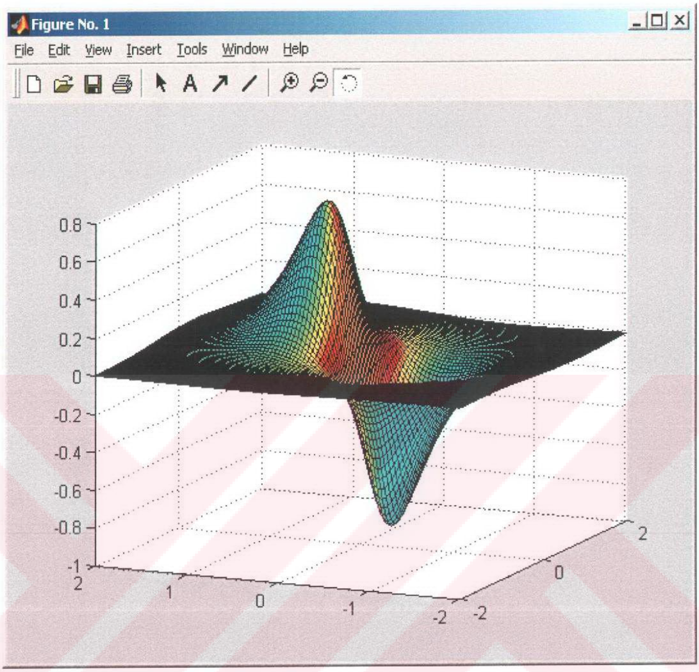

Figure 3.7: Modelling Part6 of Assumtion1 (E1)

Conclusion:

In the figure above part 6 generates approximately the some type of graph as in part 4. So the expected graph is similar to the one in figure 4.

3.2.1.7 Drawing $A1E_1P7$:

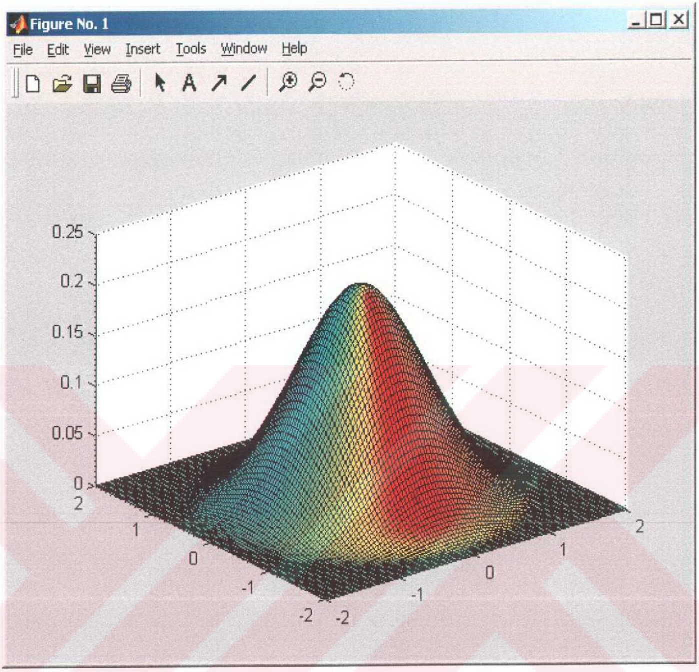

Figure 3.8: Modelling $A1E_1P7$

$x = (x_1, x_2, x_3) \in R^3$,

$t = 1$,

$a_1 = 2.0$

Conclusion:

$\theta_2(\Gamma) = \theta(\Gamma)\frac{\Gamma^2}{2}$ is the explicit form of part 7. Graph of θ_2 is similar to θ_1, except the scalar value that we multiplied is larger then the one that we used in θ_1. So the peak inside the front is sharper.

3.2.1.8 Drawing Part $A1E_1P6$ & $A1E_1P7$ together:

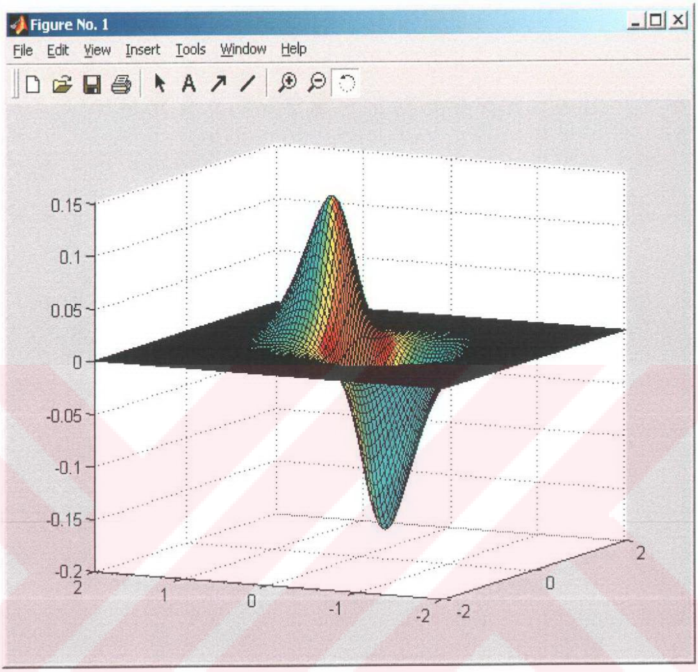

Figure 3.9: Modeling $A1E_1P6$ **&** $A1E_1P7$

Conclusion:

Combination of part 6 and part 7 is similar to part 4 and part 5. When we reduce the formulas without constants we will get the formulas like multiplication of formula (3.22) and part 5. So the resulting figure is as we expected.

3.2.1.9 Drawing All Parts Together:

Following is the graph of all parts that we draw separately up to now. We don't have any hesitation about the correctness of the distinct parts that we have investigated above. From the point of those analyses of distinct parts we can assume that combination all those parts are with resulting figure (10) is correct.

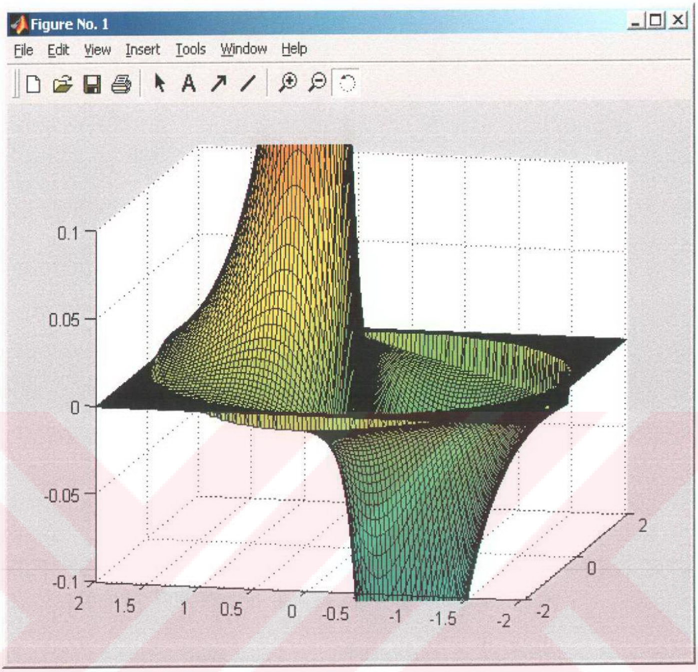

Figure 3.10: Modeling Assumtion1 E component

3.2.2 Assumption 1, H1

In formula (3.12b) we have found the value of A and H. Now we will observe the first component of H, where

$$\mathrm{H}_1 = \frac{\partial A_3}{\partial x_2} - \frac{\partial A_2}{\partial x_3}.$$

From the definition of $\mathbf{A}$ and $\mathbf{H}_1$ we get the following formula

$$H_1 = \frac{e_3}{4\pi a_3^{\,2}} \frac{\partial}{\partial x_2}\left(\frac{\theta_1\left(t - \frac{|x|}{a_3}\right)}{|x|} \right) - \frac{e_2}{4\pi a_2^{\,2}} \frac{\partial}{\partial x_3}\left(\frac{\theta_1\left(t - \frac{|x|}{a_2}\right)}{|x|} \right)$$

Now we translate the formula above, to the following explicit one. Without any derivations included, we can draw the graph for the first component $\mathbf{H}_1$ of $\mathbf{H}$.

$$H_1 = A1H_1P1 - A1H_1P2, \qquad (3.25)$$

$$A1H_1P1 = \frac{e_3}{4\pi a_3^{\,2}} \left(\frac{\theta\left(t - \frac{|x|}{a_3}\right)\frac{-x_2}{a_3} - \frac{x_2}{|x|}\theta_1\left(t - \frac{|x|}{a_3}\right)}{|x|^2} \right)$$

$$A1H_1P2 = \frac{e_2}{4\pi a_2^{\,2}} \left(\frac{\theta\left(t - \frac{|x|}{a_2}\right)\frac{-x_3}{a_2} - \frac{x_3}{|x|}\theta_1\left(t - \frac{|x|}{a_2}\right)}{|x|^2} \right)$$

3.2.2.1 Drawing $A1H_1P1$:

We will analyze only the part 1 of the formula (3.25). Because when we take the x_3 equal to 0, part 2 will be zero for all other range of variables. From this point, there is no need to analyze part 2.

Analysis:

Following is the part 1 of the formula (3.25):

$$f(x) = \frac{e_3}{4\pi a_3^{\,2}} \left(\frac{\theta\left(t - \frac{|x|}{a_3}\right)\frac{-x_2}{a_3} - \frac{x_2}{|x|}\theta_1\left(t - \frac{|x|}{a_3}\right)}{|x|^2} \right)$$

When we reduce the constants and simplify the similar variable operations to uniform, it will be easier to make some estimation about the resulting graph.

Conclusion:

Resulting simplified formula will look like as the function below:

$$f(x) = \frac{x_2}{|x|}\theta_1\left(t - \frac{|x|}{a_3}\right)$$

This formula is the combination of the formulas that we draw in (figure 3.2) and (figure 3.5). So the resulting graph is as in (figure 3.11) that we can guess from the simplified version above.

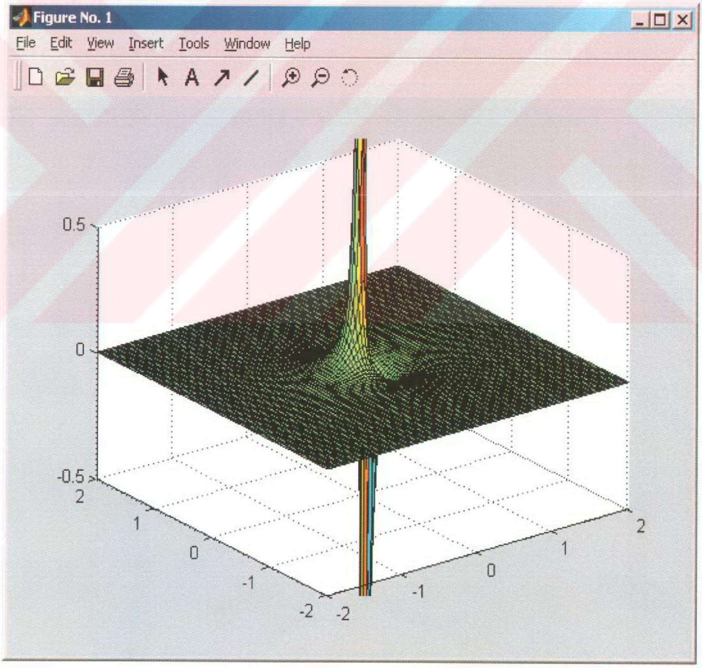

Figure 3.11: Modeling $A1H_1P1$

$$x=(x_1,x_2,x_3)\in R^3,$$
$$t=1,$$
$$a_3=8.0,$$
$$e_3=8.0.$$

3.2.3 Assumption 2, E1

In formula (3.14a) we have the formula in raw format. Explicit form of this formula is:

$$E_1=(A2E_1P1*A2E_1P2*A2E_1P3)-(A2E_1P4*A2E_1P5)+(A2E_1P6*A2E_1P7)$$, where

$$A2E_1P1=\frac{\theta(t)}{4\pi a_1^{\,2}|x|},$$

$$A2E_1P2=\left[\sum_{j=1}^{3}\left(\frac{x_j\vec{e}_j}{|x|}\right)\frac{x_1}{|x|}-e_1\right],$$

$$A2E_1P3=\delta\left(t-\frac{|x|}{a_1}\right),$$

$$A2E_1P4=\frac{1}{4\pi a_1}\left[\frac{-(e_1x_1+e_2x_2+e_3x_3)}{|x|^3}\frac{x_1}{|x|}+\frac{e_1|x|^2-2x_1^{\,2}e_1}{|x|^4}-\frac{2x_1x_2e_2}{|x|^4}-\frac{2x_1x_3e_3}{|x|^4}\right],$$

$$A2E_1P5=\theta\left(t-\frac{|x|}{a_1}\right),$$

$$A2E_1P6=\frac{1}{4\pi}\left(\frac{-e_1|x|^3+3x_1|x|(e_1x_1+e_2x_2+e_3x_3)}{|x|^6}\right),$$

$$A2E_1P7=\theta_1\left(t-\frac{|x|}{a_1}\right).$$

Now we will draw the pieces of assumption 2 separately. In the following headings each of these parts will be drawn except the ones which are very similar to the formulas that we draw in assumption 1.

3.2.3.1 Drawing $A2E_1P1$:

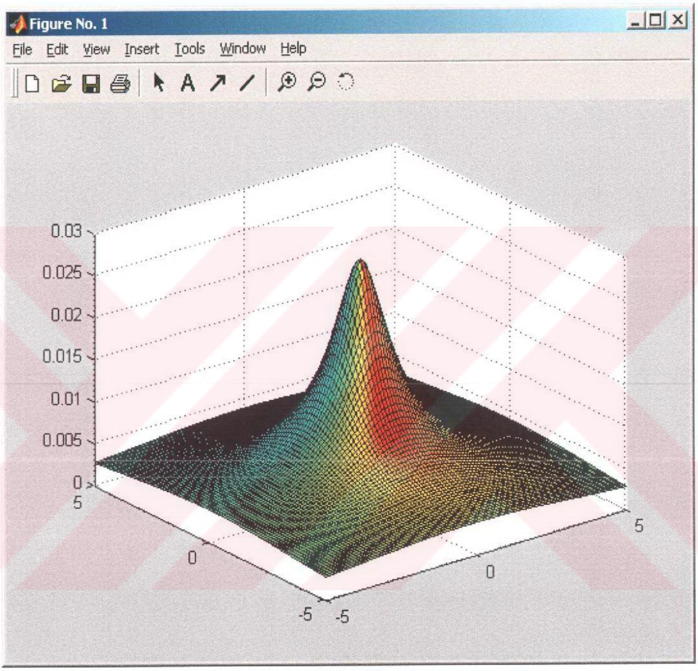

Figure 3.12: Modeling $A2E_1P1$

Time	1
Epsilon	$\frac{1}{2^{15}}$

3.2.3.2 Drawing $A2E_1P2$:

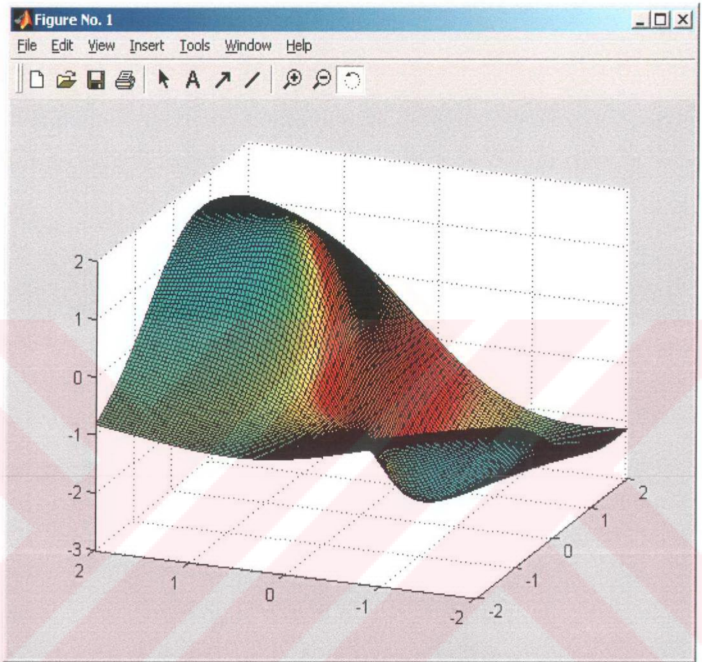

Figure 3.13: Modeling $A2E_1P2$

Time	1
Epsilon	$\frac{1}{2^{15}}$

3.2.3.3 Drawing $A2E_1P1$ & $A2E_1P2$:

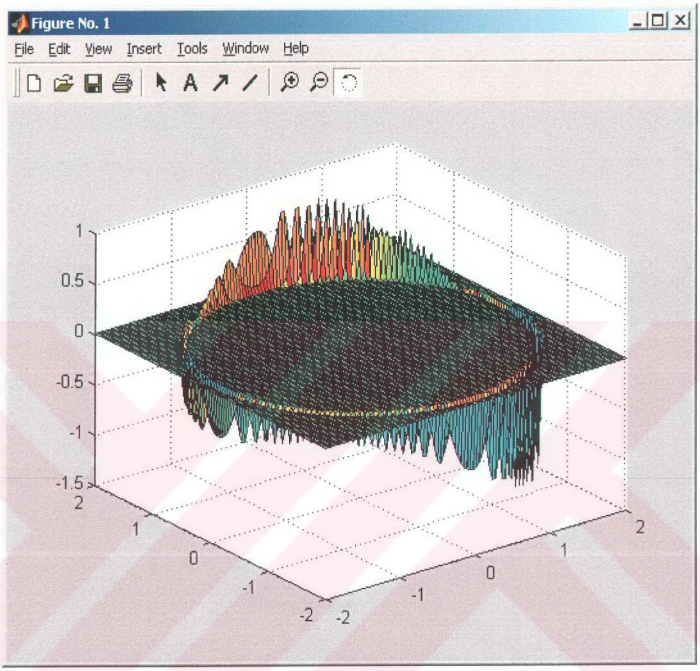

Figure 3.14: Modeling $A2E_1P1$ **&** $A2E_1P2$

Time	1
Epsilon	$\frac{1}{2^{15}}$

3.2.4 Assumption 2, H1

From the formula (3.14b) we have

$$\mathrm{H}_1 = \frac{\partial A_3}{\partial x_2} - \frac{\partial A_2}{\partial x_3}$$

When we make substitution and rewrite the formula we get

$$\mathrm{H}_1 = \frac{e_3}{4\pi a_3^{\,2}} \frac{\partial}{\partial x_2}\left(\frac{\theta\left(t - \frac{|x|}{a_3}\right)}{|x|} \right) - \frac{e_2}{4\pi a_2^{\,2}} \frac{\partial}{\partial x_3}\left(\frac{\theta\left(t - \frac{|x|}{a_2}\right)}{|x|} \right)$$

We will rewrite the formula above in the explicit form and divide it into two pieces. These will simplify the controlling mechanism of formulas and we will be sure that error proof formulas generated. Below is the explicit form with distinct parts and their corresponding modeling.

$$\mathrm{H}_1 = A2H_1P1 - A2H_1P1,$$

$$A2H_1P1 = \frac{e_3}{4\pi a_3^{\,2}} \left(\frac{\delta\left(t - \frac{|x|}{a_3}\right)\frac{-x_2}{a_3} - \frac{x_2}{|x|}\theta\left(t - \frac{|x|}{a_3}\right)}{|x|^2} \right),$$

$$A2H_1P2 = \frac{e_2}{4\pi a_2^{\,2}} \left(\frac{\delta\left(t - \frac{|x|}{a_2}\right)\frac{-x_3}{a_2} - \frac{x_3}{|x|}\theta\left(t - \frac{|x|}{a_2}\right)}{|x|^2} \right).$$

3.2.4.1 Drawing $A2H_1P1$:

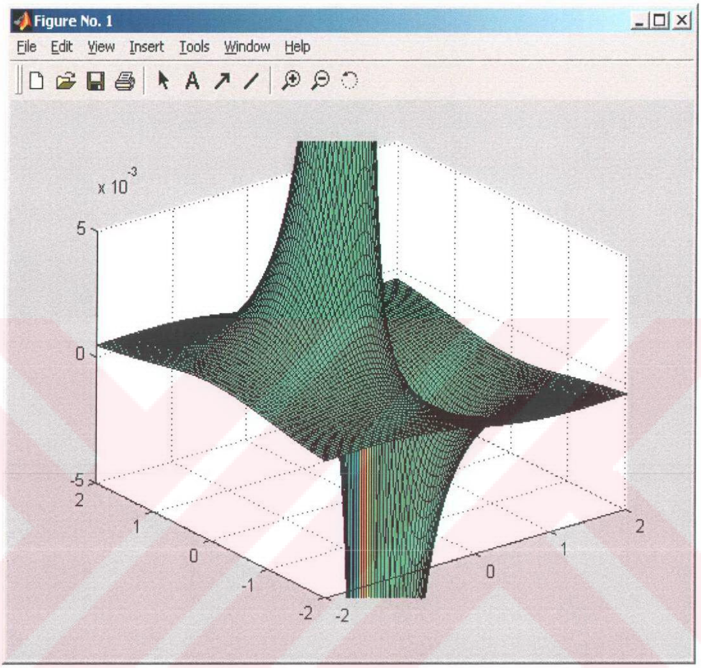

Figure 3.15: Modeling $A2H_1P1$

Time	1
Epsilon	$\frac{1}{2^{15}}$

3.2.5 Assumption 3, E1

From the formula (3.16a) we have the explicit form as:

$$\mathrm{E}_1(\mathrm{x},\mathrm{t}) = (A3E_1P1 * A3E_1P2) - A3E_1P3 + A3E_1P4,$$

$$A3E_1P1 = \frac{\theta(t)}{4\pi a_1^{\,2}|x|}\left[\sum_{j=1}^{3}\left(\frac{x_j\vec{e}_j}{|x|}\right)\frac{x_1}{|x|} - e_1\right],$$

$$A3E_1P2 = \delta'\left(t - \frac{|x|}{a_1}\right),$$

$$A3E_1P3 = \frac{1}{4\pi a_1}\left[\frac{-(e_1x_1 + e_2x_2 + e_3x_3)}{|x|^3}\frac{x_1}{|x|}\right.$$

$$\left. + \frac{e_1|x|^2 - 2x_1^{\,2}e_1}{|x|^4} - \frac{2x_1x_2e_2}{|x|^4} - \frac{2x_1x_3e_3}{|x|^4}\right]\delta\left(t - \frac{|x|}{a_1}\right),$$

$$A3E_1P4 = \frac{1}{4\pi}\left(\frac{-e_1|x|^3 + 3x_1|x|(e_1x_1 + e_2x_2 + e_3x_3)}{|x|^6}\right)\theta\left(t - \frac{|x|}{a_1}\right).$$

As we did in the previous assumptions, we divide the whole formula into small pieces. Below we will explore each of these pieces separately. Their related modeling and parameters are given in the headings below.

3.2.5.1 Drawing $A3E_1P2$:

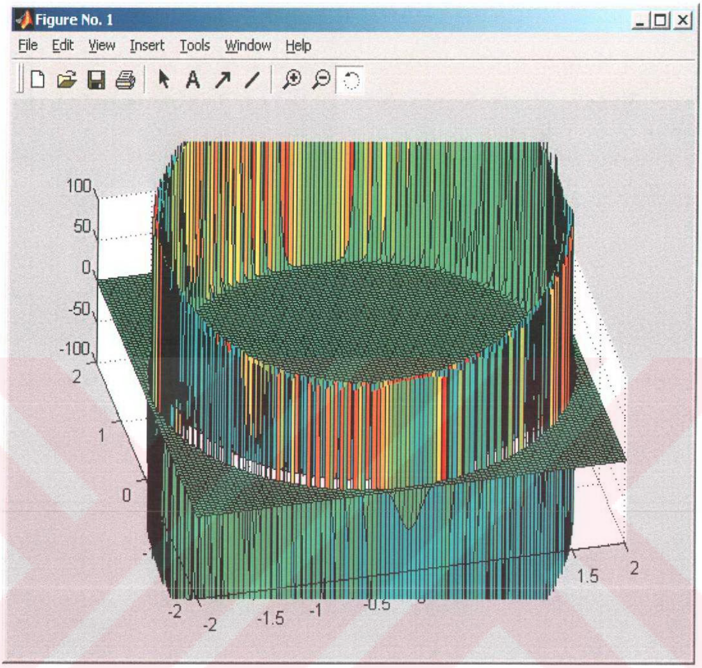

Figure 3.16: Modeling $A3E_1P2$

Time	1
Epsilon	$\frac{1}{2^{15}}$

3.2.5.2 Drawing $A3E_1P1$ & $A3E_1P2$ together:

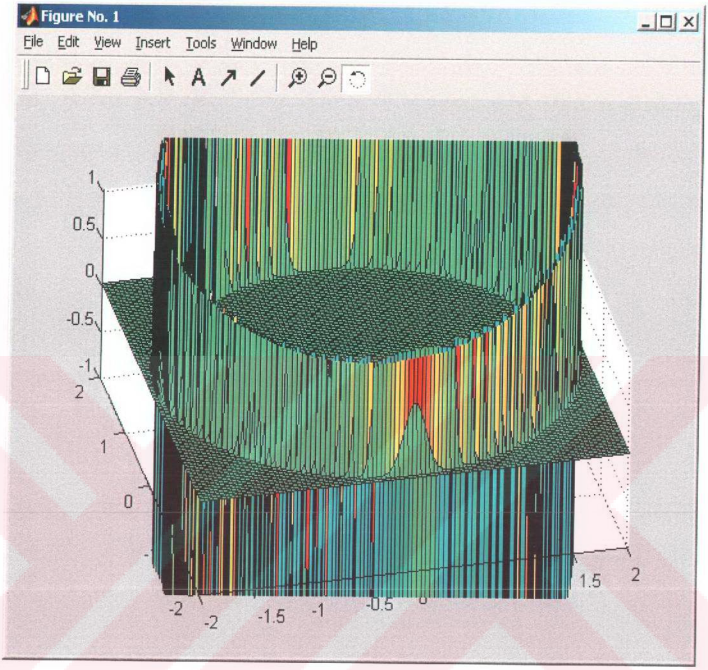

Figure 3.17: Modeling $A3E_1P1$ **&** $A3E_1P2$

Time	1
Epsilon	$\frac{1}{2^{15}}$

3.2.6 Assumption 3, H1

From the formula (3.16b) we have

$$\mathrm{H}_1 = \frac{\partial A_3}{\partial x_2} - \frac{\partial A_2}{\partial x_3}$$

$$\mathrm{H}_1 = \frac{e_3}{4\pi a_3^{\,2}} \frac{\partial}{\partial x_2}\left(\frac{\delta\left(t - \frac{|x|}{a_3}\right)}{|x|} \right) - \frac{e_2}{4\pi a_2^{\,2}} \frac{\partial}{\partial x_3}\left(\frac{\delta\left(t - \frac{|x|}{a_2}\right)}{|x|} \right)$$

$$\mathrm{H}_1 = A3H_1P1 - A3H_1P2,$$

As we did in previous headings we rewrite the formula in explicit form and divide the formula into more manageable pieces.

$$A3H_1P1 = \frac{e_3}{4\pi a_3^{\,2}} \left(\frac{ \frac{\delta\left(t - \frac{|x|}{a_3}\right)}{\left(t - \frac{|x|}{a_3}\right)} \frac{x_2}{a_3} - \frac{x_2}{|x|}\delta\left(t - \frac{|x|}{a_3}\right) }{|x|^2} \right),$$

$$A3H_1P2 = \frac{e_2}{4\pi a_2^{\,2}} \left(\frac{ \frac{\delta\left(t - \frac{|x|}{a_2}\right)}{\left(t - \frac{|x|}{a_3}\right)} \frac{x_3}{a_2} - \frac{x_3}{|x|}\delta\left(t - \frac{|x|}{a_2}\right) }{|x|^2} \right).$$

$$A3H_1P2 = \frac{e_2}{4\pi a_2^{\,2}} \left(\frac{\dfrac{\delta\left(t - \dfrac{|x|}{a_3}\right)}{\left(t - \dfrac{|x|}{a_3}\right)} \dfrac{x_3}{a_2} - \dfrac{x_3}{|x|}\,\delta\left(t - \dfrac{|x|}{a_2}\right)}{|x|^2} \right).$$

Here we don't take second part in consideration because we take the constant x_3 equal to zero. These make the second part also zero for all conditions of other variables.

3.2.6.1 Drawing $A3H_1P1$ a:

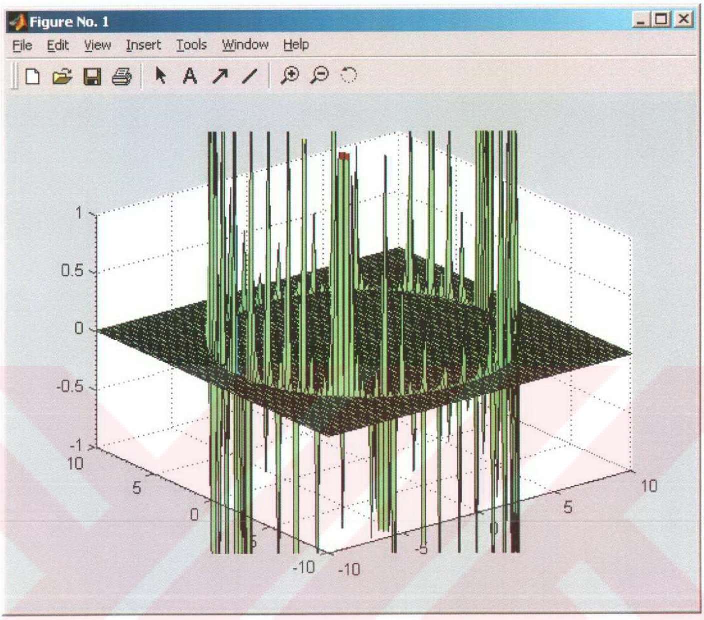

Figure 3.18: Modeling $A3H_1P1$

Time	1
Epsilon	$\frac{1}{2^{15}}$

3.2.6.2 Drawing $A3H_1P1$ b:

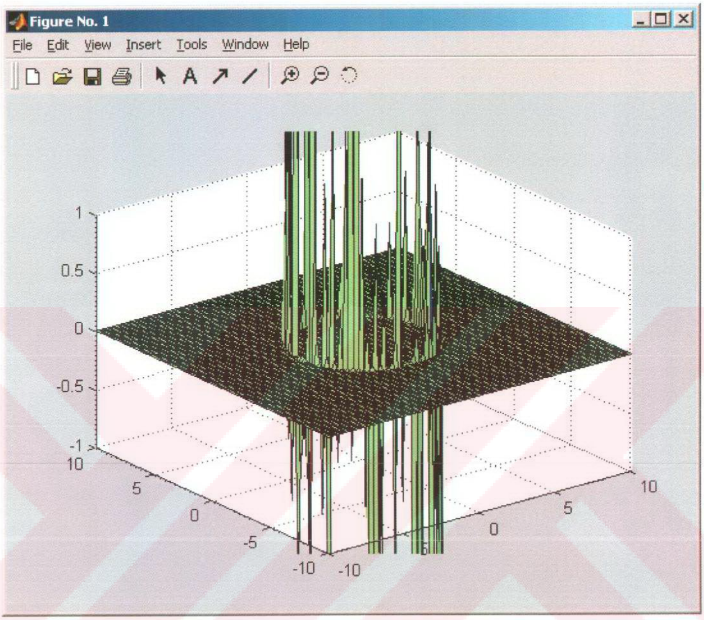

Figure 3.19: Modeling $A3H_1P1$

Time	0,5
Epsilon	$\frac{1}{2^{15}}$

CHAPTER FOUR

THE CAUCHY PROBLEM FOR MAXWELL'S FOR THE ANISOTROPIC MEDIUM

4.3 Anisotropic Simple Case

In anisotropic simple case we take the electromagnetic constant ε as diagonal matrix. Other constants and assumptions that we used in this case are defined below.

$$\varepsilon = \begin{pmatrix} \varepsilon_1 & 0 & 0 \\ 0 & \varepsilon_2 & 0 \\ 0 & 0 & \varepsilon_3 \end{pmatrix}, \qquad \sigma = 0\,,\; j = \varepsilon\vec{e}\,\delta(x)\theta(t)\,, \qquad f(t)\big|_{t<0} = 0$$

Equations (1.8) and (1.8a) may be written as

$$\frac{\partial^2 A_i}{\partial t^2} - a_i^{\,2}\Delta A_i + \mu a_i^{\,2}\sigma_i\frac{\partial A_i}{\partial t} = \mu a_i^{\,2} j\,, \qquad A\big|_{t<0} = 0\,, \qquad i = 1,2,3$$

$$a_i^{\,2} = \frac{1}{\mu\varepsilon_i}$$

and

$$\frac{\partial}{\partial t}\phi_i + \sigma_i\phi_i = a_i^{\,2}\frac{\partial}{\partial x_i}divA\,, \qquad \phi_i\big|_{t<0} = 0\,, \qquad i = 1,2,3$$

for $j = \vec{e}\,\delta(x) f(t)$, $\sigma = 0$ we have

$$\frac{\partial^2 A}{\partial t^2} - a_i^{\,2} \Delta A_i = e_i \delta(x) f(t), \qquad i = 1,2,3, \qquad A_i\big|_{t<0} = 0$$

The solution of this problem is given by the formula

$$\begin{aligned} A_i &= \iiiint\limits_{R^4} \varepsilon_i(x-\xi, t-\tau)\delta(\xi) f(t) dt d\xi \\ &= \int\limits_R \varepsilon_i(x, t-\tau) f(t) dt \\ &= \int_{-\infty}^{+\infty} \frac{\theta(t-\tau)}{4\pi a_i^{\,2}|x|} e_i \delta\left(t - \tau - \frac{|x|}{a_i}\right) f(\tau) d\tau \\ &= \frac{e_i}{4\pi a_i^{\,2}|x|} f\left(t - \frac{|x|}{a_i}\right) \end{aligned}$$

where $\varepsilon_i(x,t) = \dfrac{e_i}{4\pi a_i^{\,2}} \delta\left(t - \dfrac{|x|}{a_i}\right)$

Let,

$$f(t) = \begin{cases} \sin(\omega t), t \geq 0 \\ 0, t < 0 \end{cases} \qquad\qquad j = \frac{1}{\varepsilon} e \delta(x) f(t)$$

then

$$f'(t) = \begin{cases} \omega \cos(\omega t), t \geq 0 \\ 0, t < 0 \end{cases}, \quad f''(t) = \begin{cases} -\omega^2 \sin(\omega t), t \geq 0 \\ 0, t < 0 \end{cases}$$

We have for $div_x A$ and $\dfrac{\partial}{\partial x_i} div_x A$

Here we can calculate the

Here we can calculate the

$$div_x A = \frac{\theta(t)}{4\pi}\sum_{j=1}^{3}\left[\frac{\partial}{\partial x_j}\left(\frac{e_j}{a_j^2|x|}\right)f\left(t-\frac{|x|}{a_j}\right)-\frac{e_j x_j}{a_j^3|x|^2}f'\left(t-\frac{|x|}{a_j}\right)\right]$$

$$\frac{\partial}{\partial x_i}div_x A = \frac{\theta(t)}{4\pi}\left\{\sum_{j=1}^{3}\frac{e_j}{a_j^2}\frac{\partial^2}{\partial x_i \partial x_j}\left(\frac{1}{|x|}\right)f\left(t-\frac{|x|}{a_j}\right)\right.$$
$$-\sum_{j=1}^{3}\frac{e_j}{a_j^3}\left[\frac{\partial}{\partial x_j}\left(\frac{1}{|x|}\right)\left(\frac{x_i}{|x|}\right)\right.$$
$$\left.+\frac{\partial}{\partial x_i}\left(\frac{x_j}{|x|^2}\right)\right]f'\left(t-\frac{|x|}{a_j}\right)$$
$$\left.+\sum_{j=1}^{3}\frac{e_j}{a_j^4}\left(\frac{x_j}{|x|}\right)\left(\frac{x_i}{|x|}\right)\frac{1}{|x|}f''\left(t-\frac{|x|}{a_j}\right)\right\}$$

We find

$$\phi_i(x,t) = \theta(t)_{(t)} * a_i^2\frac{\partial}{\partial x_i}div_x A$$
$$= \int_{-\infty}^{+\infty}\theta(t-\tau)a_i^2\frac{\partial}{\partial x_i}div_x A(x,\tau)d\tau$$

$$\phi_i = \frac{a_i^2}{4\pi}\sum_{j=1}^{3}\frac{e_j}{a_j^2}\theta\left(t-\frac{|x|}{a_j}\right)\left\{\omega\frac{1}{a_j^2}\left(\frac{x_j}{|x|}\right)\left(\frac{x_i}{|x|}\right)\frac{1}{|x|}\right.$$
$$+\frac{\partial^2}{\partial x_i \partial x_j}\left(\frac{1}{|x|}\right)\left[-\frac{1}{\omega}\cos\left[\omega\left(t-\frac{|x|}{a_j}\right)\right]+\frac{1}{\omega}\right]$$
$$-\frac{1}{a_j}\left(\frac{\partial}{\partial x_j}\left(\frac{1}{|x|}\right)\left(\frac{x_i}{|x|}\right)+\frac{\partial}{\partial x_i}\left(\frac{x_j}{|x|^2}\right)\right)\sin\left[\omega\left(t-\frac{|x|}{a_j}\right)\right]$$
$$\left.+\frac{1}{a_j^2}\left(\frac{x_j}{|x|}\right)\left(\frac{x_i}{|x|}\right)\frac{1}{|x|}\left[\omega\cos\left[\omega\left(t-\frac{|x|}{a_j}\right)\right]-\omega\right]\right\}$$

Explicit form of the function above has two different form according to the possibility of the value of i and j.

First form, where $i = j$

$$
\begin{aligned}
\phi_i = \frac{a_i^{\,2}}{4\pi} \sum_{j=1}^{3} \frac{e_j}{a_j^{\,2}} \theta\left(t - \frac{|x|}{a_j}\right) & \left\{ \omega \frac{1}{a_j^{\,2}} \left(\frac{x_j}{|x|}\right)\left(\frac{x_i}{|x|}\right)\frac{1}{|x|} \right. \\
& + \left(-\frac{1}{|x|^3} + \frac{3x_i x_j}{|x|^5}\right)\left[-\frac{1}{\omega}\cos\left[\omega\left(t - \frac{|x|}{a_j}\right)\right] + \frac{1}{\omega}\right] \\
& - \frac{1}{a_j}\left(\left(\frac{-x_j}{|x|^3}\right)\left(\frac{x_i}{|x|}\right) + \left(\frac{|x|^2 - 2x_i x_j}{|x|^4}\right)\right)\sin\left[\omega\left(t - \frac{|x|}{a_j}\right)\right] \\
& \left. + \frac{1}{a_j^{\,2}}\left(\frac{x_j}{|x|}\right)\left(\frac{x_i}{|x|}\right)\frac{1}{|x|}\left[\omega\cos\left[\omega\left(t - \frac{|x|}{a_j}\right)\right] - \omega\right]\right\}
\end{aligned}
\qquad \textit{(4.1)}
$$

Second form, where $i \neq j$

$$
\begin{aligned}
\phi_i = \frac{a_i^{\,2}}{4\pi} \sum_{j=1}^{3} \frac{e_j}{a_j^{\,2}} \theta\left(t - \frac{|x|}{a_j}\right) & \left\{ \omega \frac{1}{a_j^{\,2}} \left(\frac{x_j}{|x|}\right)\left(\frac{x_i}{|x|}\right)\frac{1}{|x|} \right. \\
& + \left(-\frac{3x_i x_j}{|x|^5}\right)\left[-\frac{1}{\omega}\cos\left[\omega\left(t - \frac{|x|}{a_j}\right)\right] + \frac{1}{\omega}\right] \\
& - \frac{1}{a_j}\left(\left(\frac{-x_i}{|x|^3}\right)\left(\frac{x_i}{|x|}\right) - \frac{2x_i x_j}{|x|^4}\right)\sin\left[\omega\left(t - \frac{|x|}{a_j}\right)\right] \\
& \left. + \frac{1}{a_j^{\,2}}\left(\frac{x_j}{|x|}\right)\left(\frac{x_i}{|x|}\right)\frac{1}{|x|}\left[\omega\cos\left[\omega\left(t - \frac{|x|}{a_j}\right)\right] - \omega\right]\right\}
\end{aligned}
\qquad \textit{(4.2)}
$$

Up to now we find the $\frac{\partial}{\partial x_i} div_x A$ and ϕ_i seperately. Here we take E(x, t) as

$$E(x,t) = -\frac{\partial A_i}{\partial t} + \phi_i ,$$

When we make substitution in the above formula, we get:

$$\frac{\partial A_i}{\partial t} = \frac{e_i}{4\pi a_i^{\,2}|x|}\theta\left(t - \frac{|x|}{a_i}\right)\sin\left[\omega\left(t - \frac{|x|}{a_i}\right)\right] \qquad \textit{(4.3)}$$

Here H is defined as usual as the following way,

$$H = \frac{1}{\mu} curlA$$

We omit the modeling of H because of the similarity between the previous ones.

Now we will model the formulas separately. First we will explore the formula ϕ_i.

4.3.1 Modeling formula (4.2):

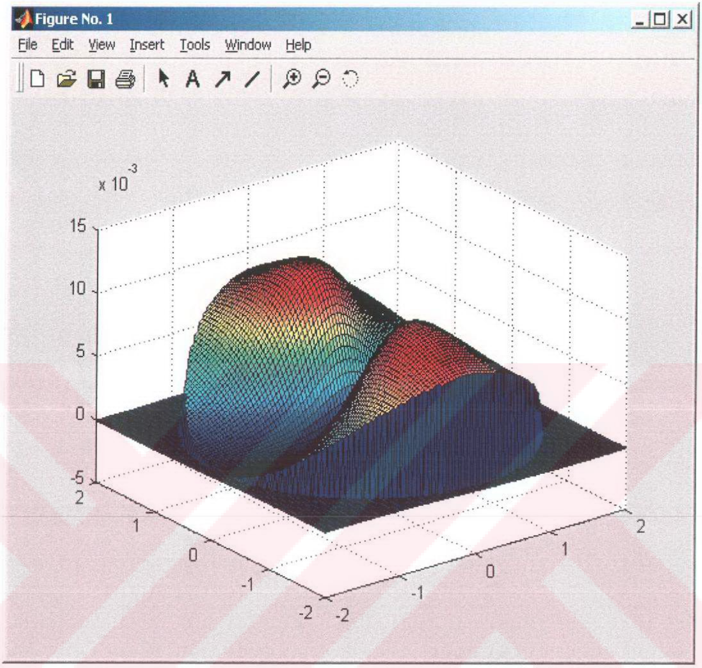

Figure 4.1: Modelling formula (4.2)

Time	1
Epsilon	$\frac{1}{2^{15}}$
X	[x, x, 0.7]
a	[2.0, 5.0, 8.0]
e	[1.0, 2.0, 4.0]
Frequency	100
w	1

Here because the formula contains Heaviside function multiplied with a combination of trigonomeric functions results the figure in 4.1.

4.3.2 Modeling formula (4.3):

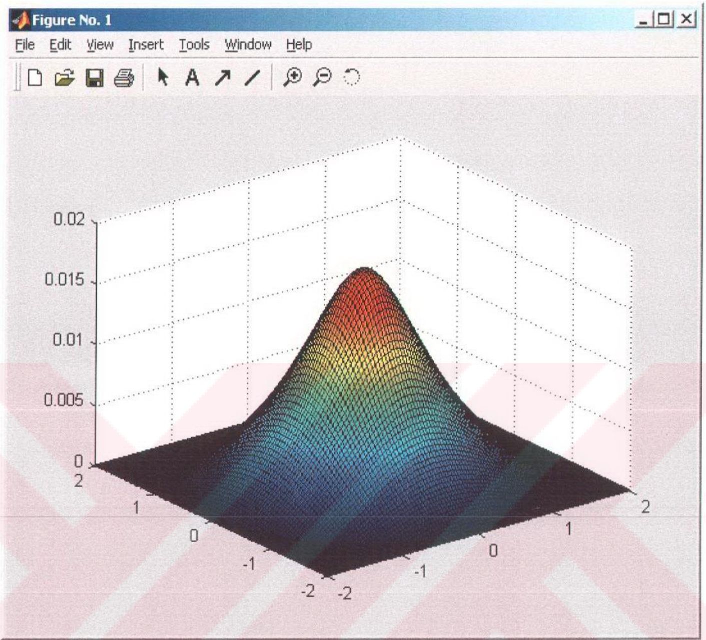

Figure 4.2: Modelling formula (4.3)

Time	1
Epsilon	$\frac{1}{2^{15}}$
X	[x, x, 0.7]
a	[2.0, 5.0, 8.0]
e	[1.0, 2.0, 4.0]
Frequency	100
w	1

In modeling of formula 4.3 because of Heaviside function there is a strict circle around the graph which is the cut off point of the Heaviside function. And also because multiplication with trigonometric function sinus, we also get a pick which's center is at the origin where the sin function gets its largest value.

4.3.3 Modeling formula (4.2) & (4.3) together:

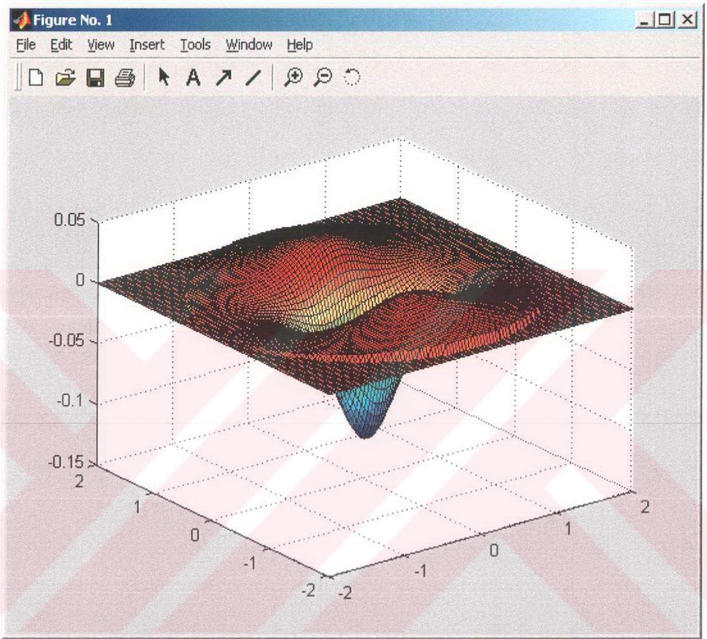

Figure 4.3: Modelling formula (4.2) & (4.3) together

Time	1
Epsilon	$\frac{1}{2^{15}}$
X	[x, x, 0.7]
a	[2.0, 5.0, 8.0]
e	[1.0, 2.0, 4.0]
Frequency	100
w	1

Combination of these two distinct parts 4.2 and 4.3, results in the figure 4.3.

4.4 Anisotropic Complex Case

Modeling Maxwell's equations under anisotropic medium with material constant ε, which is arbitrary is more complex and time consuming operation. To overcome this problem first we will observe initial value problem for the vector A and then diagonalization problem of arbitrary matrix ε.

4.4.1 Initial Value Problem for the Vector A

For finding A, we consider the following initial boundary problem

$$\frac{\partial^2 A}{\partial t^2} + Q\frac{\partial A}{\partial t} - K\Delta A = F, \qquad x \in G, \tag{4.1}$$

$$A\big|_{t=0} = 0, \qquad \frac{\partial A}{\partial t}\bigg|_{t=0} = 0, \tag{4.2}$$

where $Q = 4\pi\sigma\omega$, $K = \frac{c}{\mu}\omega$, $F = 4\pi\omega j$ and G is the given bounded domain in R^3 with smooth boundary ∂G.

Assume that ω is symmetric matrix, that is $\omega_{ij} = \omega_{ji}$. Since ω is symmetric, K is symmetric also.

4.4.2 Diagonalization of the Matrix K

To solve the problem (4.1), (4.2) we will use the following theorem of diagonalization of a symmetric transformation:

Theorem: Let **K** a symmetric transformation of the space L_3. Then the matrix K of **K** can always be reduced to diagonal form by transforming to a new orthonormal basis e_1, e_2, e_3.

Before proving this theorem, we need some information about basis transformation:

Let e_1, e_2, e_3 be an orthonormal basis in the space L_3 and let e_1, e_2, e_3 be another orthonormal basis in L_3. Clearly, the vectors of the "new" basis e_1, e_2, e_3 can be expressed as linear combinations of the vectors of the "old" basis e_1, e_2, e_3. Let $\gamma_{i'i}$ denote the coefficent of e_i in the expension of $e_{i'}$ with respect to the old basis vector. Then, the expansion of the new basis vectors with respect to the old basis vectors, take the form:

$$\begin{aligned} e_{1'} &= \gamma_{1'1}e_1 + \gamma_{1'2}e_2 + \gamma_{1'3}e_3 \\ e_{2'} &= \gamma_{2'1}e_1 + \gamma_{2'2}e_2 + \gamma_{2'3}e_3 \\ e_{3'} &= \gamma_{3'1}e_1 + \gamma_{3'2}e_2 + \gamma_{3'3}e_3 \end{aligned}$$

or more concisely, $e_{i'} = \gamma_{i'i}e_i$.

The numbers $\gamma_{i'i}$ can be written in the form of a matrix, called T:

$$T = \begin{pmatrix} \gamma_{1'1} & \gamma_{1'2} & \gamma_{1'3} \\ \gamma_{2'1} & \gamma_{2'2} & \gamma_{2'3} \\ \gamma_{3'1} & \gamma_{3'2} & \gamma_{3'3} \end{pmatrix} \text{ or more concisely, } T = (\gamma_{i'i}).$$

$$T^{-1} = \begin{pmatrix} \gamma_{11'} & \gamma_{12'} & \gamma_{13'} \\ \gamma_{21'} & \gamma_{22'} & \gamma_{23'} \\ \gamma_{31'} & \gamma_{32'} & \gamma_{33'} \end{pmatrix} \text{ or more concisely, } T^{-1} = (\gamma_{ii'})$$

Proof: A symmetric linear transformation K of the space L_3 has three (pairwise) orthogonal eigenvectors. Using this, we assume that a_1, a_2, a_3 are eigenvectors of K, such that $(K - \lambda_i I)\vec{a}_i = 0$. Suppose we normalize these vectors, by setting

$$\frac{a_i}{|a_i|} = e_{i'}.$$

Then vectors e_1, e_2, e_3 make up an orthonormal basis, and are also eigenvectors of K.

Since $Ke_{1'} = \lambda_1 e_{1'}$, $Ke_{2'} = \lambda_2 e_{2'}$, $Ke_{3'} = \lambda_3 e_{3'}$, the transformation K has the matrix form:

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}$$

in this basis. Since the original basis e_1, e_2, e_3 and new basis $e_{1'}, e_{2'}, e_{3'}$ are both orthonormal, the transformation

$$e_{i'} = \gamma_{i'i} e_i$$

from the former to the latter is described by an orthogonal matrix

$$T = (\gamma_{i'i})$$

we then have

$$\Lambda = T^{-1} A T$$

where K is the matrix of **K** in the old basis and Λ is its matrix in the new basis.

4.4.3 Reduction the Problem (4.4.1)-(4.4.2)

Using this theorem, we say that there exists T such that

$$T^{-1} K T = \Lambda,$$

where Λ is the diagonal matrix.

The solution of the problem (4.4.1), (4.4.2) we seek in the form

$$A = T^{-1} Y,$$

where Y is the unknown function and $T^{-1} = (\gamma_{ii'})$ is the matrix which we found before.

To find the formula for $TK\Delta_x A$:

Substituting $A = T^{-1}Y$ into $TK\Delta_x A$, we get

$$TK\Delta_x A = TK\Delta_x \left(T^{-1} Y\right) = TKT^{-1}\left(\Delta_x Y\right) = \Lambda \Delta_x Y$$

Hence we obtain

$$TK\Delta_x A = \Lambda \Delta_x Y .$$

Multiplying by T equation (4.4.1) and substituting $A = T^{-1}Y$, we get

$$T\frac{\partial^2}{\partial t^2}\left(T^{-1}Y\right)+TQ\frac{\partial}{\partial t}\left(T^{-1}Y\right)-\Lambda\Delta_x Y=TF$$

$$TT^{-1}\frac{\partial^2 Y}{\partial t^2}+TQT^{-1}\frac{\partial Y}{\partial t}-\Lambda\Delta_x Y=TF$$

setting $TQT^{-1}=M=\left(m_{ij}\right)$

$TF=\vec{g}$

the equation (4.4.1), (4.4.2) will be written in the terms of Y_i as follows:

$$\frac{\partial^2 Y_i}{\partial t^2}+\sum_{j=1}^{3} m_{ij}\frac{\partial Y_j}{\partial t}-\lambda_i \Delta Y_i=g_i,\ x\in G, \qquad \textit{(4.3)}$$

$$Y_i\big|_{t=0}=0,\qquad \left.\frac{\partial Y_i}{\partial t}\right|_{t=0}=0,\ x\in G, \qquad \textit{(4.4)}$$

for $i=1,2,3$.

4.4.4 Sample Modeling for Anisotropic Complex Case

After now most of the operational steps are similar to the anisotropic simple case. As a sample modeling we assume the current density function j as $j=\varepsilon e\delta(x)\theta(t)$ and to find E we know that the formula $E(x,t)=-\frac{\partial A_i}{\partial t}+\phi_i$, will be used. To find E we will do the same things as we did in anisotropic simple case. After all we will have the resulting explicit form of E as below.

First form, where $i=j$

$$\mathbf{E}_i(\mathbf{x},\mathbf{t}) = \frac{\theta(t)}{4\pi} a_i^{2} \left\{ \sum_{j=1}^{3} \frac{x_i x_j e_j}{|x|^3 a_j} \delta\left(t - \frac{|x|}{a_j}\right) - \frac{e_i}{a_j^{4}} \frac{1}{|x|} \delta\left(t - \frac{|x|}{a_i}\right) - \sum_{j=1}^{3} \frac{e_j}{a_j^{3}} \left[\frac{x_i}{|x|} \frac{x_i}{|x|} \frac{x_j}{|x|^3} + \left(\frac{|x|^2 - 2x_i^2}{|x|^4} \right) \right] \theta\left(t - \frac{|x|}{a_j}\right) \sum_{j=1}^{3} \frac{e_j}{a_j^{2}} \left(\frac{-|x|^2 + 3x_i x_j}{|x|^5} \right) \theta_1\left(t - \frac{|x|}{a_j}\right) \right\} \qquad \textit{(4.5a)}$$

Second form, where $i \neq j$

$$\mathrm{E}_i(\mathrm{x},\mathrm{t}) = \frac{\theta(t)}{4\pi} a_i^{2} \left\{ \sum_{j=1}^{3} \frac{x_i x_j e_j}{|x|^3 a_j} \delta\left(t - \frac{|x|}{a_j}\right) - \frac{e_i}{a_j^{4}} \frac{1}{|x|} \delta\left(t - \frac{|x|}{a_i}\right) - \sum_{j=1}^{3} \frac{e_j}{a_j^{3}} \left[\frac{x_i}{|x|} \frac{x_i}{|x|} \frac{x_j}{|x|^3} + \left(\frac{-2x_i x_j}{|x|^4} \right) \right] \theta\left(t - \frac{|x|}{a_j}\right) + \sum_{j=1}^{3} \frac{e_j}{a_j^{2}} \left(\frac{-3x_i x_j}{|x|^5} \right) \theta_1\left(t - \frac{|x|}{a_j}\right) \right\} \qquad \textit{(4.5b)}$$

with this formulas we havent finished the modeling prerequisites. We must find the matrix K that we mentioned in the heading 4.4.2.

$$K = \frac{c}{\mu} \omega$$

Here μ and c are electromagnetic constants and also ω is constant represented as matrix. To make our modeling we must find the value of K. ω can be any arbitrary matrix, first we must transform this matrix into symetric one. This will make our modeling process very simple.

To generate a symetric matrix from an arbitrary one, we use MatLAB to make the operations quicker. In MatLAB there are predefined functions which help us to find the Eigen value and Eigen matrix of any arbitrary square matrix. Assume the following arbitrary square matrix

$$K = \begin{bmatrix} 1.0 & 3.4 & 6.0 \\ 6.1 & 3.7 & 4.0 \\ 7.2 & 5.5 & 8.7 \end{bmatrix}$$

To find the symetric form of the matrix above, we trace the following operations in MatLAB.

Code Segment 4.1: Step of Operations from MatLAB Command Window

```
(1)>> K= [1 3.4 6; 6.1 3.7 4 ; 7.2 5.5 8.7]

K =
    1.0000    3.4000    6.0000
    6.1000    3.7000    4.0000
    7.2000    5.5000    8.7000

(2)>> [E, EV] = eig(K)

E =
   -0.4204   -0.7860    0.1197
   -0.4661    0.5821   -0.8670
   -0.7785    0.2082    0.4838

EV =
   15.8813         0         0
         0   -3.1069         0
         0         0    0.6256

(3)>> IE = inv (E)
```

```
IE =
  -0.5377  -0.4715  -0.7119
  -1.0477   0.1282   0.4890
  -0.4143  -0.8139   0.7111

(4)>> IE * K * E

ans =
  15.8813   0.0000        0
  -0.0000  -3.1069  -0.0000
  -0.0000   0.0000   0.6256
>>
```

In the above code segment, we numerate the user inputs of MatLAB. In the first input (1), we initialize the arbitrary matrix and give it the name K. In the second command (2) we find the Eigen values and Eigen vectors of K. To check the correctness of resulting matrix and vector we take the inverse of Eigen vector and do the following formulation

$$\Lambda = T^{-1}AT$$

As in the formula above, multiplication of Eigen vectors inverse, the matrix itself and its Eigen vector results into the Eigen values of the vector.

By this way we find the value of matrix K and from this point a_1, a_2, a_3 are eigenvectors of K. Notice that a_1, a_2, a_3 are constants that we used in vector-function E defined in (4.5a) and (4.5b).

Now we can model the formula in (4.5). We translate this formula in to MatLAB and generate a screenshot with specific parameters given belove.

4.4.4.1 Modeling formula (4.5) - 1

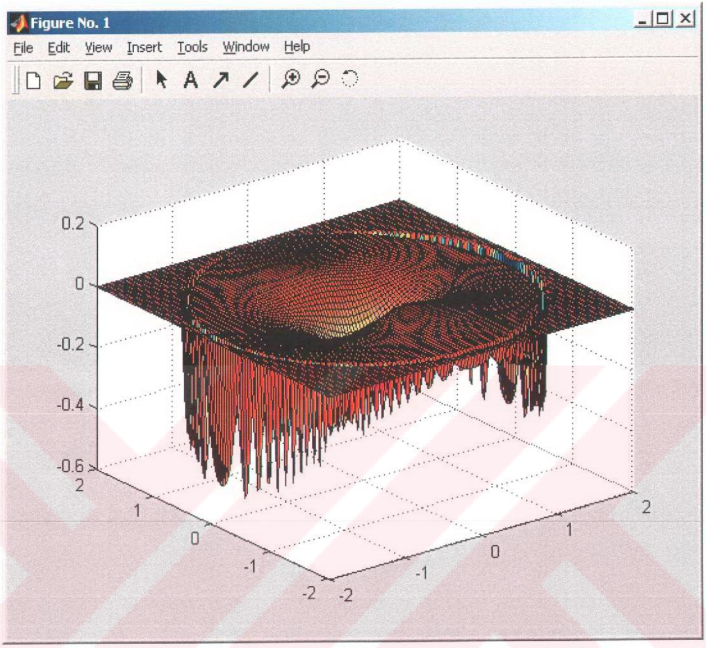

Figure 4.4: Modeling formula 4.5 with the first component of vector E

Time	1
Epsilon	$\frac{1}{2^{15}}$
X	[x, x, 0.7]
a	[2.0, 5.0, 8.0]
e	[1.0, 2.0, 4.0]
Frequency	100

4.4.4.2 Modeling formula (4.5) - 2

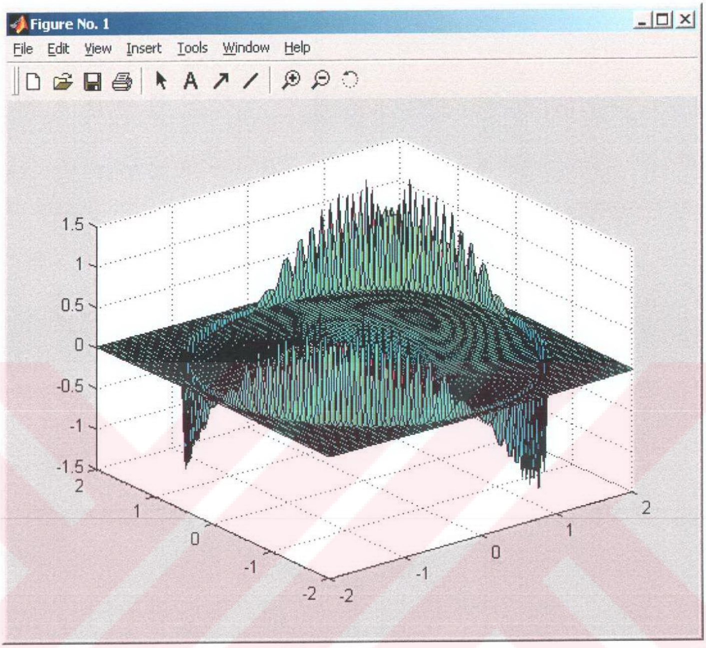

Figure 4.5: Modeling formula 4.5 with the second component of vector E

Same parameters used as in figure(4.4)

4.4.4.3 Modeling formula (4.5) - 3

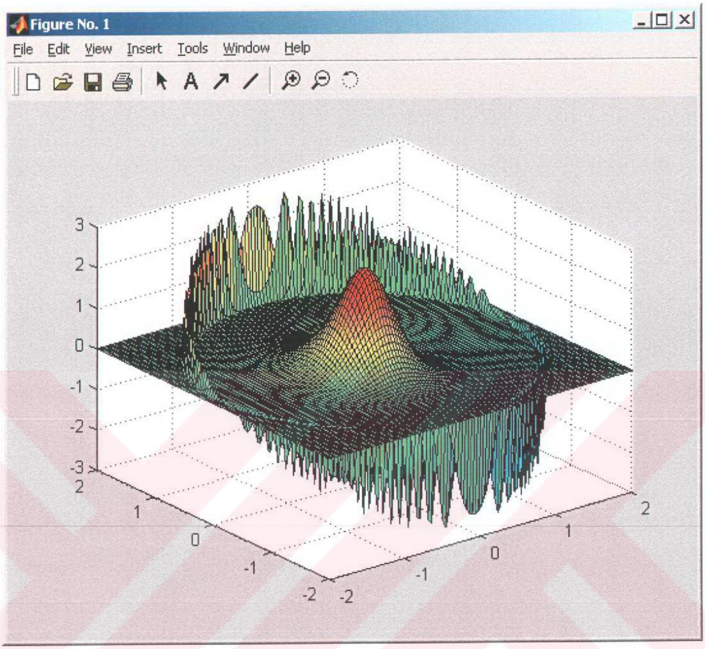

Figure 4.6: Modeling formula 4.5 with the third component of vector E

Same parameters used as in figure(4.4)

In figure 4.4, 4.5, 4.6 we represent the snapshot of the wave propagation under anisotropic complex case. These three snapshot represents three different components of the electromagnetic vector-function E.

CHAPTER FIVE

CONCLUSION & FUTURE WORKS

The main results of this thesis are the following

- Explicit formulas for the solutions of the Cauchy problem for the Maxwell system of anisotropic electrodynamics were obtained for the case when dielectric permeability is a symmetrical positive defined matrix, the magnetic permeability is a positive constant, the conductivity vanishes, density of the current is one of the following electric sources:
  - A point pulse source;
  - A point source with a constant amplitude with respect to time;
  - A point source with function amplitude depending on time.
- Special techniques for animation of wave propagations are developed.
- Graphics and animations were generated by MatLAB and C++ programming language. It gave opportunity to make very fast running applications.

In the future of this study, we will use matrices to define all the medium properties. This will require more complex explicit solutions. Again with these solutions we can generate animations in low frequency values.

Combining the vector components in one graphic representation is another aim. In the current study, each component of electromagnetic wave vector represented explicitly.

Also technological development in computer graphic hardware is very simultaneous. We can use the graphic cards special futures to decrease the animation building process time.

REFERENCES

1. Cohen, G.C., Heikkola, E., Joly, P., Neittaanmaki, P. (2003). Mathematical and Numerical Aspects of Wave Propagation. Springer.
2. Muller, C. (1969). Foundations of the Mathematical Theory of Electromagnetic Waves, Springer
3. Yakhno, V.G. (2002). Multidimensional Inverse Problems for Hyperbolic Equtions With Point Sources. Ill-Poused and Inverse Problems.VSP. Utrecht, The Netherlands. pp. 443-468.
4. Yakhno, V.G., Merazhov, I.Z. (2000). Advances in Mathematics. Direct Problems and One-Dimensional Inverse Problems of Electroelasticity for "Slow" Waves.Siberian. v.10, N.1. Allerton Press, Inc. New York. pp.87-150.
5. Yakhno, V.G. (1998). Tomography and Image Processing. Inverse Problem for Differential Equations System of Electromagnetoelasticity in Linear Approximation, Inverse Problems. Plenum, New York. pp.211-240.
6. Yakhno, V.G., Sevimlican, A. (2001). Selcuk Journal of Applied Mathematics. Fundamental Solution of the Cauchy Problem for an Anisotropic Electrodynamics. v.2, N.1. pp.83-94.
7. Tikhonov, A.N. ,and Samarskii, A.A. (1963). Equations of Mathematical Physics. Pergamon Press.
8. Alp R. Kut, Mustafa Kasap. (2002). 3D Graphics in OpenGL with Applications. Izmir, Turkey. Dokuz Eylül University press.
9. Microsoft Coorporation. (2003). Microsoft Visual Studio .NET 2003 Documentation.
10. The MathWorks Inc. (2001). MatLAB R12 Product Documentation.

11. Prosise, Jeff. (1999). Programming Windows with MFC. Washington Microsoft Press.

APPENDICES

Appendix A: Development under Microsoft Visual C++

In this chapter we will talk about preparing the OpenGL programming environment by means of MFC (The Microsoft Foundation Class Library). MFC encapsulates the Windows API and is a set of thin C++ wrapper around the API. The set of wizards that come along with Visual C++ ease the creation of a MFC Windows application development by providing the basic framework required. Thus we can focus on just adding OpenGL support to this instead of creating a Windows application from scratch.

With this brief introduction to the OpenGL extension for Windows it is time to write our first OpenGL program using MFC.

Do the followings step by step

- Run Visual C++ IDE (Integrated Development Environment).
- From the "File" menu create a new workspace by choosing "New" option.
- Choose "MFC AppWizard (exe)" as the project type.
- Enter project name as "Example1" and Choose an appropriate location
- In the wizard, choose the option "Single Document" and remove "Printing and Print Preview" and accept all other defaults.
- Compile and Execute the code.

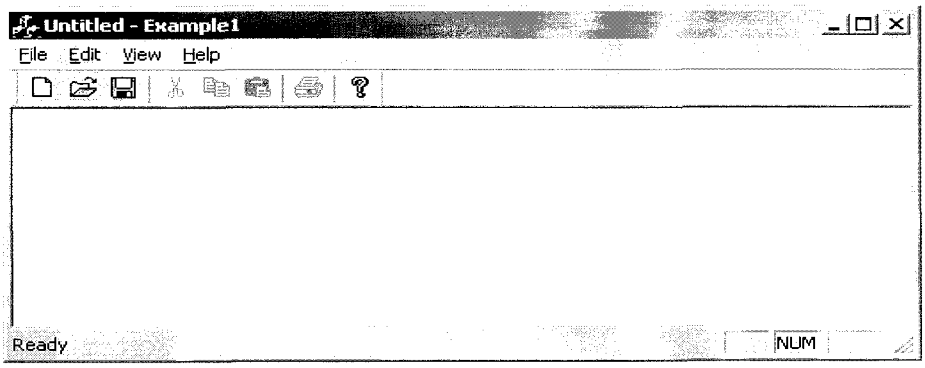

Figure A.1: Result window after operations above

We have the basic stuff to start programming OpenGL code. After this initial step we will use the same user interface to do coding. As we know, it is the View class that is responsible for drawing to the Window. So we will make most of the modifications over View Class.

- Open ClassWizard and select COpenGLView class. Add message handlers to the following messages - WM_CREATE (for OnCreate), WM_DESTROY (for OnDestroy), WM_SIZE (for OnSize), WM_ERASEBACKGROUND (for OnEraseBkground).
- Add the include statements for the OpenGL header files to the precompiled header file stdafx.h so that they would not be processed every time we compile.

Stdafx.h is project specific include files that are used frequently, but are changed Infrequently

```
//OpenGL Headers
#include <gl\gl.h>       //OpenGL Main Library Header
#include <gl\glu.h>      //OpenGL Utility Library Header
#include <gl\glaux.h>    //OpenGL Auxiliary Library Header
#include <gl\glut.h>     //OpenGL GLUT Library Header
```

After adding necessary library files you should compile it for testing your work.

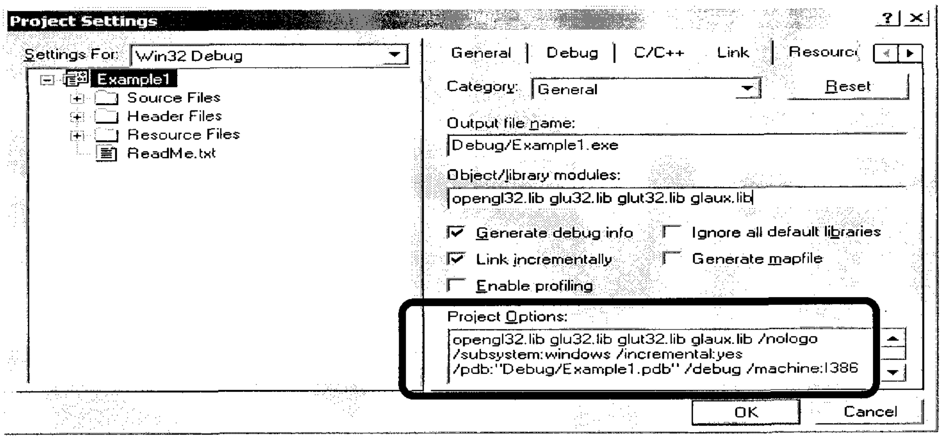

Figure A.2: Adding Library Files while linking

A.1 Editing PreCreateWindow()

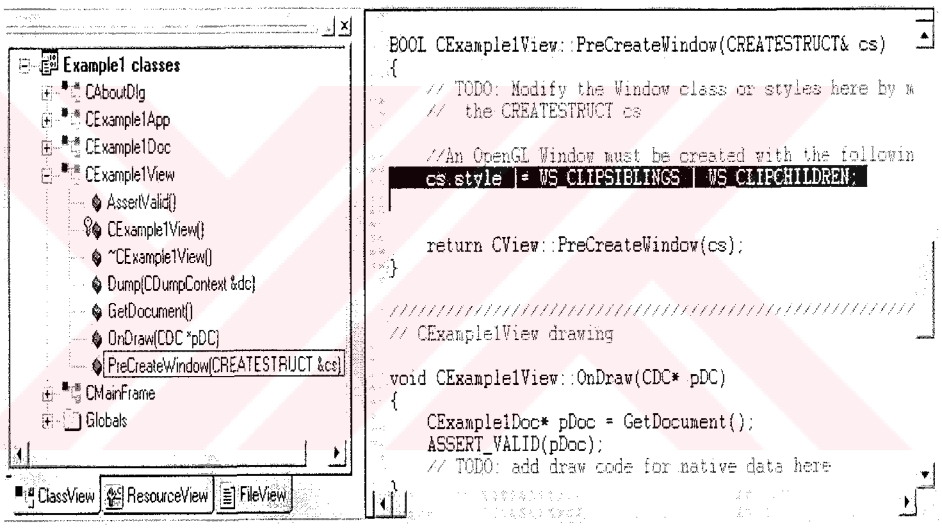

Figure A.3: Editing the PreCreateWindow thru ClassView Panel

Add the WS_CLIPCHILDREN and WS_CLIPSIBLINGS flags to the windows create structure. "WS_CLIPCHILDREN" excludes the area occupied by child windows when drawing occurs within the parent window. "WS_CLIPSIBLINGS" clips child windows relative to each other; that is, when a particular child window receives a WM_PAINT message, the WS_CLIPSIBLINGS style clips all other overlapping child windows out of the region of the child window to be updated.

A.2 Editing OnCreate() and Setting up a Pixel Format and Rendering Context

When we receive WM_CREATE message in our CExample1View class it is time to setup Pixel format and initialize the rendering context. Now Open the OpenGLView.h header file and add the following lines in the public section of the class:

HGLRC m_hRC; //Rendering Context
CDC m_pDC; //Device Context*

These variables will be used to set up the rendering context of the window and its device context.

To capture the WM_CREATE message we will add the OnCreate method to our class. To add OnCreate message handler do the following:

- Right click over the CExample1View on the ClassView panel.
- Select the Add "Windows Message Handler…" option.

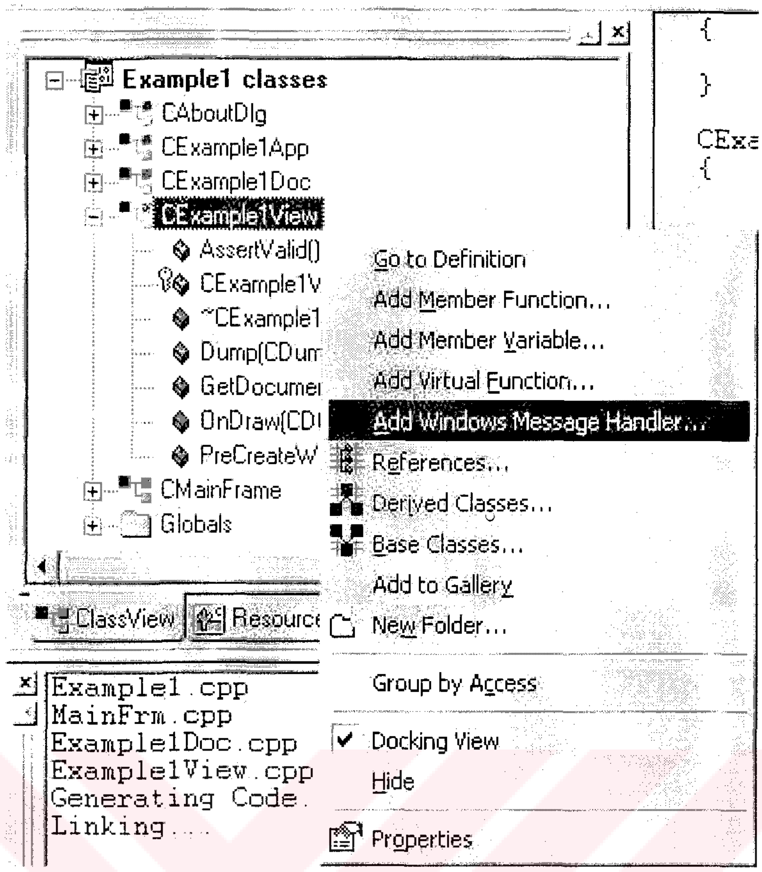

Figure A.4: Adding OnCreate Message Handler

In the following step you will see the menu window below. Select the WM_CREATE then click on the “Add and Edit” button.

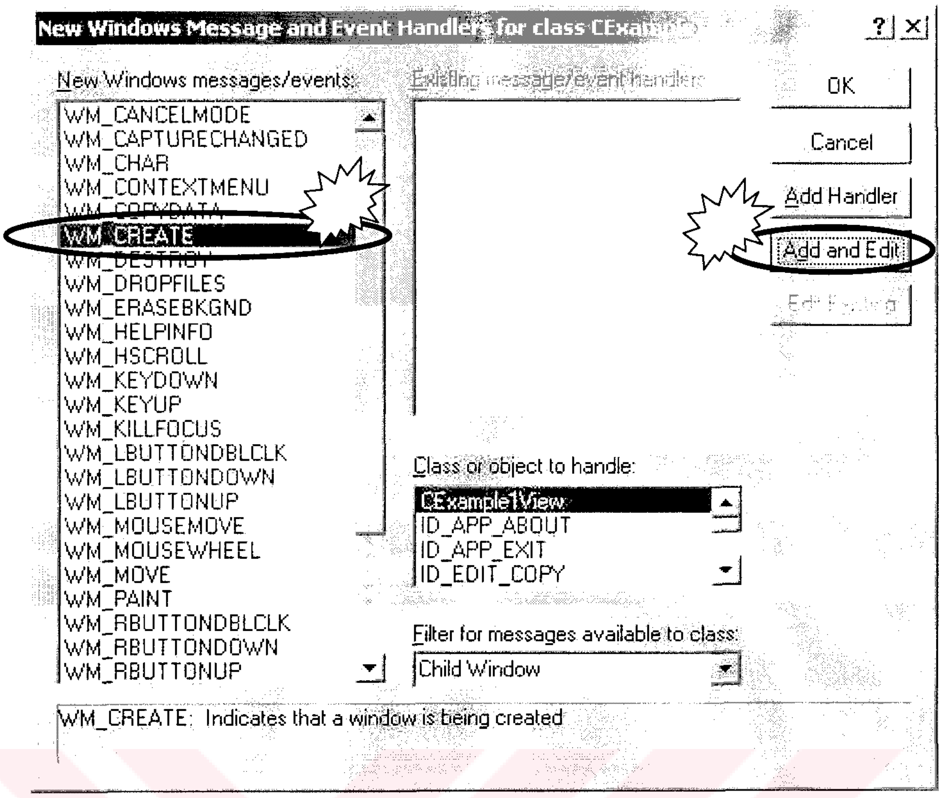

Figure A.5: Using WM_CREATE method

Skeleton of OnCreate method will be like this

```
int CExample1View::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CView::OnCreate(lpCreateStruct) == -1)
            return -1;
            // TODO: Add your specialized creation code here
            return 0;
}
```

Instead of writing all the code in OnCreate method we just call InitializeOpenGL function in OnCreate method. And also we call another function named SetupPixelFormat in InitializeOpenGL function.

Now again right click over CExample1View class as shown in Picture Mvc4 and click over the "Add Member Function…" you will see the menu below

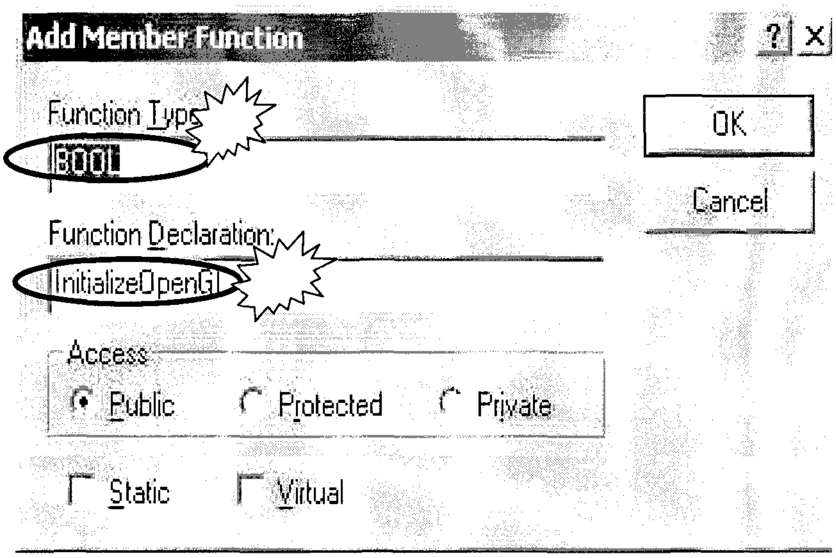

Figure A.6: Add Member Function

Type BOOL in the Function Type edit box. It means that our function will return Boolean value as a result. And type InitializeOpenGL in the Function Decleration edit box. It is the name of our function.

After clicking the OK button the skeleton of our InitializeOpenGL function is created as below

```
BOOL CExample1View::InitializeOpenGL()
{
}
```

Now type the code below in InitializeOpenGL function

```
    //Get a DC for the Client Area
    m_pDC = new CClientDC(this);

    //Failure to Get DC
    if(m_pDC == NULL)
    {
            MessageBox("Error Obtaining DC");
```

```
        return FALSE;
}
//Failure to set the pixel format
if(!SetupPixelFormat())
{
        return FALSE;
}
//Create Rendering Context
m_hRC = ::wglCreateContext (m_pDC->GetSafeHdc ());

//Failure to Create Rendering Context
if(m_hRC == 0)
{
        MessageBox("Error Creating RC");
        return FALSE;
}
//Make the RC Current
if(::wglMakeCurrent (m_pDC->GetSafeHdc (), m_hRC)==FALSE)
{
        MessageBox("Error making RC Current");
        return FALSE;
}
//Specify Black as the clear color
::glClearColor(0.0f,0.0f,0.0f,0.0f);
//Specify the back of the buffer as clear depth
::glClearDepth(1.0f);
//Enable Depth Testing
::glEnable(GL_DEPTH_TEST);
return TRUE;
```

Now place a call to InitializeOpenGL function in OnCreateMethod as shown below.

```
int CExample1View::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CView::OnCreate(lpCreateStruct) == -1)
                return -1;
        // TODO: Add your specialized creation code here
```

```
        //Initialize OpenGL Here
        InitializeOpenGL();
        return 0;
}
```

Up to now we prepare the OnCreate method and InitializeOpenGL function. But SetupPixelFormat function is not ready yet, and also there is a call to SetupPixelFormat function in InitializeOpenGL function. Now we must prepare the SetupPixelFormat function. To do so do the same operations described in Picture Mvc6 but in the Function Decleration edit box write SetupPixelFormat then click OK. Now skeleton of SetupPixelFormat function is ready as below.

```
BOOL CExample1View::SetupPixelFormat()
{
}
```

Now write the following code in SetupPixelFormat function.

```
static PIXELFORMATDESCRIPTOR pfd =
    {
    sizeof(PIXELFORMATDESCRIPTOR),  // size of this pfd
    1,                              // version number
    PFD_DRAW_TO_WINDOW |            // support window
    PFD_SUPPORT_OPENGL |            // support OpenGL
    PFD_DOUBLEBUFFER,               // double buffered
    PFD_TYPE_RGBA,                  // RGBA type
    24,                             // 24-bit color depth
    0, 0, 0, 0, 0, 0,               // color bits ignored
    0,                              // no alpha buffer
    0,                              // shift bit ignored
    0,                              // no accumulation buffer
    0, 0, 0, 0,                     // accum bits ignored
    16,                             // 16-bit z-buffer
    0,                              // no stencil buffer
    0,                              // no auxiliary buffer
    PFD_MAIN_PLANE,                 // main layer
    0,                              // reserved
```

```
  0, 0, 0                          // layer masks ignored
};

int m_nPixelFormat = ::ChoosePixelFormat(m_pDC->GetSafeHdc(), &pfd);

  if ( m_nPixelFormat == 0 )
  {
 return FALSE;
  }

if ( ::SetPixelFormat(m_pDC->GetSafeHdc(), m_nPixelFormat, &pfd) == FALSE)
{
 return FALSE;
  }

return TRUE;
```

Let us examine the code step by step. In InitializeOpenGL function first we get a Device Context for the client area of our drawing panel. For more information about Device Context refer to the GDI subject in this book. Then we setup the pixel format of our device context according to our needs. After having a device Context with the specified pixel format we create a rendering context over this context. For initial screen font color we use black. We specify the back of the buffer as clear depth and we enable the depth test for drawn objects on the rendering context.

In SetupPixelFormat function we specify the properties of the rendering context. To do this we use a PixelFormatDescripter structure. First we fill the structure with our desired values, and make a call to ChoosePixelFormat function. Operating System looks for the desired pixel format and if the device capability supports our needs nothing changed in the structure. If device capability doesn't support the desired values then structure is filled with the nearest best values. After choosing the format we set the pixel format of our rendering context by calling SetPixelFormat function.

A.3 Editing OnSize()

Viewport and Viewing Frustum of the scene depends on the size of window that we will use to draw on. So we must take care about the sizing operation over the window. In the initial size operation (when the window created) and every other resize operation occurred by user must be captured and Viewport and Viewing Frustum must be reinitialized.

The basic operations that occur are setting up the viewport, selecting the Projection matrix, initializing it and setting up the viewing frustum. The last operations are selecting the Modelview matrix, initializing it and setting up the viewing transformations.

As described in Picture Mvc5 add and edit the WM_SIZE message handler to our CExample1View class. Skeleton of the OnSize method to handle the WM_SIZE message will be like this.

```
void CExample1View::OnSize(UINT nType, int cx, int cy)
{
	CView::OnSize(nType, cx, cy);
	// TODO: Add your message handler code here
}
```

Now add the following code to OnSize method:

```
GLdouble aspect_ratio; // width/height ratio

if ( 0 >= cx || 0 >= cy )
{
	return;
}

// select the full client area
::glViewport(0, 0, cx, cy);
```

```
// compute the aspect ratio
// this will keep all dimension scales equal
aspect_ratio = (GLdouble)cx/(GLdouble)cy;

// select the projection matrix and clear it
::glMatrixMode(GL_PROJECTION);
::glLoadIdentity();

// select the viewing volume
::gluPerspective(45.0f, aspect_ratio, .01f, 200.0f);

// switch back to the modelview matrix and clear it
::glMatrixMode(GL_MODELVIEW);
::glLoadIdentity();
```

In the code we declare a variable to store the aspect ratio i.e the width to height ratio of the Window as the viewing frustum depends on this. We exit from the method if the either the width or the height of the window is 0. One reason is to handle the division by zero error when calculating the aspect ratio. The glViewport function sets the viewport. Then we compute the aspect ratio, select the projection matrix and clear it. Setting the Viewing Volume. This means that all further commands will affect the projection matrix. No we select the Modelview matrix and initialize it. This means that all further commands will affect the Modelview matrix.

A.4 Editing OnDraw()

Most important part of the program is where all rendering operations performed. Click the OnDraw method name on the Class View Panel. This code already added by the MFC wizard. This method is called by the framework to render an image of the document. The framework calls this method to perform screen display, printing, and print preview, and it passes a different device context in each case.

There is a clear sequence of events that occurs in OpenGL programs when it is time to rerender the scene. These are:

- Clear the buffers
- Render the scene
- Flush the rendering pipeline
- Swap the contents of the back buffer if double buffering is being used.

To do the steps described above we add the following code in to the OnDraw event :

```
// Clear out the color & depth buffers
::glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

RenderScene();

// Tell OpenGL to flush its pipeline
::glFinish();

// Now Swap the buffers
::SwapBuffers( m_pDC->GetSafeHdc() );
```

In the code we place a call to the RenderScene method. This method is the place where all rendering operations takes place i.e. drawing circle, lines, triangles etc.

Now it is time to add the RenderScene method. As described before in Picture Mvc6 we add a member function

```
void COpenGLView::RenderScene ()
{
}
```

Type of the function is **void** and name is **RenderScene.** Just now we don't write any code in it.

A.5 Editing OnEraseBkgnd() and OnDestroy()

If we have done everything in the right way our program can be compiled and executed. After executing the program try resizing the window, you will see a very irritating flicker. Close the application and you will see that there is a memory leak reported by VC++ debugger in the output window. So we have two problems left to solve. The flicker occurs because Windows paints the background first and then OpenGL next. Since we have OpenGL doing the job of clearing the background, we'll turn off Windows from clearing the background. This can be done by editing OnEraseBkgnd() member function appropriately.

We do this by returning true from the function.

```
BOOL COpenGLView::OnEraseBkgnd(CDC* pDC)
{
  // TODO: Add your message handler code here and/or call default
  //comment out the original call
  //return CView::OnEraseBkgnd(pDC);
  //Tell Windows not to erase the background
  return TRUE;
}
```

You can add the OnEraseBkgnd function as described in Picture Mvc5. Just select the WM_ERASEBKGND and click Add & Edit button.

The next problem of memory leak occurs because we have used the new operator to allocate memory for the CClientDC object in the SetupPixelFormat function. So we have to explicitly de-allocate this memory by using the delete operator.

In OnDestroy() first make RC non-current, then delete the RC and then delete the DC. To add OnDelete Event we follow the same steps described above as in the OnEraseBkgnd. We just select the WM_DESTROY message instead of WM_ERASEBKGND. Code will be as below:

```
void CExample1View::OnDestroy()
{
      CView::OnDestroy();

      // TODO: Add your message handler code here

      //Make the RC non-current
      if(::wglMakeCurrent (0,0) == FALSE)
      {
            MessageBox("Could not make RC non-current");
      }

      //Delete the rendering context
      if(::wglDeleteContext (m_hRC)==FALSE)
      {
            MessageBox("Could not delete RC");
      }

      //Delete the DC
      if(m_pDC)
      {
            delete m_pDC;
      }
      //Set it to NULL
      m_pDC = NULL;
}
```

Build and execute the application. It should work fine now without any memory leaks or flickering. You can find the full code of the program in EXAMPLE1.

Up to now we prepare the base code of the application. After now we can modify the RenderScene method to draw objects in the scene. To get more abstract visual feedbacks from our program we modify the RenderScene method as below

```
void CExample1View::RenderScene()
{
    //Replace the current matrix with Identity Matrix
    glLoadIdentity();
    glTranslatef(0.0f,0.0f,-5.0f);
    glRotatef(-30.0f,1.0f,1.0f,0.0f);

    glutWireTeapot(1.0f);
}
```