SIGNAL PROCESSING BASED METHOD FOR MODELING AND SOLVING INVERSE SCATTERING PROBLEMS

by

Richard Shane Ritter

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2011

Approved by:

____
Dr. Michael A. Fiddy

____
Dr. Lee Casperson

____
Dr. Mohamed-Ali Hasan

____
Dr. Zbigniew Ras

ABSTRACT

RICHARD SHANE RITTER. Signal processing based method for modeling and solving inverse scattering problems. (Under direction of DR. MICHAEL A. FIDDY)

A mature and difficult problem, still preoccupying many research communities in different application areas, is the recovery of a quantitative image of some unknown penetrable strongly scattering object. In most fields, such as ground penetrating radar, seismic and medical applications, the problem is compounded by the availability of only limited angle and noisy data. One of the more common approximate solution methods is based on diffraction tomography that relies on the first Born approximation method, which limits applications to weakly scattering situations. More sophisticated methods are typically iterative in nature, computationally intense and may not converge. We have studied an alternative nonlinear filtering approach and developed a new way to implement it, as well as evaluating different filter functions to find an optimal form. We have applied this approach to a number of classes of objects and developed a user-friendly scattered field simulator as a resource for this and related inverse scattering problems. We also re-investigated the widely accepted limitations of the first Born approximation and found that when close to a scattering resonance, the first Born approximation can yield a good estimate of the object's scattering cross section. Tied to all of these imaging applications is the issue of limited data: how many sources and how many receivers are required for a given quality and reliability of the resulting image. We took a fundamental look at this issue in terms of the number of degrees of freedom of the entire source-measurement domain and deduced clear guidelines on the minimum data sets necessary that should be measured, in order to expect a reasonable image.

DEDICATION

The most important thing to me in my life and what basically inspires me in virtually everything that I do is my faith in God and His son Jesus Christ. This work is therefore dedicated to Him for the life he has given me and the ability to learn about, explore, examine, and enjoy His creation.

In conjunction with this, my life is made complete by my family. I gladly dedicate this effort to my loving wife Julie who has supported, encouraged, and loved me unconditionally throughout everything, and has been my biggest fan. Also, to my wonderful children, Kayla, Ruth Anne, Sarah, Susanna, Daniel, Abbie, Aaron, and Micaiah who make life interesting and wonderful and who have been patient and understanding with me as I have worked on this research.

Finally, I want to thank and dedicate this work to my parents who also supported and encouraged me in all aspects of my life and education, especially my dad who introduced me to the field of electricity and demonstrated excellence in his chosen profession.

ACKNOWLEDGEMENTS

I would first like to acknowledge and thank my advisor and committee chair, Dr. Michael A. Fiddy. Rarely have I worked or interacted with someone who has such a complete mastery of his field. Working with Dr. Fiddy was both inspiring and at times a little intimidating, but always instructive and productive. I am extremely grateful for the opportunity to work with him and for his time and efforts. His comments, ideas, questions, and suggestions were always extremely well thought out and insightful and led to many (if not most) of the ideas implemented in this research.

I would also like to acknowledge and express my gratitude to the members of my committee Dr. Lee Casperson, Dr. Mohamed-Ali Hasan, and Dr. Zbyszek Ras. I greatly appreciate their time and efforts in serving on my committee. I also very much appreciate the benefit I received from their courses and the insights they provided in my research.

There are also some key professors in my past that greatly encouraged me to continue in my pursuit of my educational goals and always had words of encouragement. Those individuals are Dr. Falih Ahmad (formally with UNCC), Dr. Frank Ingels (MSU), and Dr. Dan Hogan (HCC).

Finally, I would like to acknowledge and thank Ms. Pat Winter in the Electrical Engineering Department who helped me to maneuver and wade through all of the paperwork, deadlines and other political processes involved in pursuing a PhD. Without her help I would have been lost in the process.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Θ, Ω	solid angle in three dimensions
$\Phi(\mathbf{r})$	complex phase function
Ψ	total field
Ψ_{inc}	incident field
Ψ_{s}	scattered field
$\delta(\mathbf{r})$	delta or impulse function
ε	permittivity
ε_r, er	relative permittivity
ε_0	permittivity of free space
θ, φ	angle
κ	weighting function or constant
λ	wavelength (=c/*f*)
μ_r	relative permeability
c	speed of light (=$3x10^8$ m/s)
f	frequency in hertz
$f(k\hat{\mathbf{r}}, k\hat{\mathbf{r}}_{inc})$	scattering amplitude in image space
$f^{BA}(k\hat{\mathbf{r}}, k\hat{\mathbf{r}}_{inc})$	first Born approximation in image space
$\hat{g}$	complex cepstrum of signal *f(x)*
k	wave number (=$2\pi/\lambda$)
n	index of refraction
$\mathbf{r}$	direction vector ($\mathbf{r}$ = (x,y))
$F(u)$	Fourier transform of *f(x)*

$G_o(\mathbf{r},\mathbf{r}')$	Greens function in free space
$H_o^{(1)}$	Hankel function of the first kind
$J_{n+1/2}$	Bessel function of the first kind
N	degrees of freedom
Nr	number of receivers
Ns	number of sources
P_n^1	Legendre polynomial of the first kind
Q	efficiency factor of scattering for a Mie scatterer
R	reference signal
V	volume
V(**r**)	Target function that describes the fluctuations of the permittivity relative to free space in terms of **r**
Y_n	Bessel function of the second kind

CHAPTER 1: INTRODUCTION

1.1 The Inverse Scattering Problem Overview

For some time now, there has been a great deal of effort invested to develop an algorithm or method for reconstructing a target's image (and its intrinsic properties such as a quantitative map of its permittivity or refractive index) from scattered field data. This is an enormous challenge since the general inverse problem in itself is ill-posed, and can be mathematically difficult and computationally challenging. A general model or approach for this type of problem, that is quite widely used, is called the diffraction tomography method [6]. This method is historically, generally restricted to so-called weakly scattering targets. In this method a target is illuminated by a known quasi-monochromatic incident wave or waves, and the scattered field(s) are then measured around the target by some number of receivers. The challenge is then to take the measured field data from all of the receivers and use this in conjunction with knowledge of the incident field to reconstruct the original target. The need for this algorithm has many modern day applications including target identification, biomedical imaging, remote sensing, geophysical imaging, structure synthesis and non-destructive testing to name a few. An illustration of a typical general experimental setup for this method is shown in Figure 1.1

The diffraction tomography method utilizes the knowledge that if one deals with weak scatterers, the measured fields are altered by a target having some complex permittivity distribution in such a way that the data can be processed using an inverse Fourier transform and other processing methods to reconstruct the original target.

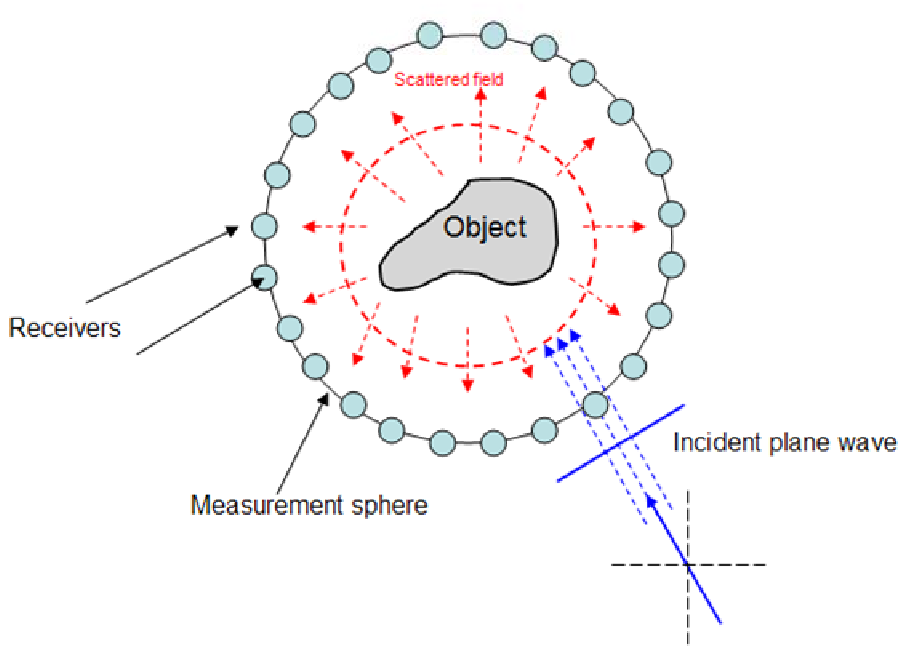


Figure.1.1. Typical experimental setup for diffraction tomography. The target or object is illuminated with a monochromatic plane wave and the scattered waves, after the interaction of the incident plane wave with the scattering object, are measured by receivers placed either in the near or far field, all around the object.

There has been limited success in implementing and utilizing this approach because it is limited to weak scatterers, and most targets of practical interest would more likely be deemed as strong scatters. This being the case, it would be desirable to either extend the current methods or perhaps develop new methods to achieve the same thing with non-weak scatterers, which are defined and discussed in Section 2.1.

1.2 Modeling and Data Generation

One of the challenges to developing and testing various inverse imaging algorithms is the fact that realistic, accurate data sets from strongly scattering targets can be difficult to acquire, making evaluation of imaging algorithms suspect. One reason for this is that the process for setting up and executing the lab experiments necessary to measure data can be difficult, time-consuming, and expensive. Even with the data that are available through various institutions such as US Air Force Research Laboratory's (AFRL) [1]-[4] and Institut Fresnel [5] among others, these data sets are for a very limited and discrete set of measurement conditions. This data is useful in the testing of algorithms, but in order to fully test and evaluate the accuracy of algorithms one needs the ability to vary or sweep various parameters over a wide range of values to observe the effects on the algorithm.

While this can be quite a challenge in the laboratory, it can be investigated much more easily and quickly in the virtual modeling realm by using (hopefully reliable) computer models to generate data sets to be used in conjunction with imaging algorithms. In addition to the advantages mentioned above, these models can also be used to control and vary numerous parameters in ways that would be difficult, if not impossible, to accomplish in the laboratory. This gives the ability to thoroughly examine the performance of an algorithm and its abilities, and limitations to the fullest extent in order to make adjustments for improved performance. As a part of this research, the ability to generate simulated data that can be used to test imaging algorithms is accomplished utilizing currently available modeling software and techniques.

1.3 Imaging Algorithm Evaluation and Improvements

With this newly developed data generation method, some of the existing imaging algorithms will be evaluated in great detail. Various parameters will be varied and the effects and limitations on the algorithms will be observed and documented. There has been an attempt recently to understand why one particular class of nonlinear (cepstral) algorithms, described later, developed for strong scatterers is not consistent in its reliability at recovering a quantitative estimate of the target's profile. This has led to questions about the underlying assumptions made in its formulation, as well as more deeply understanding issues arising from sampling rates and absolute values of scattered field data, when using a nonlinear technique. Efforts have been made to quantify the performance of these algorithms using data for various scenarios. Most targets can be simulated for virtually any range of incident frequencies, angles, permittivities, etc. which will allow for some very interesting and fundamental investigations.

Great effort and advancement has been made to the understanding and implementation of the cepstrum filtering algorithm [6] to attempt to improve the reliability and accuracy of this method of reconstruction. In particular, the general anomaly observed in which the general dimension of the reconstruction of the target incorrectly increases as the permittivity of the actual physical target increases will be addressed. The new data simulation/generation method developed for this research will be utilized to test multiple target scenarios for various modifications to identify the underlying causes and hopefully improvements. From these investigations progress will be made towards the optimal parameters and methodology that ensure the reliability of this reconstruction method.

1.4 Outline of Dissertation

This dissertation is organized as follows: Chapter 2 gives some of the basic background theory and mathematical basis for the imaging techniques examined in this research related to the scattering problem in general. Chapter 3 evolves from Chapter 2 in that it examines some of the common techniques or algorithms used to date to try and solve scattering problems. This chapter discusses the theory involved and gives the mathematical basis that will be built upon later in the research. Also in this chapter the concept of degrees of freedom is explained and discussed as to how it applies to the 2 dimensional models. Chapter 4 examines and explains the systematic method or approach used in this research to generate simulated scattered field data from known structures. The ability to generate this data this way proved essential for some of the analysis required for this research. Chapter 5 examines and explains some of the more recent efforts to extend the techniques introduced in Chapter 2 for weak scatterers to non-weak scatterers and the proposed attempts to improve these methods. In particular, independent source processing and filter optimization and modifications are discussed. Chapter 6 presents the results of the analysis done in this research. This includes the characterization of previous reconstruction methods and the attempts to improve on more recent methods and techniques. In this section, data sets are presented that will give a better understanding of the Born Approximation method and its capabilities. In addition to this, results from the improved Cepstrum method will be presented and compared to previous results for evaluation of performance. Finally in Chapter 7 the entire effort of this research is concluded and summarized. In this conclusion, results will be evaluated and issues that still remain will be documented and speculated on. Also thoughts on possible future endeavors to extend this work are offered and commented on. Finally, in the appendix, the entire MATLAB code used to perform these reconstruction methods, filtering, random analysis, and image reconstruction is included for examination, documentation, and use for further research and improvement of these methods.

CHAPTER 2: THEORETICAL BACKGROUND ON THE FORMULATION OF THE INVERSE SCATTERING PROBLEM

2.1 Categorization of Inverse Scatting Problems

In general, inverse scattering problems can be placed in one of two categories which are:

1) Weak Scattering
2) Non-Weak or Strong Scattering

Weak scattering occurs when the incident wave is only “scattered” once and this incident wave basically undergoes very little perturbation as it travels through or interacts with the target. This is significant in that in most of these cases the wave inside the target can generally be approximated as the incident wave which allows the problem to be linearized in order to find a solution. This approximation is what is exploited in the approach to the solution used in the Born and Rytov approximation methods [7] [10]. In these approaches, by linearizing the problem, one is able to establish a Fourier relationship between the measured scattered field data and the target or scattering function. In principle, these methods are only supposed to work well with weak scatterers due to the dependence on this approximation. More precisely, the Rytov approximation requires targets whose permittivity or index varies only very slowly on the scale of the wavelength. This concept of weak scattering will be examined and in some sense challenged in this research to see to what extent these methods actually work. When dealing with strong scatters, the field inside the target is scattered multiple times and can incur significant perturbation which introduces nonlinearities to the integral equation of scattering [9] which negates the linear approach to this problem, making it quite difficult to solve if and when the above mentioned methods do not perform well. A solution for the problem involving strong scattering is highly desired since most targets of interest in real life would fall into this category.

2.2 Mathematical Treatment of Inverse Scattering Problems

A common setup for a 2-dimensional inverse scattering experiment is shown in Figure 2.1. Typically there is some fixed number of receivers or receiver locations set up in some configuration centered around the center of the target, usually equally spaced. An incident plane wave then illuminates the target at some known angle ϕ_{inc} with respect to the x-axis and the scattered field is measured at an angle of ϕ_s in the far-field. Now a target or scattering object, represented by V(**r**), is placed in a homogeneous background, which has a permittivity of ε_0, where ε_0 is usually the permittivity of free space. This target or scattering object has a permittivity of $\varepsilon(\mathbf{r})$ which is related to the target by the equation

$$V(\mathbf{r}) = \varepsilon_{r0}(\mathbf{r}) - 1 \tag{2.1}$$

V(**r**) is an expression that describes the fluctuations of the permittivity relative to free space in terms of the coordinate system described in Fig. 2.1 where **r** = (x,y). Examining this relationship outside of the target boundary yields the following

$$V(\mathbf{r}) = \varepsilon_r(\mathbf{r}) - 1 = \varepsilon_0 - 1 = 1 - 1 = 0 \tag{2.2}$$

This demonstrates that V(**r**) is zero at all points outside of the target boundary which has a compact support domain of D which implies that the Fourier transform of V(r) is an entire function which is completely determined by its exact values on some region in

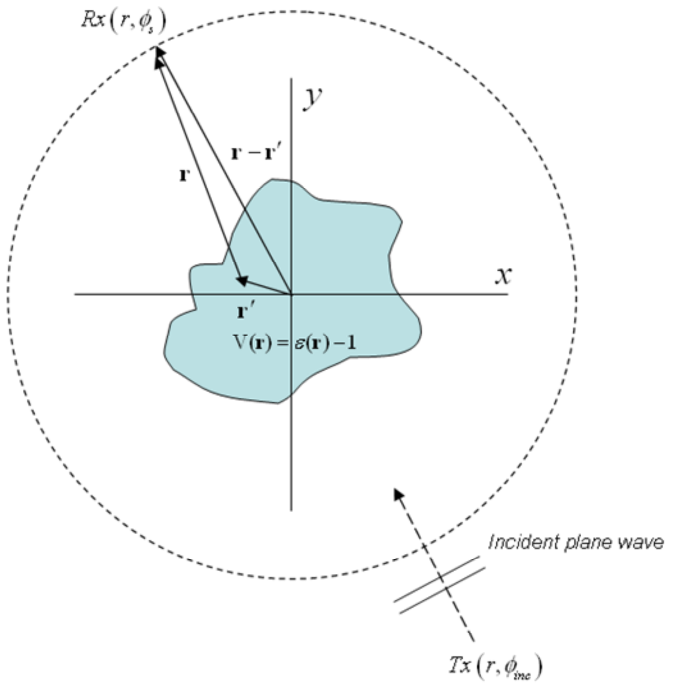


Figure. 2.1. A typical 2D inverse scattering experimental setup. Transmitter Tx transmits incident quasi-monochromatic plane wave to the scattering object V(r). The receivers Rx are located all around the target which collects scattered field data after the interaction of incident wave with scattering object.

Fourier or "k" space from which it could in principle be determined everywhere using analytic continuation [6]. The incident plane wave (in the absence of the target) is governed by the scalar homogeneous Helmholtz equation [7] as given here:

$$\left(\nabla^2 + k^2\right)\Psi_{inc}(\mathbf{r}) = 0 \tag{2.3}$$

where k is the wave number as defined by $k = 2\pi/\lambda$. The solution for the incident field, Ψ_{inc}, can be written in terms of the standard exponential form of a plane wave

$$\Psi_{inc}(\mathbf{r}) = e^{ik\hat{\mathbf{r}}_{inc}\cdot\mathbf{r}} \tag{2.4}$$

where $\hat{\mathbf{r}}_{inc}$ is the unit vector that specifies the direction of the incident field. An expression for the total field $\Psi(\mathbf{r})$, in terms of the coordinate system $\mathbf{r}$, can be obtained by looking at the interaction of the incident field, $\Psi_{\text{inc}}(\mathbf{r})$, and the target $\mathrm{V}(\mathbf{r})$. This relationship can be express as an inhomogeneous Helmholtz equation [7] as follows:

$$\left(\nabla^2 + k^2\right)\Psi(\mathbf{r}) = -k^2\mathrm{V}(\mathbf{r})\Psi(\mathbf{r}) \tag{2.5}$$

The total field in Eq. 2.5 can now be expressed generally as the sum of the incident field and the scattered field as follows:

$$\Psi(\mathbf{r}) = \Psi_{inc}(\mathbf{r}) + \Psi_s(\mathbf{r}) \tag{2.6}$$

where Ψ_s is the scattered field. All fields in Eq. 2.6 are expressed in terms of the position as defined by the point $\mathbf{r}$. Now the total field, $\Psi(\mathbf{r})$ can be expressed in terms of an inhomogeneous Fredholm integral equation of the first kind [11] as follows:

$$\Psi(\mathbf{r},\hat{\mathbf{r}}_{inc}) = \Psi_{inc}(\mathbf{r}) - k^2\int_D \mathrm{V}(\mathbf{r}')\Psi(\mathbf{r}',\hat{\mathbf{r}}_{inc})G_o(\mathbf{r},\mathbf{r}')d\mathbf{r}' \tag{2.7}$$

where $G_o(\mathbf{r},\mathbf{r}')$ is the (free space) Green's function which is the solution of scalar Helmholtz equation (Eq. 2.4) and it satisfies differential equation [11]

$$\left(\nabla^2 + k^2\right)G_o(\mathbf{r},\mathbf{r}') = \delta(\mathbf{r}-\mathbf{r}') \tag{2.8}$$

The Greens function basically gives the field amplitude at any point **r**, generated by any given point source located at **r'**. Since the modeling space in this case is homogonous and rotationally symmetrical, the Greens function can be solved in spherical coordinates with **r'** at the origin as follows:

$$\left(\nabla^2 + k^2\right)G_o(\mathbf{r}) = -\delta(\mathbf{r}) = -\delta(x)\delta(y)\delta(z) \tag{2.9}$$

Which for homogenous, spherically symmetrical partial differential equations (PDE) the solution for Greens function in free space can be written as

$$G_o(\mathbf{r}) = \frac{Ce^{ikr}}{r} + \frac{Ue^{-ikr}}{r} \tag{2.10}$$

Utilizing the radiation boundary condition, which is simply that the sources cannot be present at infinity, this translates to that only the outgoing wave solution(s) exists and therefore *U* must equal to 0 (*U*=0). This simplifies Eq. 10 as follows:

$$G_o(\mathbf{r}) = \frac{Ce^{ikr}}{r} \tag{2.11}$$

This (Eq. 2.11) can now be substituted back into Eq. 2.9 and the entire equation can then be integrated over a small volume ΔV about the origin as shown here:

$$\int_{\Delta V} dV\left(\nabla^2 + k^2\right)\frac{Ce^{ikr}}{r} = \int_{\Delta V} dV(-\delta(x)\delta(y)\delta(z)) \tag{2.12}$$

where the right hand side can be simplified due to the identity of the delta function as follows:

$$\int_{\Delta V} dV\left(\nabla^2 + k^2\right)\frac{Ce^{ikr}}{r} = -1 \tag{2.13}$$

Now the left side of Eq. 2.13 can be simplified by applying the distributive property as follows:

$$\int_{\Delta V} \nabla^2 \frac{Ce^{ikr}}{r} dV + \int_{\Delta V} k^2 \frac{Ce^{ikr}}{r} dV = -1 \tag{2.14}$$

As V is taken to be smaller and smaller, the second term in Eq. 2.14 gradually goes to zero due to the fact that dV in this term is defined as

$$dV = r^2 \sin\theta dr d\theta d\phi \tag{2.15}$$

and as *V* gets smaller so does the r^2 term in Eq. 2.15 which causes the second term to go to zero as stated earlier. This further simplifies Eq. 2.14 as follows:

$$\int_{\Delta V} \nabla^2 \frac{Ce^{ikr}}{r} dV = -1 \tag{2.16}$$

Eq. 2.16 can then be rearranged as follows:

$$\int_{\Delta V} \nabla^2 \frac{Ce^{ikr}}{r} dV = \int_{\Delta V} dV \nabla \cdot (\nabla \frac{Ce^{ikr}}{r}) = -1 \tag{2.17}$$

This now allows the divergence theorem to be applied to Eq. 2.17 as follows:

$$\int_{\Delta V} dV \nabla \cdot (\nabla \frac{Ce^{ikr}}{r}) = \oiint_{S} \nabla \frac{Ce^{ikr}}{r} d\mathbf{S} = -1 \tag{2.18}$$

which can be expanded as

$$\oiint_{S} \left(\frac{\partial}{\partial r} \frac{Ce^{ikr}}{r} d\mathbf{S} \right) r^2 \sin\theta d\theta d\phi = -1 \tag{2.19}$$

so that now if we utilize the fact that as **r** approaches zero, 4π*C* = -1, which can be solved for *C* as follows:

$$C = \frac{1}{4\pi} \tag{2.20}$$

This expression for *C* can now be substituted back into Eq. 21 which yields

$$G_o(\mathbf{r}) = \frac{e^{ikr}}{4\pi r} \tag{2.21}$$

This can now be generalized by shifting the source back to **r’** which gives Green’s function in the general form for this application as

$$G_o(\mathbf{r}, \mathbf{r}') = \frac{e^{ik|\mathbf{r}-\mathbf{r}'|}}{4\pi |\mathbf{r}-\mathbf{r}'|} \tag{2.22}$$

This equation in Eq. 2.22 gives Green’s function in the form necessary to solve the inhomogeneous Helmholtz equations. In this research, all of the experiments are done in 2-dimensions, therefore the 2-dimensional form of the Green’s function can be expressed in terms of a zero order Hankel function of the first kind [9] [12] as follows:

$$G(\mathbf{r},\mathbf{r}') = \frac{-i}{4} H_o^{(1)}\left[k\left|\mathbf{r}-\mathbf{r}'\right|\right]$$

(2.23)

Now an asymptotic assumption can be applied in that as $\mathbf{r} \to \infty$**,** the zero order Hankel function of the first kind becomes

$$H_o^{(1)}\left[k\left|\mathbf{r}-\mathbf{r}'\right|\right] = \frac{4}{i}\frac{1}{\sqrt{8\pi kr}} e^{i(kr+\pi/4)} e^{-ik\hat{\mathbf{r}}\bullet|\mathbf{r}-\mathbf{r}'|} \tag{2.24}$$

where $\hat{\mathbf{r}} = \frac{\mathbf{r}}{|\mathbf{r}|}$**.** Substituting Eq. 2.24 back into Eq. 2.23 the approximated 2-dimensional Green's function now becomes

$$G(\mathbf{r},\mathbf{r}') \simeq -\frac{1}{\sqrt{8\pi kr}} e^{i(kr+\pi/4)} e^{-ik\hat{\mathbf{r}}\bullet|\mathbf{r}-\mathbf{r}'|} \tag{2.25}$$

Finally, a general equation for the scattered field can be obtained by taking Eq. 2.25 and substituting it back into Eq. 2.6 and Eq. 2.7 which gives the following:

$$\Psi_s(\hat{\mathbf{r}},\hat{\mathbf{r}}_{inc}) = \frac{1}{\sqrt{8\pi kr}} e^{i(kr+\pi/4)} k^2 \int_D \mathrm{V}(\mathbf{r}')\, e^{-ik\hat{\mathbf{r}}\bullet|\mathbf{r}-\mathbf{r}'|} \Psi(\mathbf{r}',\hat{\mathbf{r}}_{inc}) d\mathbf{r}' \tag{2.26}$$

Where $\hat{\mathbf{r}}_{inc} = (\cos\phi_{inc}, \sin\phi_{inc})$ is the unit vector that describes the direction of the incident plane wave and $\hat{\mathbf{r}} = (\cos\phi_s, \sin\phi_s)$ is the unit vector that describes the direction of the scattered wave. This equation is of the form of a Fredholm integral equation of the first kind. In order to solve this equation for V(**r**), we must know or be able to adequately approximate what Ψ(**r**) is inside the target domain D.

2.3 Lorenz-Mie Scattering Problems

One very specialized and important scenario of the inverse scattering problem that will be looked at in detail later in this research involves a target that is an isotropic, homogeneous, dielectric sphere in 3-dimensions or a disk in 2-dimensions. This type of problem is called a Lorenz-Mie scattering which was the first to published in 1908. The basic setup with variable definitions is shown in Figure 2.2. In general, Maxwell's equations are solved in spherical coordinates utilizing the separation of variables method. The problem is solved for the case when the field is determined at a distance that is much larger than the wavelength; otherwise known as the far-field condition or zone. In the far-field, the solution can be expressed in terms of two scattering functions as follows [27] [36]:

$$S_1(\Theta) = \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)} \left[a_n \pi_n(\cos\Theta) + b_n \tau_n(\cos\Theta) \right] \tag{2.27}$$

$$S_2(\Theta) = \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)} \left[b_n \pi_n(\cos\Theta) + a_n \tau_n(\cos\Theta) \right] \tag{2.28}$$

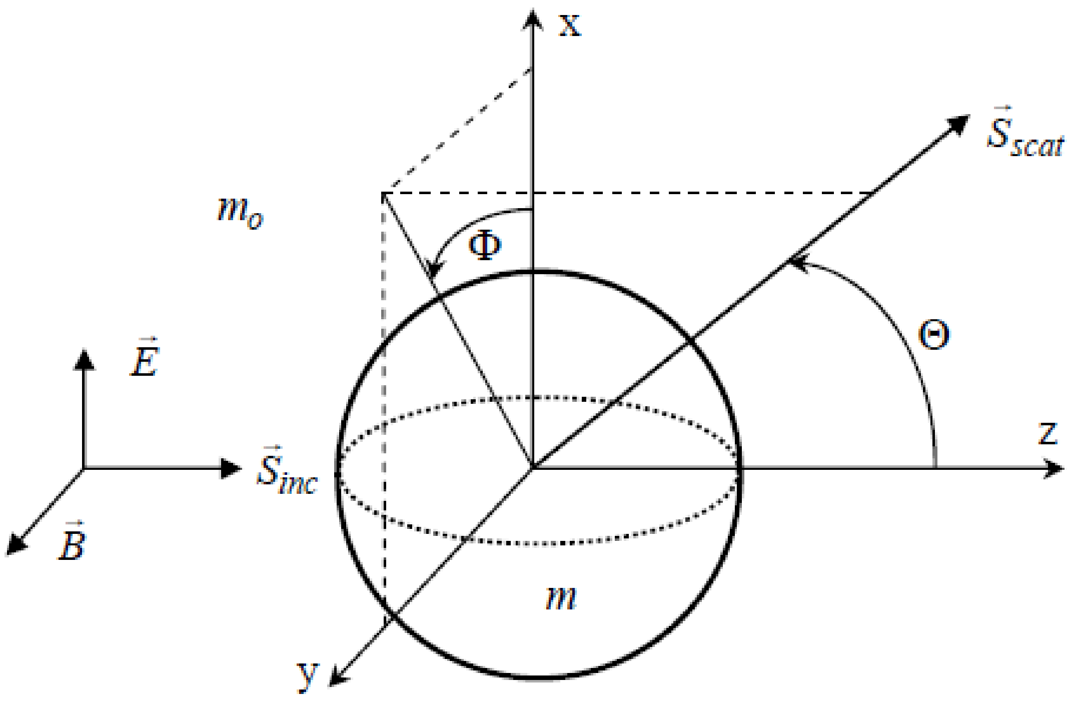


Figure 2.2. Typical setup and coordinate geometry with variable definitions for the Lorenz-Mie scattering problem.

where Θ is the scattering angle as defined in Figure 2.2. Furthermore,

$$\pi_n\left(\cos\Theta\right)=\frac{1}{\sin\Theta}P_n^1(\cos\Theta) \tag{2.29}$$

$$\tau_n\left(\cos\Theta\right)=\frac{d}{d\Theta}P_n^1\left(\cos\Theta\right) \tag{2.30}$$

where P_n^1 are represents the associated Legendre polynomials of the first kind. In addition

$$a_n=\frac{\psi_n'\left(m\alpha\right)\psi_n\left(\alpha\right)-m\psi_n\left(m\alpha\right)\psi_n'\left(\alpha\right)}{\psi_n'\left(m\alpha\right)\xi_n\left(\alpha\right)-m\psi_n\left(m\alpha\right)\xi_n'\left(\alpha\right)} \tag{2.31}$$

$$b_n=\frac{m\psi_n'\left(m\alpha\right)\psi_n\left(\alpha\right)-\psi_n\left(m\alpha\right)\psi_n'\left(\alpha\right)}{m\psi_n'\left(m\alpha\right)\xi_n\left(\alpha\right)-\psi_n\left(m\alpha\right)\xi_n'\left(\alpha\right)} \tag{2.32}$$

The size parameter**,** α, is defined as follows

$$\alpha=\frac{2\pi a m_0}{\lambda_0} \tag{2.33}$$

Finally, in Eqs. 2.31 and 2.32 the Ricatti-Bessel functions are defined as

$$\psi_n\left(z\right)=\left(\frac{\pi z}{2}\right)^{1/2}J_{n+1/2}\left(z\right) \tag{2.34}$$

$$\xi_n\left(z\right)=\left(\frac{\pi z}{2}\right)^{1/2}H_{n+1/2}\left(z\right)=\psi_n\left(z\right)+iX_n(z) \tag{2.35}$$

$$X_n\left(z\right)=-\left(\frac{\pi z}{2}\right)^{1/2}Y_{n+1/2}\left(z\right) \tag{2.36}$$

where Y_n is a typical Bessel function of the second kind.

In addition to these relationships, there are other characteristics of Mie scatterers that are of great interest in this research, in particular the efficiency factor of scattering which can be approximated by [35] [37]

$$Q = 2 - \frac{4}{p}\sin p + \frac{4}{p^2}\left(1 - \cos p\right) \tag{2.37}$$

where

$$p = 4\pi r\left(n - 1\right)/\lambda \tag{2.38}$$

The equations in Eq. 2.37 and Eq. 2.38 are very useful in that they are easily implemented and they approximate the scattering cross-section, predicted by the Lorenz-Mie theory, to within 1% [35] [38]. Later in this research, the "Q" factor will be utilized in explaining a cyclic phenomenon associated with the performance of reconstructed images obtained form the Born approximation method. It will be shown that resonance, or lack thereof, plays a vital role in the ability to successfully reconstruct an image using the techniques implemented in this research.

CHAPTER 3: COMMON METHODS FOR SOLVING INVERSE SCATTERING PROBLEMS

3.1 Data Inversion with Ewald Circles

Basically, there are two algorithms that are commonly used for performing Fourier-based methods for inversion of scattering data. They are:

1) Fourier based interpolation
2) Filtered back propagation.

Other methods, which are not addressed in this research, are more complex and computationally intensive methods such as modified gradient techniques that are iterative in nature and which do not necessarily converge for strong scatterers. These two methods are discussed at length in [7]. The basic methodology used in the Fourier interpolation method is that the scattered field data are placed on semicircular arcs in 2-dimensional k-space or the Fourier domain where the loci of points are defined by the Ewald circles or, in 3-dimensions, Ewald spheres [8]. The Ewald circles arise naturally from equation 2.26 when adopting the first Born approximation. They are tangent to the origin of 2-dimensional *k*-space with a radius of *k*, where *k* represents the magnitude of the scattered field's wavenumber. The transmitted, i.e. forward scattered data lying on the part of the circle closest to the origin and the reflected, i.e. backscattered data lying on the part of the circle farthest away from the origin as depicted in Figure 3.1. The position of the circle is relative to the direction of the source of the incident wave that is illuminating the target. As the incident wave is moved or rotated around the target, and the subsequent data gathered from the stationary receivers and mapped onto the respective Ewald circle, this forms additional Ewald circles of data in k-space.

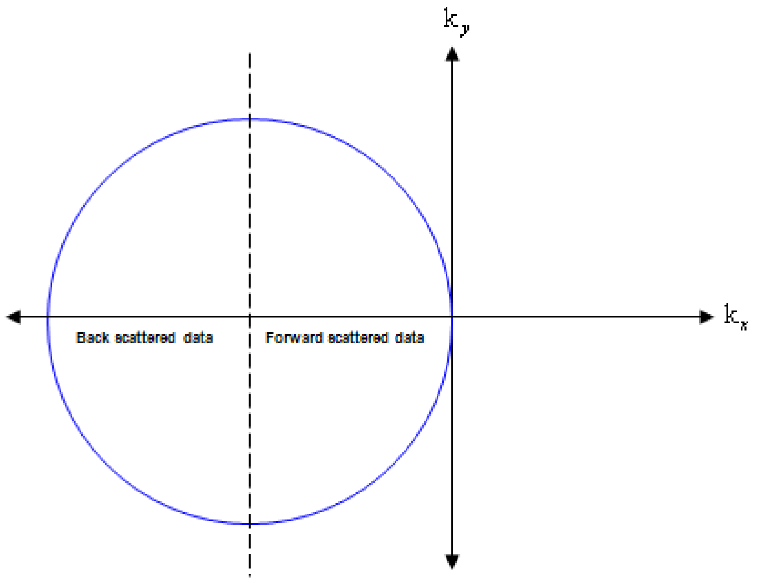


Figure. 3.1. k-space or spatial frequency space map showing that the forward scattered field data lies close to the origin and backscattered data lies furthest from the origin assuming the target is illuminated from the –x direction.

Ideally, as the source is rotated all the way around the target, k-space is filled to some degree with data out to a “limiting circle” of radius 2k as shown in Figure 3.2. This method can be used to develop an estimate of the Fourier transform of the target for the given incident frequency but, theoretically this is only representative of an image in the weakly scattering limit known as the first Born approximation. For more general scattering targets, these Fourier data must be interpreted in terms of the integral equation in Eq. 2.26 indicating dependence on both the scattering distribution as well as the total field that resides inside the scattering volume. Other incident frequencies can be used, which will in turn vary the radius of the corresponding Ewald circle to help to fill k-space. The radius of the Ewald circle is directly proportional to the frequency of the incident illuminating source as illustrated in Figure 3.3.

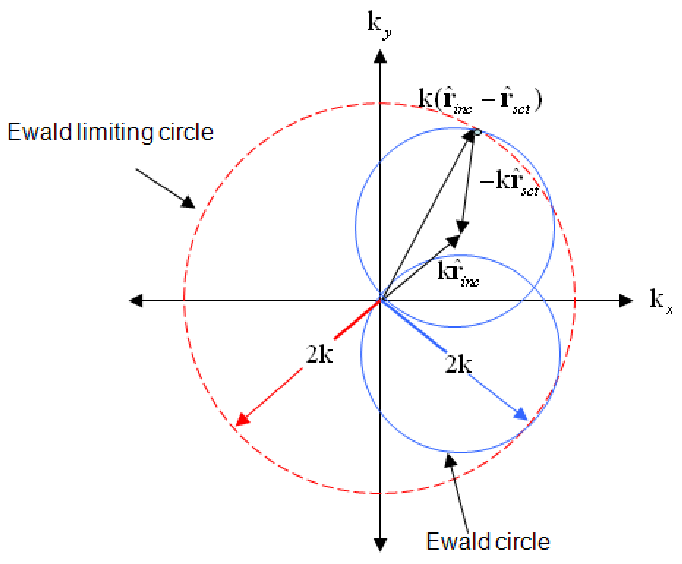

k_y
$k(\hat{\mathbf{r}}_{inc} - \hat{\mathbf{r}}_{sct})$
Ewald limiting circle
$-k\hat{\mathbf{r}}_{sct}$
$k\hat{\mathbf{r}}_{inc}$
k_x
2k
2k
Ewald circle

Figure. 3.2. Fourier space (k-space) of the object as a result of interaction of different incident plane waves with scattering object. The direction of the incident field $\hat{\mathbf{r}}_{inc}$ and the direction $\hat{\mathbf{r}}_{sct}$ of a particular plane wave component of the scattered field define a point at the Ewald circle. Changing incident field directions $\hat{\mathbf{r}}_{inc}$ fills the interior of Ewald limiting circle.

In the filtered back propagation method mentioned above, the scattered field data are "propagated" backwards into the object domain using an appropriate Green's function, which for this (and most) cases is assumed to be Green's function for free space [9] [10]. It is this assumption that also leads to the Fourier transform relationship exploited as seen later in the discussion on the Born approximation. However, as mentioned above, the back-propagation step leads to a field distribution in the target domain that only is proportional to the target itself in the weakly scattering limit in theory.

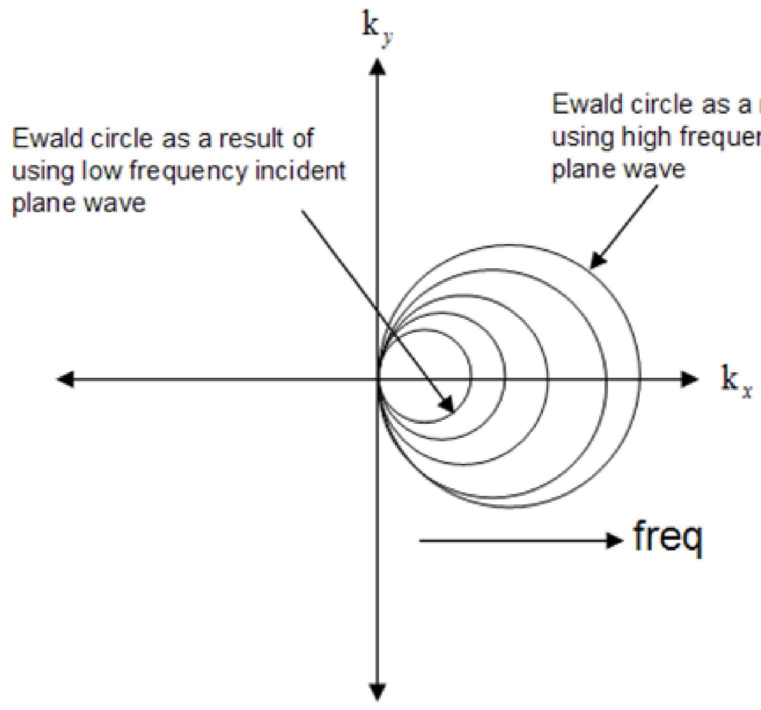

k_y
Ewald circle as a result of using low frequency incident plane wave
Ewald circle as a result of using high frequency incident plane wave
k_x
freq

Figure. 3.3 Radius of Ewald circle increases by increasing the frequency of the incident plane wave.

3.2 First Born Approximation

One of the most common approaches in imaging from scattered fields using the diffraction tomography method is the Born approximation. In general, this method assumes that the target is a weakly scattering object, and it is used in conjunction with the data inversion method described in the previous section. Many current inverse scattering algorithms utilize this approach [7] [10] even when it is not strictly valid to do so. As already indicated, when using this approach, the problem is linearized and establishes a Fourier relationship between the measured scattered field and some function of the target's scattering properties from which we hope to compute an image. A brief introduction of the first Born approximation is given here.

Recalling from Chapter 2 that the total measured scattered field at the receivers can be generally expressed as

$$\Psi(\mathbf{r}) = \Psi_{inc}(\mathbf{r}) + \Psi_s(\mathbf{r}) \tag{3.1}$$

this can be expanded in terms of an inhomogeneous Fredholm integral equation of the first kind [11] as follows

$$\Psi(\mathbf{r},\hat{\mathbf{r}}_{inc}) = e^{ik\hat{\mathbf{r}}_{inc}\bullet\mathbf{r}} + \frac{1}{\sqrt{8\pi kr}} e^{i(kr+\pi/4)} k^2 \int_D \mathrm{V}(\mathbf{r}')\, e^{-ik\hat{\mathbf{r}}\bullet\mathbf{r}'} \Psi(\mathbf{r}',\hat{\mathbf{r}}_{inc}) d\mathbf{r}' \tag{3.2}$$

The first term in the above equation is the contribution from the incident wave. This term can be pre-measured and subtracted out which leaves only the second term in above equation which is the scattered field expressed as follows

$$\Psi_s(\mathbf{r},\hat{\mathbf{r}}_{inc}) = \frac{1}{\sqrt{8\pi kr}} e^{i(kr+\pi/4)} k^2 \int_D \mathrm{V}(\mathbf{r}')\, e^{-ik\hat{\mathbf{r}}\bullet\mathbf{r}'} \Psi(\mathbf{r}',\hat{\mathbf{r}}_{inc}) d\mathbf{r}' \tag{3.3}$$

$$\Psi_s(\mathbf{r},\hat{\mathbf{r}}_{inc}) = \frac{1}{\sqrt{8\pi kr}} e^{i(kr+\pi/4)} f(k\hat{\mathbf{r}}, k\hat{\mathbf{r}}_{inc}) \tag{3.4}$$

where $f(k\hat{\mathbf{r}}, k\hat{\mathbf{r}}_{inc})$ is the scattering amplitude which is defined as

$$f(k\hat{\mathbf{r}}, k\hat{\mathbf{r}}_{inc}) = k^2 \int_D \mathrm{V}(\mathbf{r}')\, e^{-ik\hat{\mathbf{r}}\bullet\mathbf{r}'} \Psi(\mathbf{r}',\hat{\mathbf{r}}_{inc}) d\mathbf{r}' \tag{3.5}$$

The basis for the Born approximation is that the total field, Ψ, inside the target can be approximated by the incident field in the above integral in Eq. 3.5 as follows:

$$\Psi(\mathbf{r},\hat{\mathbf{r}}_{inc}) = e^{ik\hat{\mathbf{r}}_{inc}\bullet\mathbf{r}} \tag{3.6}$$

In theory, for the first Born approximation to be valid (i.e. for an object to be classified as a weak scatterer) a necessary condition is that the product of the target's permittivity, its characteristic dimension, and wave number should be much less than unity [13]. This is typically expressed mathematically as

$$k\mathrm{V}(\mathbf{r})\mathrm{a} << 1 \tag{3.7}$$

where k is the wave number, the object and permittivity are related by $\mathrm{V}(\mathbf{r}) = \varepsilon_r(\mathbf{r}) - 1$ where in this example $\varepsilon_r(r)$ is the relative permittivity of the target and 'a' specifies some measure of the physical size of the target. It is widely expected and believed by many that as the dimensions of the object increase, or the magnitude of permittivity fluctuations increase, the first Born approximation becomes increasingly poor. While this does seem to be the case in general, this assumption will be examined in more detail and the limits of this will be better defined as a result of this research.

In order to examine the Born approximation in more detail, Eq. 3.6 can be substituted into Eq. 3.5 which yields the linearized version of the inversion problem as follows:

$$f^{BA}(k\hat{\mathbf{r}}, k\hat{\mathbf{r}}_{inc}) = k^2 \int_D \mathrm{V}(\mathbf{r}')\, e^{-ik(\hat{\mathbf{r}}-\hat{\mathbf{r}}_{inc})\cdot\mathbf{r}'} d\mathbf{r}' \tag{3.8}$$

which illustrates where the Ewald circles originate and where the scattered field within the first Born approximation is expressed as

$$\Psi_s^{BA}(\mathbf{r}, \hat{\mathbf{r}}_{inc}) = \frac{1}{\sqrt{8\pi kr}} e^{i(kr+\pi/4)} k^2 \int_D \mathrm{V}(\mathbf{r}')\, e^{-ik(\hat{\mathbf{r}}-\hat{\mathbf{r}}_{inc})\cdot\mathbf{r}'} d\mathbf{r}' \tag{3.9}$$

This equation in Eq. 3.9 gives the Fourier relationship between the target or scattering object, V(r), and the measured scattering amplitude at the receivers $f^{BA}(k\hat{\mathbf{r}}, k\hat{\mathbf{r}}_{inc})$. If the integral in Eq. 3.9 is isolated as follows:

$$\int_D \mathrm{V}(\mathbf{r}')\, e^{-ik(\hat{\mathbf{r}}-\hat{\mathbf{r}}_{inc})\cdot\mathbf{r}'} d\mathbf{r}' = \frac{\left(\sqrt{8\pi kr}\right) e^{-i(kr+\pi/4)}}{k^2} \Psi_s^{BA}(\mathbf{r}, \hat{\mathbf{r}}_{inc}) \tag{3.10}$$

The right hand side of the Eq. 3.10 is what is implemented in MATLAB code in this research. This gives a representation not of V(**r**), but it gives an estimate for the product V(**r**)Ψ / Ψ_{inc}. The challenge is now to determine how to identify Ψ and effectively remove or minimize it to recover a valid representation of V(**r**).

One challenge with the Born approximation is an inconsistency mentioned by Ramm in [14]. In this inconsistency, Ramm demonstrated that the target or scattering object V(**r**) is real if and only if its Fourier transform is Hermitian. This means that in order for V(**r**) to be real it must be equal to the transpose of its complex conjugate. This is expressed as

$$-\frac{i}{2}\left\{ f^{BA}(k\hat{\mathbf{r}},k\hat{\mathbf{r}}_{inc}) - \left[f^{BA}(k\hat{\mathbf{r}},k\hat{\mathbf{r}}_{inc}) \right]^{*} \right\} = \mathrm{Im}[f^{BA}(k\hat{\mathbf{r}},k\hat{\mathbf{r}}_{inc})] \tag{3.11}$$

which can also be written as

$$\frac{k}{4\pi}\int_{\Omega\in H^2} d\Omega \left| f^{BA}(k\hat{\mathbf{r}},k\hat{\mathbf{r}}_{inc}) \right|^2 = 0 \tag{3.12}$$

where Ω is defined as the solid angle over which H^2 is integrated in either 2 or 3-dimensions. Eq. 3.12 clearly suggests that the target must be zero everywhere for the Born approximation to be valid. This simply is not possible for a real scattering object so therefore this indicates that there can be no exact solution for real objects for the Born approximation. Physically, V(r) is always complex, as dictated by dispersion relations that follow directly for causal systems. This notwithstanding, for sufficiently small scattering objects with negligible noise levels a consistent stable estimate of V(**r**)Ψ / Ψ_{inc} can be obtained by using a regularized inversion of the Fourier data [10], such as the Discrete (Inverse) Fourier Transform (DFT).

Taking another look at Eq. 3.8 it is evident that the inverse Fourier transform of the complex far-field scattering amplitude can provide a reasonable representation of the scattering object. As already discussed in the previous section this can be accomplished with one of two methods, namely the Fourier based interpolation and the filtered back propagation method [7]. When the target or scattering object is considered to be a non-weak or a strong scatterer, theoretically the Born approximation is said to be invalid. When this is the case, the Fourier inversion of many Ewald circles discussed earlier results in an approximation of V($\mathbf{r}$) which is expressed as

$$V_{BA}(\mathbf{r},\hat{\mathbf{r}}_{inc}) \approx \mathrm{V}(\mathbf{r})\left\langle \frac{\Psi(\mathbf{r},\hat{\mathbf{r}}_{inc})}{\Psi_{inc}(\mathbf{r},\hat{\mathbf{r}}_{inc})} \right\rangle \tag{3.13}$$

In this equation the $\langle\ \rangle$ symbols indicate an averaged dependence on the direction of the incident plane wave. The form of this equation suggests that the Ψ averaged term is independent from the V($\mathbf{r}$) term which if true implies that it should be separable. It is unclear at this time if this approximation, i.e., that these terms are independent of each other, holds for all classes of targets but we can argue that for increasingly complex targets, the averaged internal multiplied scattered fields will become increasingly noise-like, while V(r) remains unchanged. Eq. 3.13 is also characterized as an approximation since in principle the Fourier transformation can only be taken for $\hat{\mathbf{r}}_{inc}$ = constant and the overall accuracy will be affected by the limited data area coverage range, which is the case for the experiments in this research.

3.3 Rytov Approximation

While the Born Approximation is one of the more commonly used methods for linearizing an inverse scattering problem, there are other methods available. One of the more common alternative methods is the Rytov approximation. In this approach, the total field is represented in terms of a complex phase [7] [15] shown in Eq. 3.14

$$\Psi(\mathbf{r},\hat{\mathbf{r}}_{inc}) = \Psi_{inc}(\mathbf{r})e^{i\Phi_s(\mathbf{r})} \tag{3.14}$$

Where $\Phi(\mathbf{r})$ is the complex phase function and is defined as

$$\Phi(\mathbf{r}) = \Phi_{inc}(\mathbf{r}) + \Phi_s(\mathbf{r}) \tag{3.15}$$

In Eq. 3.15, the $\Phi_s(\mathbf{r})$ term is the phase function of the scattered field. If this new representation of the total field in Eq.3.14 is now substituted back into the general inhomogeneous Helmholtz Eq. 2.5 and utilizing defined identity functions yields the following:

$$\nabla^2\left[\Psi_{inc}(\mathbf{r})\Phi_s(\mathbf{r})\right] = \left[\nabla^2\Psi_{inc}(\mathbf{r})\right]\Phi_s(\mathbf{r}) + 2\nabla\Psi_{inc}(\mathbf{r})\bullet\nabla\Phi_s(\mathbf{r}) + \left[\nabla^2\Phi_s(\mathbf{r})\right]\Psi_{inc}(\mathbf{r}) \tag{3.16}$$

Simplifying, the inhomogeneous Helmholtz equation then becomes [10]

$$\left(\nabla^2 + k^2\right)\left[\Psi_{inc}(\mathbf{r})\Phi_s(\mathbf{r})\right] = i\left\{k^2 V(\mathbf{r}) - \left[\nabla\Phi_s(\mathbf{r})\right]^2\right\}\Psi_{inc}(\mathbf{r}) \tag{3.17}$$

As in Chapter 2, we utilize the free-space Green's function in Eq. 3.17, the complex phase function can then be written as

$$\Phi_s(\mathbf{r}) = \frac{ik^2}{\Psi_{inc}(\mathbf{r})}\int_D \mathrm{V}(\mathbf{r}')\Psi_{inc}(\mathbf{r}')G_o(\mathbf{r},\mathbf{r}')d\mathbf{r}' - \frac{i}{\Psi_{inc}(\mathbf{r})}\int_D \left[\nabla\Phi_s(\mathbf{r})\right]^2\Psi_{inc}(\mathbf{r}')G_o(\mathbf{r},\mathbf{r}')d\mathbf{r}' \tag{3.18}$$

Interestingly, in Eq. 3.18, the second term can be approximated as zero if the following criteria are met [10]

$$\left|k^2\mathbf{V}(\mathbf{r})\right| \gg \left|\left(\left[\nabla\Phi_s(\mathbf{r})\right]^2\right)\right| \tag{3.19}$$

$$\left|\varepsilon(\mathbf{r}) - 1\right| \gg \left|\left(\left[\frac{\lambda\nabla\Phi_s(\mathbf{r})}{2\pi}\right]^2\right)\right| \tag{3.20}$$

The implication or impact from these inequalities is that in order for these criteria to be met, the incident wavelength must be very small in comparison to the mean size of the target or scattering object, or that the spatial rate of phase change induced by the target or scattering object must be very small in terms of the unit wavelength. Eq. 3.20 implies that this approximation begins to break down as V(**r**) approaches zero. A global condition for the validity of the Rytov approximation is

$$\left| k^2 \int_D \mathrm{V}(\mathbf{r}')\Psi_{inc}(\mathbf{r}')G_o(\mathbf{r},\mathbf{r}')d\mathbf{r}' \right| \gg \left| \int_D \left[\nabla \Phi_s(\mathbf{r})\right]^2 \Psi_{inc}(\mathbf{r}')G_o(\mathbf{r},\mathbf{r}')d\mathbf{r}' \right| \tag{3.21}$$

So, when this criterion is satisfied, and the Rytov approximation is valid, the complex scattered phase can be expressed as

$$\Phi_s(\mathbf{r}) = \frac{ik^2}{\Psi_{inc}(\mathbf{r})} \int_D \mathrm{V}(\mathbf{r}')\Psi_{inc}(\mathbf{r}')G_o(\mathbf{r},\mathbf{r}')d\mathbf{r}' \tag{3.22}$$

This can now be substituted back into Eq. 3.14 to compute the total field as follows:

$$\Psi(\mathbf{r},\hat{\mathbf{r}}_{inc}) = \Psi_{inc}(\mathbf{r}) e^{i\left[\frac{ik^2}{\Psi_{inc}(\mathbf{r})}\int_D \mathrm{V}(\mathbf{r}')\Psi_{inc}(\mathbf{r}')G_o(\mathbf{r},\mathbf{r}')d\mathbf{r}'\right]} \tag{3.23}$$

If the argument of the exponent is now isolated by dividing by the incident field and the logarithm applied, the resulting form of this equation is

$$\Psi_{inc}(\mathbf{r},\hat{\mathbf{r}}_{inc}) \ln\left[\frac{\Psi(\mathbf{r},\hat{\mathbf{r}}_{inc})}{\Psi_{inc}(\mathbf{r},\hat{\mathbf{r}}_{inc})}\right] = -k^2 \int_D \mathrm{V}(\mathbf{r}') e^{ik\hat{\mathbf{r}}_{inc}\cdot\mathbf{r}} G_o(\mathbf{r},\mathbf{r}')d\mathbf{r}' \tag{3.24}$$

This equation is comparable to Eq. 3.9 from the Born approximation analysis in that it basically defines an inverse Fourier relationship or procedure to recover V(**r**). Eq. 3.24 can be very difficult to evaluate due to the nature and challenges of dealing with the multi-valued issues of the natural logarithm [16]. We will encounter this same difficulty in a later chapter.

Also, comparable with the Born approximation, when the conditions for the Rytov approximation are not valid, $\mathrm{V}_{RA}(\mathbf{r})$, is recovered in lieu of V(r) where

$$V_{RA}(\mathbf{r},\hat{\mathbf{r}}_{inc}) = \mathrm{V}(\mathbf{r}) - \frac{1}{k^2}\left[\nabla \Phi_s(\mathbf{r})\right]^2 \tag{3.25}$$

3.4 Discussion on Degrees of Freedom

In the above discussions, the scattering problem is viewed as an inverse Fourier problem, which is a valid approach, but possibly not a complete approach. There is another way of viewing this problem which may give more insight into what is actually going on and could possible give a better criterion for performance. If one were to view one isolated Ewald circle map of the data from most penetrable targets, one would notice that most of the information (non-zero information that is) is located near the origin or in the forward scattering section of the circle as shown in Figure 3.4. This being the case, since most of the information is in the forward scattering mode one could think of this as the information about the scatterer is "transmitted" through the target. This could mean that another valid approach to this problem would be to treat it as a transmission problem [23] where you have a source (the incident wave), a transmission medium (the target), and a receiver (the receivers located in the forward half of the Ewald circle). It is known from transmission theory [24] [25] or analysis that in order for certain types of signals to be successfully transmitted over a given medium that a minimum number of modes or bandwidth must be present to represent the necessary amount of information at the other end of the medium. This is sometimes referred to as the minimum degrees of freedom necessary for minimum image reconstruction. In this case, the available degrees of freedom are merely a function of the physical characteristics of the medium alone for a given wavelength [23] [24] [25]. Using this approach and applying it to this application as shown and described in [6] [23] [24] and [25] the general relationship that predicts the minimum degrees of freedom in 3-dimensions is

$$N_{3D} = B_v \cdot n_{\max} / \lambda^3 \tag{3.26}$$

where N_{3D} = is the minimum degrees of freedom required in 3 dimensions

B_v = target volume

$n_{\max}$ = maximum index of refraction

λ = wavelength

which can be modified easily for 2-dimensions as follows

$$N_{2D} = A_v \cdot n_{\max} / \lambda^2 \tag{3.27}$$

where the target volume is replaced with the target area A_v. While the criteria mentioned before in the Fourier approach do have merit, it is really more of a measure of the "weakness" of the scatterer, and also, to the validity of the assumption that the incident field is linear inside the target. The recent challenge is that this new criterion, defined above, is that it could be a better gauge of the performance or ability of an algorithm to reconstruct or "transmit" the original target. In this application, it is suggested that the degrees of freedom translate to the minimum number of independent target illumination directions or sources, or the minimum number of receivers or some combination thereof necessary to fully "transmit" all of the information related to the target. A series of experiments will be presented later to illustrate this in the following results section. In particular, each aspect of this criterion will be examined separately. More specifically, the degrees of freedom criteria will be applied to the number of sources first, then with some modifications this criteria will be applied to the number of receivers. In the conclusion some observations will be presented as to how these two aspects of imaging from scattered data may be related and some speculation also given as to a possible approach to deriving a mathematical representation that takes into account the degrees of freedom for the number of receivers and at the same time the number of receivers as well.

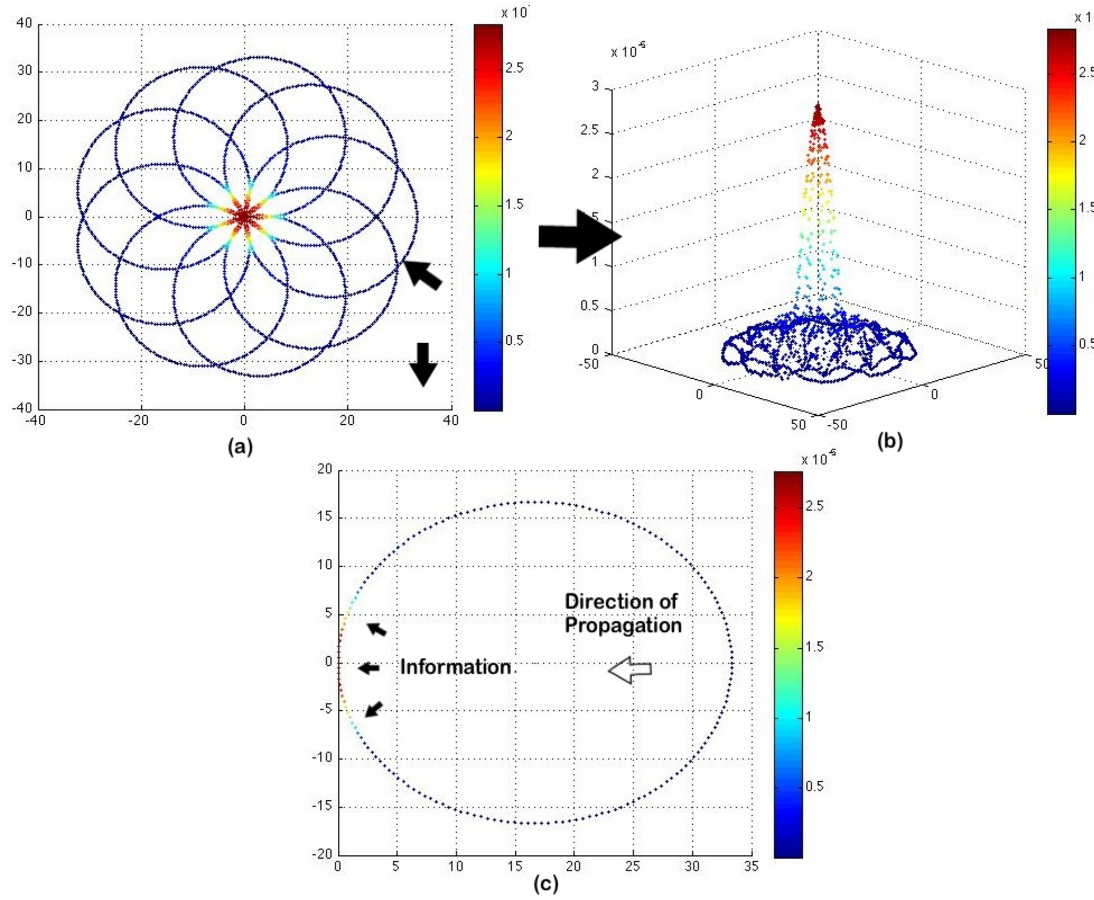


Figure 3.4. (a) Fourier representation from 9 Ewald circles obtained from 9 separate source directions around a target. (b) Isometric view of Fourier domain image in (a). (c) One Ewald circle indicated by arrow in (a) showing direction of incident wave and resultant location of non-zero receiver data showing comparison to transmission problem as the target or “transmission medium” is located in the center of the Ewald circle.

CHAPTER 4: SYSTEMATIC METHOD FOR PRODUCING SIMULATED SCATTERED FIELD DATA FROM KNOWN STRUCTURES

4.1 Target Modeling and Imaging Data Generation

The task of 2-dimensional target modeling and image data generation from scattered fields is not a trivial one. The problem lies in that there is no general solution for analytically determining scattered fields for an arbitrary target. There are some analytical solutions only for a few very simple targets, but in general, no solution exists. This means that for varying case by case situations the analytical solution would have to be derived each time for a new target, if in fact the analytical solution does exist at all in a closed empirical form, which is unlikely. A common numerical solution to this type of problem or modeling is to use the technique of Finite Element Analysis [20] [21]. In this method, the differential equations involved in calculating these scattered fields are solved for numerically in an iterative process. The basic model setup for this procedure is much like the general model shown in Figure 1.1, with the exception that there is an artificial boundary that defines the extent that the iterative calculations are performed for, since this is a finite method as shown in Figure 4.1. At this boundary the properties of the boundary are defined such that there are no reflections and it gives the "appearance" that the model space goes on forever. The general solution for an E_Z polarized field in the model space satisfies the scalar Helmholtz equation as follows:

$$\frac{\partial}{\partial x}\left(\frac{1}{\mu_r}\frac{\partial E_z}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{1}{\mu_r}\frac{\partial E_z}{\partial y}\right)+k_0^2\varepsilon_r E_z = jk_0 Z_0 J_z \qquad (4.1)$$

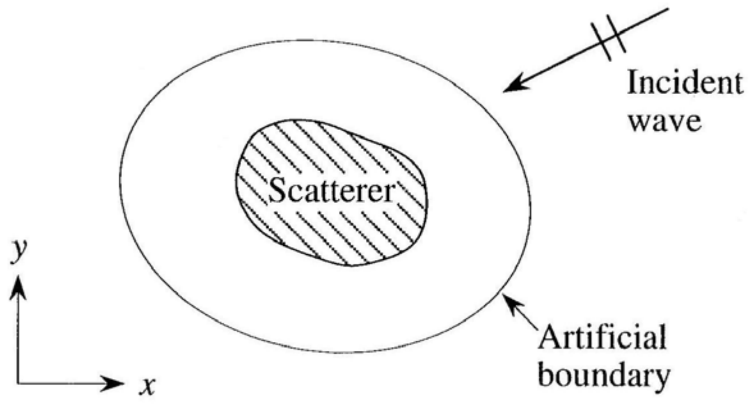


Figure 4.1. The general two-dimensional finite element scattering model.

This is the basic general equation that is used for the finite element method in the model space that is solved iteratively along the finite element mesh. The bounded area is enclosed using Perfectly Matched Layers that utilize the general relationship along the mesh of

$$\frac{\partial \phi^{sc}}{\partial \rho} + \left(jk_0 + \frac{1}{2\rho} \right) \phi^{sc} = 0 \tag{4.2}$$

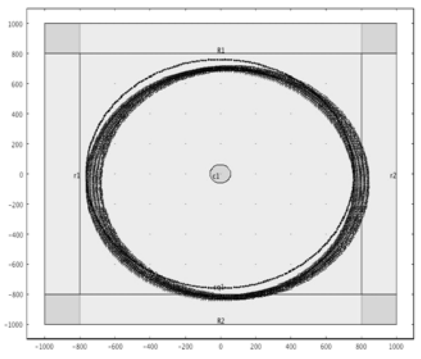
This method is implemented using the commercially available finite element software COMSOL® in this effort. This method and software are used to calculate the total field at each receiver location. This greatly reduces the complexity of the approach to these types of problems, but can be computationally costly. This software allows the user to create the target graphically, modify and/or sweep virtually any and all parameters, and then the program applies the finite element process to the model and returns both a graphical and numerical solution for the total field in the defined space. The only challenge then is to process the data into a format that can be used by imaging algorithms in MATLAB®, which can easily be done in a commercially available spreadsheet such as Microsoft® Excel.

The basic COMSOL® model is similar to that shown in the Section 2.2. The model will be set up as shown in Figure 2.1 with a fixed number of receivers equally spaced at a fixed and common distance from the target origin and a fixed number of sources equally spaced around the target. This should produce some high quality data files to be used to test new and existing imaging algorithms. For the purpose of evaluating the data generated using this method, the imaging technique utilizing the Ewald circles [8] along with the Born approximation is used on the data to produce a Fourier image of the target so that a reconstructed image can be displayed and compared to measured data for identical targets. This method is discussed at length in [6] and [22].

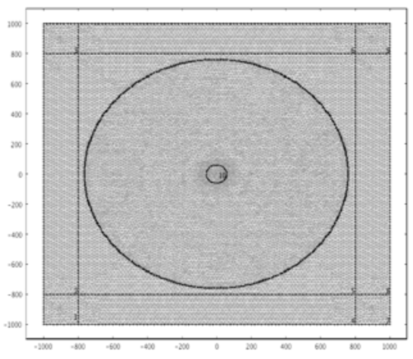
4.2 Target Modeling Environment

The typical model consists of a target area located at the origin. Receiver points, which can be used to "measure" or obtain the calculated complex scattered field values at these points, are located on an imaginary circle centered at the origin with a fixed radius of 760mm. There are 360 data points equally spaced along this circle which basically give the ability to measure the total field on 1 degree increments around the target. The illuminating source is cycled around the target in 36 equally spaced locations which basically gives the ability to view the scattered data from a source rotated in 10 degree increments around the target. This translates to having the capability of creating 36 separate Ewald circles as previously discussed in [8]. As already mentioned this basic model environment has been successfully implemented and data has been successfully gathered from the simulations and formatted for use in already developed MATLAB® algorithms used in [6]. This will be demonstrated in the next section.

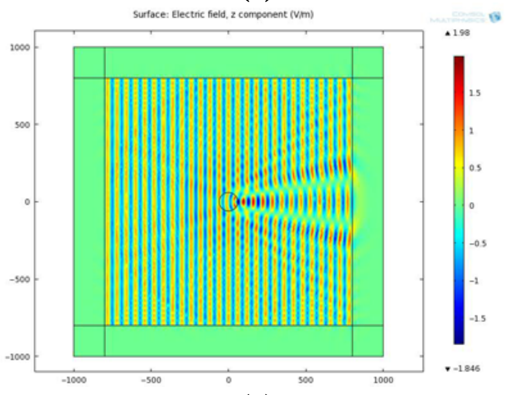
The implementation of this environment in the COMSOL® software for a basic cylinder target is shown in Figure 4.2. In this figure, the basic model, mesh, Z component of the E field which is orthogonal to the plane of propagation, and the normalized E field are shown to illustrate the capabilities of the software and model. It is apparent that this modeling technique is functioning properly and should provide valid data. The data obtained from this modeling process is now exported to a Microsoft Excel® spreadsheet and processed to be formated to be used as a data file in MATLAB.



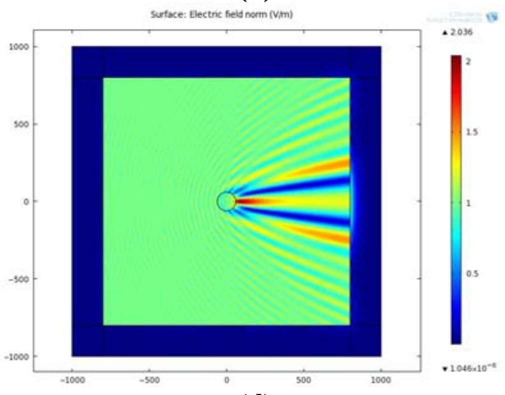
(a)



(b)



(c)



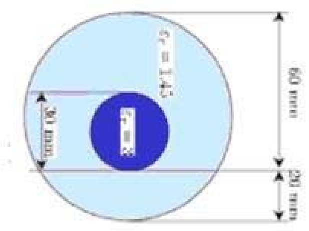
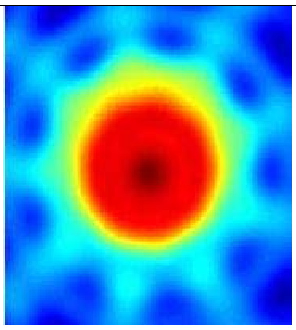
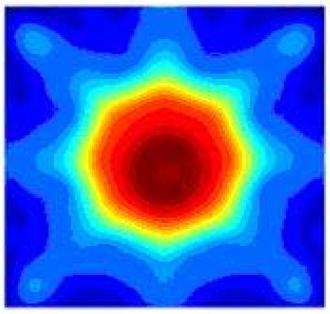
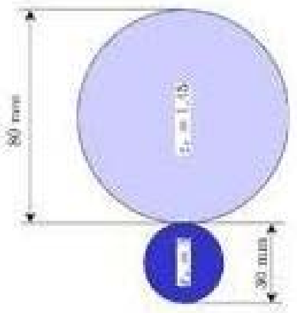
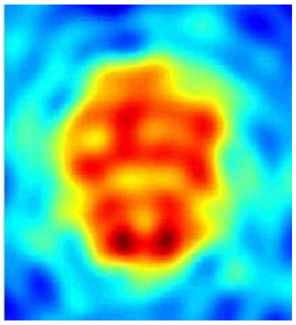
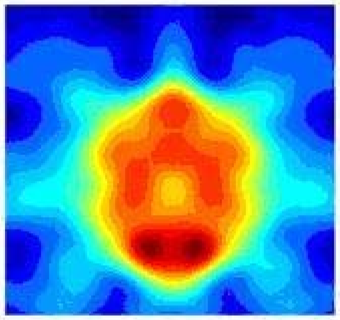
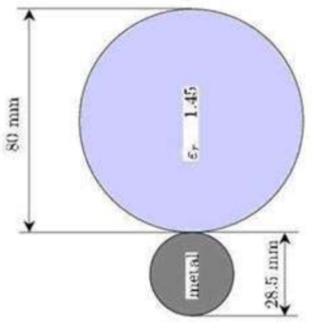
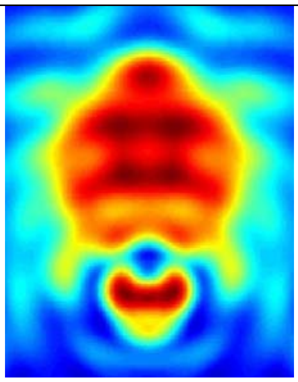
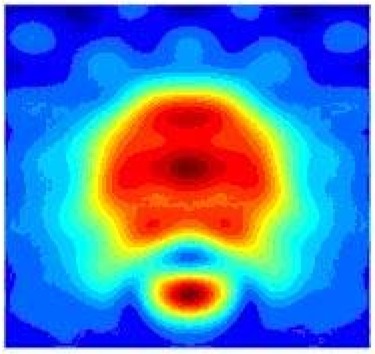
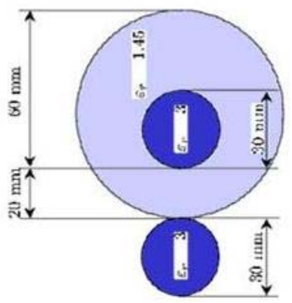
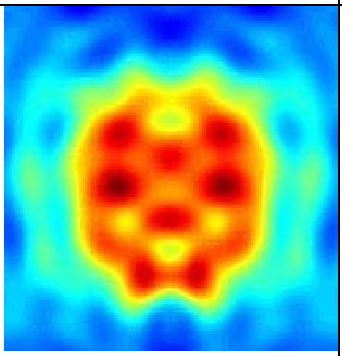
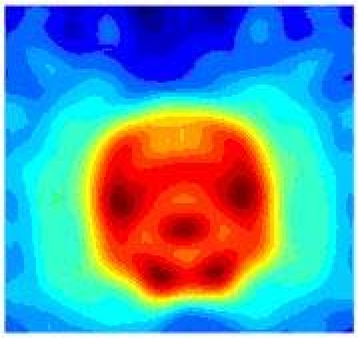
(d)

Figure 4.2. (a) Basic model of a circular target (i.e. a cylinder in 3D) in the x,y plane with a radius of 60mm. The larger circle is the location of the 360 receiver points. The square and rectangular sections around the border of the space are Matched Boundary Layers. (b) Basic target model from (a) with Finite Element Mesh Applied.
(c) Graphical representation of Ez with incident source frequency of 5GHz for a target in (a) with a relative permittivity of 1.5.
(d) Graphical representation of normalized Ez for conditions in (c)

4.3 Imaging Algorithm Results

In order to verify the validity of this modeling process, the data obtained from the COMSOL® model was extracted, similarly processed, and compared to measured data obtained from the Institut Fresnel [5] website for a range of targets. This data was similarly processed using basically the same algorithm described in [6] that maps the data for a given source frequency onto Ewald circles and then concurrently applies the Born approximation algorithm to the data to produce a first Born approximation reconstructed image in MATLAB®. To demonstrate that the data from the COMSOL® / MATLAB® modeling process is valid compared to measured data, the two images for each target, one from the reconstructed images obtained from the measured data in [6], and one from the simulated data processed using the same algorithm respectively are shown to be comparable in appearance. The target definitions, and outputs from both data sets is presented in Figure 4.3 for comparison below. There are some obvious differences in appearance which is to be expected to some degree. One major difference in the data sets for at least the first two targets is that the simulated data was constructed using nine incident sources equally spaced around the target and the measured data was constructed using eight incident sources equally spaced around the target. The last two targets both utilize 18 source locations, again equally spaced around the target. Otherwise, the setups are almost identical for each target set.

Table 4.1. Institut Fresnel Target definitions [5], Simulated Processed Data Output, and Measured Data Output [6].

	Institut Fresnel Data Target Setup	Born Image from Simulated Data	Born Image from Institut Fresnel Measured Data
FoamDielInt	 $\varepsilon_r = 1.45$ $\varepsilon_r = 3$ 30 mm 60 mm 20 mm		
FoamDielExt	 80 mm $\varepsilon_r = 1.45$ $\varepsilon_r = 3$ 30 mm		
FoamMetExt			
FoamTwinDiel	 ε_r 1.45 ε_r 3 60 mm 30 mm 20 mm ε_r 3 30 mm		

CHAPTER 5: RECENT SIGNAL PROCESSING BASED SOLUTIONS FOR SOLVING INVERSE SCATTERING PROBLEMS

5.1 Homomorphic Filtering

As explained in section 3.2 above, the Born Approximation performs well when the target is a weak scatterer, since one can replace the total field inside the target by the known incident field, and this leads naturally to the Fourier transform relation described earlier. However, as the permittivity of the target increases, the "Born" algorithm's performance decreases. This is to be expected due to the fact that less of the incident wave might be penetrating and propagating through the target, but is reflected off the surface of the target as well as being scattered multiple times from inhomogeneities that exist inside the target. The reflected wave and emerging more highly structured scattered field components need to be interpreted as carrying information about V(**r**) or, depending on how the scattered field is processed, as noise-like terms arising from strong scattering that one might be able to remove, thereby reducing a strongly scattering target to one that can be imaged more like a weakly scattering one. In the cepstral method, the total field estimated within the target volume is regarded as a form of spatial noise to be removed. When this is justified, the reconstructed image of the target, based on assuming the first Born approximation, can be expressed as

$$V_{BA}(\mathbf{r},\hat{\mathbf{r}}_{inc}) \approx \mathrm{V}(\mathbf{r})\left\langle \frac{\Psi(\mathbf{r},\hat{\mathbf{r}}_{inc})}{\Psi_{inc}(\mathbf{r},\hat{\mathbf{r}}_{inc})} \right\rangle \tag{5.1}$$

where $\left\langle \Psi(\mathbf{r},\hat{\mathbf{r}}_{inc}) / \Psi_{inc}(\mathbf{r},\hat{\mathbf{r}}_{inc}) \right\rangle$ is a symbolic representation for a complex and noise-like term with a characteristic range of spatial frequencies dominated by the average local wavelength of the source. The problem is now reduced to a complex filtering problem in which the multiplicative term $\langle \Psi(\mathbf{r},\hat{\mathbf{r}}_{inc}) / \Psi_{inc}(\mathbf{r},\hat{\mathbf{r}}_{inc}) \rangle$ term needs to be filtered out of the data.

There are a number of ways to approach this problem, but one of the more recent methods is a well known and documented method based on cepstral filtering, a technique originally developed to eliminate multiplicative noise [17] [18]. The homomorphic filtering method uses the log operation to convert a multiplicative modulation relationship into an additive relationship. This is demonstrated here by taking the complex logarithm of Eq. 5.1 as follows:

$$\log\left(\mathrm{V}(\mathbf{r})\left\langle \Psi(\mathbf{r},\hat{\mathbf{r}}_{inc}) / \Psi_{inc}(\mathbf{r},\hat{\mathbf{r}}_{inc}) \right\rangle\right)$$

$$= \log\left|\mathrm{V}(\mathbf{r})\right| + \log\left|\left\langle \Psi(\mathbf{r},\hat{\mathbf{r}}_{inc}) / \Psi_{inc}(\mathbf{r},\hat{\mathbf{r}}_{inc}) \right\rangle\right| + i\left[\arg\left[\mathrm{V}(\mathbf{r})\right] + \arg\left(\left\langle \Psi(\mathbf{r},\hat{\mathbf{r}}_{inc}) / \Psi_{inc}(\mathbf{r},\hat{\mathbf{r}}_{inc}) \right\rangle\right)\right] \quad (5.2)$$

If the Fourier transform is now taken of the equation in Eq. 5.2, the results will be the complex cepstrum representation of the target or $\mathrm{V}(\mathbf{r})\Psi / \Psi_{inc}$ [17] [18] . With the data now in this form, the phase information of the complex data is retained which is essential in doing any type of useful filtering processes. It should be noted that there are numerous potential problems that can occur when calculating the complex $\log(\mathrm{V}\Psi)$ and taking its Fourier transform. There can be unwanted harmonics introduced by taking the complex logarithm of sampled data where their magnitude is less than 1 and approaches zero, which could lead to aliasing in the cepstral domain [17]. There is also the common concern of branch cuts associated with singularities caused by zeros in the data representation of $\mathrm{V}(\mathbf{r})\Psi$ since the logarithm function of zero is strictly undefined. Thus the complex logarithmic function can be multi-valued if the imaginary part of the data representing V(**r**)Ψ exceeds 2π. When this happens then this results in what is called phase wrapping which can be extremely difficult to deal with especially in two or more dimensions. These phase wraps present discontinuities which generate many frequencies in the Fourier domain being associated with zeros in the field data corresponding to dislocated wave-fronts [19].

5.2 Cepstral Filtering with Minimum Phase

In previous research [6], it has been demonstrated that if the cepstral data are made to be causal and minimum phase, and a spatial filter in the cepstral domain is chosen properly, the ability to eliminate, or at least greatly reduce or attenuate, the presence of the $\left\langle \Psi(\mathbf{r},\hat{\mathbf{r}}_{inc}) / \Psi_{inc}(\mathbf{r},\hat{\mathbf{r}}_{inc}) \right\rangle$ term can be achieved leading to a much better reconstructed image of V(**r**) than that obtained using the Born approximation alone. The first two characteristics mentioned above, i.e., forcing the data to be causal and to be minimum phase are crucial in the success of this approach. These conditions are imperative as demonstrated in [6] and they form the basis for the success of this approach. This approach will be thoroughly examined and extended in this paper, so this being the case, it seems prudent to review the aspects of this approach here.

The issue of processing the data to be causal is a simple one in that this can be achieved by shifting all of the data into the first quadrant of an expanded space whose origin is defined to be at its center. This is basically reduced to a function of "re-indexing" the indices of the data in the computer code. The second aspect mentioned above is that of making the data to be minimum phase. In 1-dimension this concept is well understood [28]. In this case, a 1-dimensional signal, *f(x),* is said to be minimum phase if and only if its Fourier transform, *F(z)*, has a zero-free upper half plane. In other words, *F(z)* has no zeros for $v > 0$ where $z = u + iv$ and

$$F(u) = \int_{-\infty}^{+\infty} f(x) e^{i2\pi ux} dx \tag{5.3}$$

Functions of this type have some very useful properties, such as [28]

i. *f (x)* is causal.

ii. The phase of *F(u)* lies between –π and +π; i.e. its phase is always unwrapped (non-discontinuous).

iii. Most of the energy in *f(x)* lies close to origin.

iv. *F(u)* is absolutely summable.

Another very interesting and important aspect of minimum phase functions is that these functions have a “minimum energy delay” property. This is important because it results in a function that possesses the highest “partial” energy among all functions that have the same Fourier magnitude [29]. As stated before, a function, *f(x)*, is defined as minimum phase if its Fourier transform, *F(u)*, is analytic and has a zero free half plane.

With these definitions in hand, the concept of complex cepstum [30] will now be defined and discussed. The complex cepstrum for a function *f* is defined as taking the complex natural logarithm of the Fourier transform of the function *f*, then taking the inverse Fourier transform. This can be written mathematically as

$$\hat{g} = ifft\left[\log\left(F\right)\right] \;\; where \;\; F = fft(f) \tag{5.4}$$

or, another common way of expressing this is as follows

$$\hat{g}(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \log\left[F(u)\right] e^{i2\pi ux} du \tag{5.5}$$

where $\hat{g}$ is defined as the cepstrum of the input signal *f*. As alluded to earlier, in a one dimensional case, the minimum phase condition is required to ensure that the Fourier transform of a causal function has a zero-free upper half plane. This is important because this now permits Cauchy's integral formula to be used to relate the Fourier magnitude and the phase on the real u-axis. This relationship is known as the logarithmic Hilbert transform.

As discussed in [6], the concept of zero-free half plane in 2-dimensions can be problematic to say the least. Even for a function that is separable such as $F(u_1,u_2)=G(u_1)H(u_2)$, a zero-free upper-half plane in $G(z_1)$ automatically leads to $F(z_1, z_2)$ having zeros in the upper-half plane of z_2 since we know that $H(u_2)$ must have zeros. In practice, functions having a zero-free half plane are rare and there are few general conditions that are known for which these characteristics can be imposed.

In general, in terms of Fourier based theory and analysis of scattering and inverse-scattering, scattered and propagating fields are analytic functions due to the fact that the scattering objects themselves are of finite spatial extent. To go one step further, these fields are entire functions of the exponential type [31] which by definition means that they satisfy the Cauchy-Riemann equation for all finite $z = x + iy$ as expressed here

$$\frac{\partial \operatorname{Re} F}{\partial x} = \frac{\partial \operatorname{Im} F}{\partial y} \tag{5.6}$$

$$\frac{\partial \operatorname{Re} F}{\partial y} = \frac{-\partial \operatorname{Im} F}{\partial x} \tag{5.7}$$

There are of course very strict constraints on these types of functions as to how their amplitude varies or increases and how their zero crossings are distributed in general. In 1-dimension this is well understood and these concepts are, for the most part, indirectly applicable in the 2-dimensional realm. These constraints and characteristics are utilized in [6] and as follows to formulate an approach to insuring that these functions have a zero-free half-plane. The analogy in [6] goes on to show how using the Hilbert Transform relating the real and imaginary parts of the Fourier transform, one can use this relationship to generate a minimum phase function which insures that the phase of the function is unwrapped.

Finally, in [6] it is shown that the issue of insuring that the function is a minimum phase function in 1-dimension can be enforced by applying Rouche's theorem [6] [32] [39]. A 2-dimensional version of this theorem has been validated in [33]. This can be applied to the current 2-dimensional situation to enforce a minimum phase condition. In summary, it is shown in [6] and [39] that if

A has N number of zeros in or on some contour

B has M number of zeros in or on the same contour

and

$|A| > |B|$ on that contour

then

$A + B$ will have N number of zeros in or on that contour; otherwise, for $|A| < |B|$, $A+B$ will have M number of zeros. See Figure 5.1. The sum of the two functions will have the number of zeros equal to the number of zeros of the larger magnitude function. Consequently, adding a sufficiently large uniform and zero-free background or reference wave to a bandlimited function, A, where we assume $|A|<<1$, allows us to write $G = 1 + A \sim \exp(A)$ thereby satisfying this minimum phase condition.

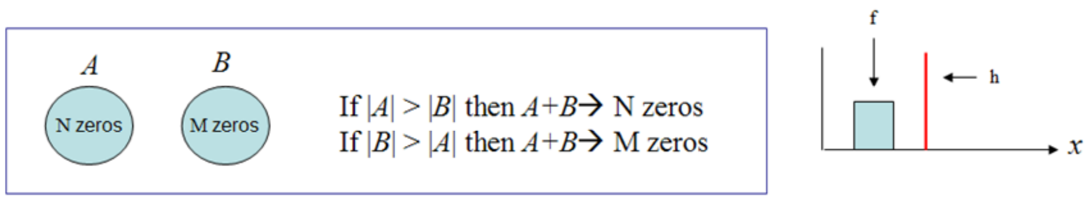

A
B
N zeros
M zeros
If |A| > |B| then A+B→ N zeros
If |B| > |A| then A+B→ M zeros
f
h
x

Figure 5.1. Pictorial description of Rouche's theorem.

In conclusion, applying Rouche's theorem summarized above to our bandlimited function *F(z)*, if we add another function *G(z)* which we will refer to as a reference function and *G(z)* has no zeros in the upper half plane and the magnitude of *G(z)* is greater than the magnitude of *F(z)*, then the function *F(z)*+*G(z)* will have no zeros in the upper half plane thus satisfying the minimum phase condition. This can easily be accomplished if we choose *G(z)* to be a constant with a magnitude greater than the magnitude of *F(z)*. This is the approach implemented successfully in [6] and continued in this research. Furthermore it has been shown [5] [34] that in order to enforce a minimum phase condition, the reference should satisfy

$$R \geq \left| \mathrm{V}\langle \Psi \rangle \right|_{\max} \tag{5.8}$$

That is that the reference point must be greater than the maximum value of the scattered field amplitude. Also from [6], if we assume that the value of *R* is chosen appropriately, we then can write

$$\log\left(R + \mathrm{V}\langle \Psi \rangle\right) \longrightarrow \log\left(1 + \frac{\mathrm{V}\langle \Psi \rangle}{R}\right) \sim \frac{\mathrm{V}\langle \Psi \rangle}{R} - \frac{1}{2}\left(\frac{\mathrm{V}\langle \Psi \rangle}{R}\right)^2 + \ldots \tag{5.9}$$

which can be re-written as

$$\log\left(1 + \frac{\mathrm{V}\langle \Psi \rangle}{R}\right) = \log\left(\left\{\frac{\mathrm{V}}{R}\right\}\left\{\frac{R}{\mathrm{V}} + \langle \Psi \rangle\right\}\right) = \log\left(\frac{\mathrm{V}}{R}\right) + \log\left(\frac{R}{\mathrm{V}} + \langle \Psi \rangle\right) \tag{5.10}$$

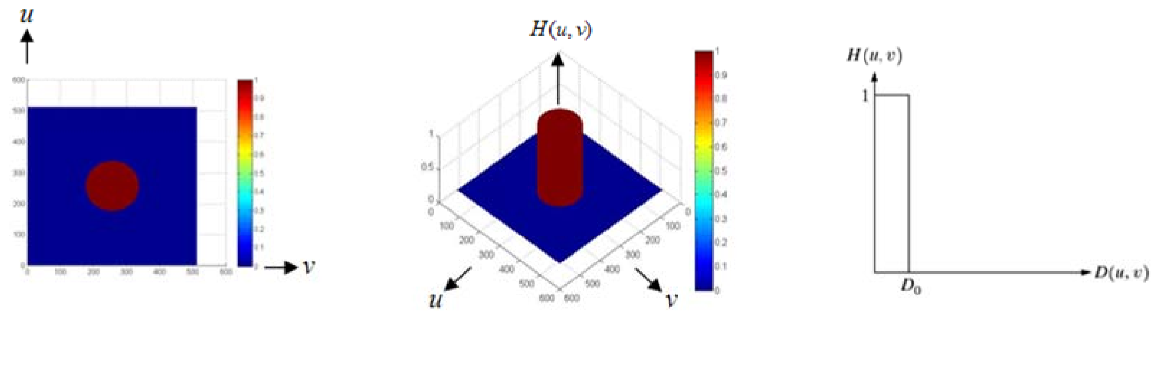
$$\approx \log\left(\frac{\mathrm{V}}{R}\right) + \log\left(R' + \langle\Psi\rangle\right) \tag{5.11}$$

$$\approx \log\left(\frac{\mathrm{V}}{R}\right) + \frac{\langle\Psi\rangle}{R'} \tag{5.12}$$

Now, the second term in Eq. 5.12 should contain spatial frequencies similar to the spatial frequencies of the illuminating incident field. It should be possible to remove or attenuate this second "noise-like" term by either choosing the proper filter as done in [6] and optimized later in this research, or possibly by subtracting out some weighted representation of the incident field in the cepstrum domain, which will be experimented with later in this research as well, or some combination of both. We note that V is recovered weighted by *R*, which for a quantitative image needs to be removed.

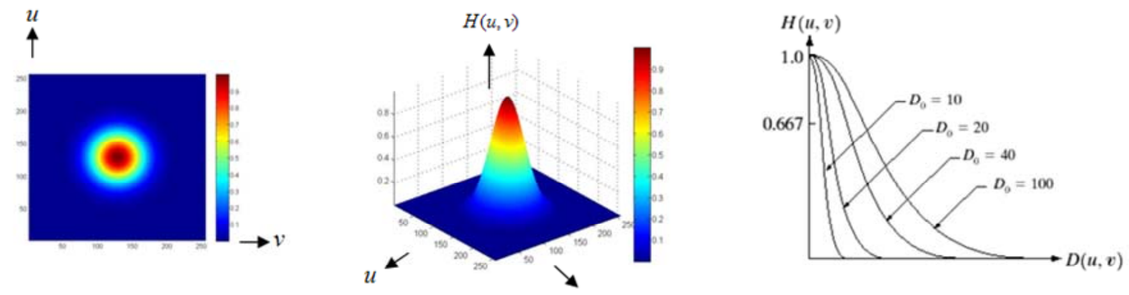
5.3 Two Dimensional Filtering Methods

Once the data are properly processed as discussed in the previous section, and represented in the cepstrum domain, it is now necessary to filter the data in an attempt to eliminate or at least attenuate the Ψ term and obtain a better representation of $\mathrm{V}(\mathbf{r})$. Since the pre-processed data are now minimum phase, we know as stated earlier that one of the main characteristics of minimum phase data is that most of the energy related to $\mathrm{V}(\mathrm{r})<\Psi(\mathrm{r})>$ may be located near the origin. This would suggest that some type of low pass filter should work well if chosen properly. In basic filter theory [40] [41], there are two basic types of low pass filters to consider. They are the ideal low pass filter illustrated in Figure 5.2, and the Gaussian low pass filter illustrated in Figure 5.3. In each of these figures, a top view, isometric view and slice or profile view is shown for each filter. In addition, for the Gaussian filter, the profile is shown for various values of sigma which in effect defines the filter bandwidth.



(a) (b) (c)

Figure. 5.2. Low pass filter (a) ideal hard-cut low pass filter 2D view, (b) low pass filter displayed in 3D, (c) cross section of ideal low pass filter



(a) (b) (c)

Figure. 5.3. Gaussian low pass filter (a) Gaussian low pass filter 2D view, (b) Gaussian low pass filter displayed in 3D, (c) Cross section of Gaussian low pass filter

As is commonly known in basic filter theory [41], and verified in [6], the performance of the Gaussian is generally better than the ideal filter since the ideal filter is prone to produce "ringing" in the reconstructed image. This being the case, the Gaussian filter will be used in the filtering process in the cepstrum domain. This still leaves the question as to what are the optimum values for the parameters for this filter. More specifically, what should the value of sigma be for the filter, which governs the bandwidth of the filter, and what constraints and effects for the peak value of the filter? These parameters will be examined in detail in Section 6.

One final aspect that will be examined in this research pertaining to filtering or attenuating the Ψ term is that of subtracting a cepstrum version of Ψ_0 in the cepstrum domain in an attempt to improve the reconstructed image, or possibly improve the scale of the reconstructed image. If we review for a moment the final version of the $V\Psi$ which strictly corresponds to $V\Psi/\Psi_{inc}$ expression obtained in Eq. 5.12 we have

$$\approx \log\left(\frac{V}{R}\right) + \frac{\langle\Psi\rangle}{R'}$$

As stated earlier it is evident that there is a "noise" term in this expression that appears to be of the form of a "weighted" incident frequency type term, depending on how strong of a scatterer the target is. This being the case, it seems that it would be beneficial to take the incident field and obtain a weighted cepstrum representation of it and subtract it from the expression above. Assuming the correct weighting factor could be found, it appears that this would have some positive benefit to the resulting cepstrum representation of V(**r**). Moreover, there will be some benefit to trying to eliminate or account for the reference *R* term as well. This obviously should have some effect on the overall scaling of the resulting data in theory. If a close representation for the weighted Ψ term can be found then the resulting form of the expression above would be

$$\approx \log\left(\frac{V}{R}\right) + \frac{\langle\Psi\rangle}{R'} - \kappa\langle\Psi_{inc}\rangle \tag{5.13}$$

Again, if the factor for "κ" can be found that is close to 1/*R'*, and $<\Psi_{inc}>$ is a close approximation to $<\Psi>$ then improvement in the resulting reconstructed image can be expected.

CHAPTER 6: EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Analysis of Requirements for Degrees of Freedom For Sources

In this section, a series of "families" of reconstructions are presented in a systematic way for a group of selected targets all illuminated by an increasing number of incident waves all with a frequency of 5GHz. The targets of choice for this analysis are defined and described in Table 6.1.

Table 6.1. Table defining the various target types and scenarios used to test the degrees of freedom requirement along with corresponding table number of results.

Table Number	Shapes	Dimensions	Permittivity Range
6.2	1 Circle	Radius = 1λ	2 - 10
6.3	1 Circle	Radius = 1λ	11 - 19
6.4	1 Circle	Radius = 2λ	2 - 10
6.5 / 6.11	2 Circles	Radius = 1λ	1.1 – 1.9
6.6	2 Circles	Radius = 1λ	2 - 10
6.7 / 6.12	1 Square	Sides = 1λ	1.1 – 1.9
6.8 / 6.13	2 Squares	Sides = 1λ	1.1 – 1.9
6.9 / 6.14	2 Triangles	Base/Height = 2λ/3λ	1.1 – 1.9

For each of these target parameters, the degrees of freedom and the weakly scattering metric (kVa), discussed in Section 3.2, is calculated and shown for reference. Also, to aid in the analysis of these images, the borders of each image are shaded in either "red" or "green" with "red" indicating that for the given parameters and number of sources, the minimum degrees of freedom are not satisfied, and the "green" indicating that for the given parameters and number of sources, the minimum degrees of freedom are satisfied. With this color code the reader should be able to distinguish as the number of sources is increased (moving “down” the table) when the minimum criteria are met.

It should be noted that there are obviously other factors that are affecting the reconstruction in these images, namely obvious resonances that occur for different parameters that definitely affect some of the images as compared to others. These phenomena will be examined in more detail in following sections. It should also be kept in mind that the measure of and meeting of the number of degrees of freedom do not necessarily indicate that the image will look exactly like the target because the inverse scattering problem remains; it simply means that for the given parameters, adequate sources have been used to “communicate” or transfer as much information about the target as is possible for the given experimental arrangement. So, with this in mind, one would expect that after the minimum degrees of freedom have been met, and for all images with higher degrees of freedom, the image of the “target” should not change significantly from one image to the next. This is not to say that the entire image will not change as there could be significant differences in the “noise” artifacts in the free space areas, but that the image of the target itself, no matter how close or far it is from the original target, should basically remain the same as the number of sources is increased.

Using the criteria described above, it seems evident that for images obtained when measurements are below the degree of freedom threshold do not seem to be well formed and do change from image to the next. While the images obtained above the degrees of freedom threshold do seem to have a consistent reconstructed image as the number of sources increases. This would strongly suggest that the concept of degrees of freedom for this scenario is valid. This issue will be examined further in relation to number of receivers in later sections.

Table 6.2. Family of first Born approximation reconstructions for a cylinder with a radius of 60mm (1λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 2 to 10. There are 360 receivers, a resolution of 250x250 on each image with increasing number of sources going down the page for each permittivity value. Calculated minimum degrees of freedom (number of sources) and weak scattering metric are shown for reference.

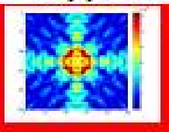
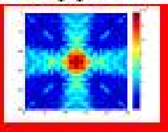
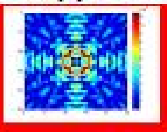
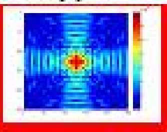
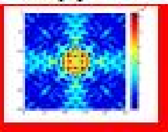
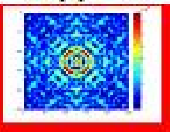
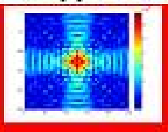
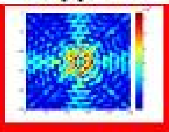
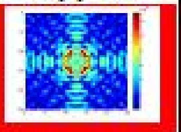
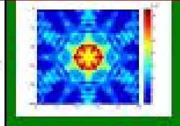
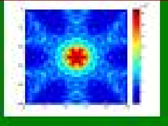
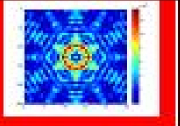
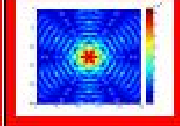
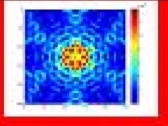
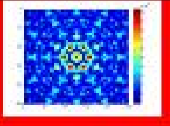
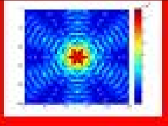
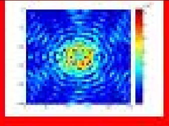
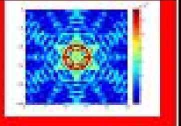
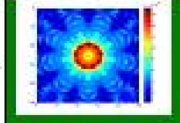
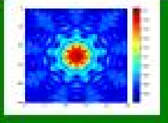
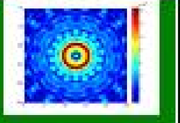
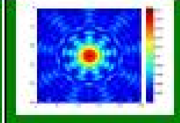
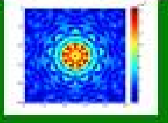
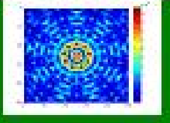
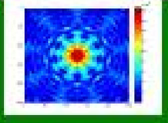
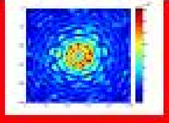
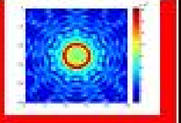
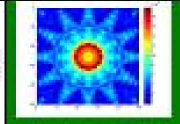
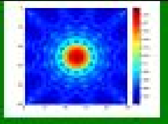
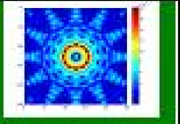
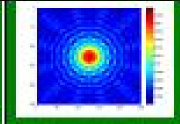
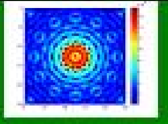
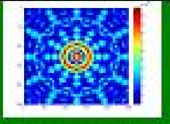
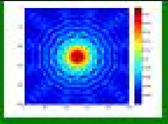
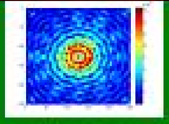
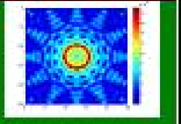
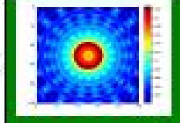
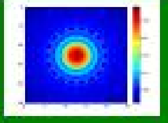
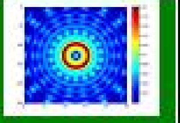
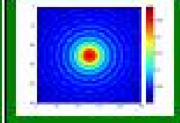
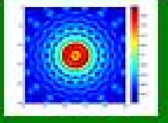
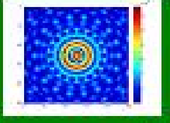
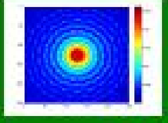
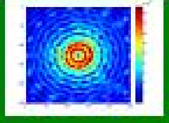
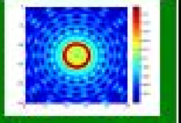
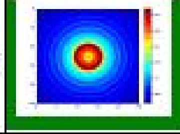
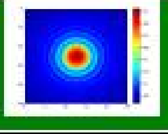
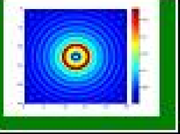
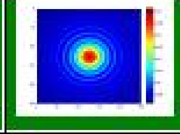
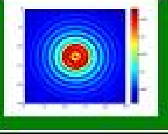
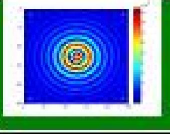
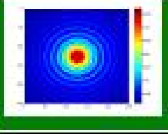
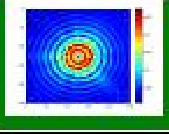
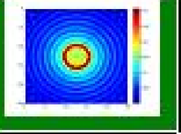
	er=2	er=3	er=4	er=5	er=6	er=7	er=8	er=9	er=10
	N=4.44	N=5.44	N=6.28	N=7.02	N=7.70	N=8.31	N=8.89	N=9.42	N=9.93
	kV(r)a=6	kV(r)a=13	kV(r)a=19	kV(r)a=25	kV(r)a=31	kV(r)a=38	kV(r)a=44	kV(r)a=50	kV(r)a=57
4	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
6	🟩 	🟩 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
9	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟥 	🟥 
12	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
18	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
36	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 

Table 6.3. Family of first Born approximation reconstructions for a cylinder with a radius of 60mm (1λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 11 to 19. There are 360 receivers, a resolution of 250x250 on each image with increasing number of sources going down the page for each permittivity value. Calculated minimum degrees of freedom (number of sources) and weak scattering metric are shown for reference.

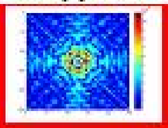
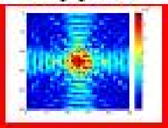
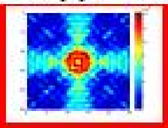
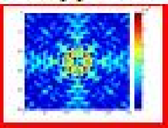
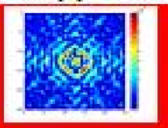
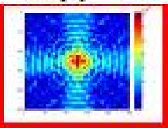
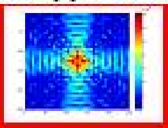
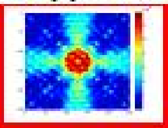
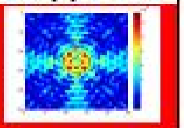
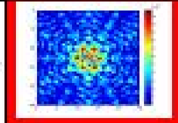
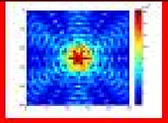
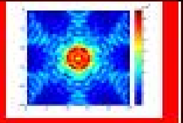
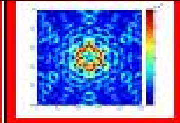
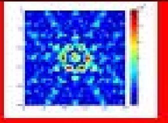
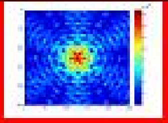
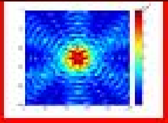
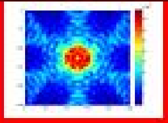
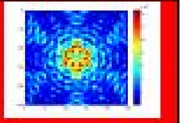
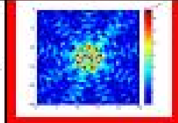
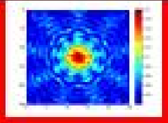
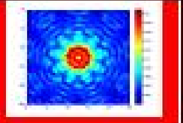
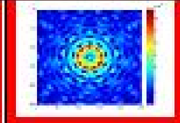
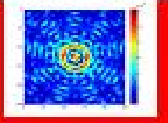
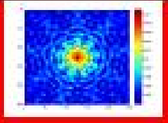
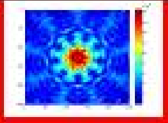
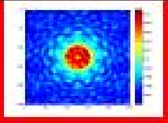
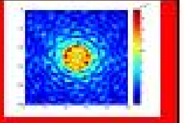
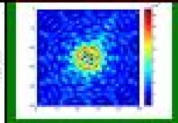
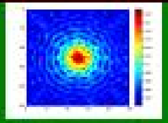
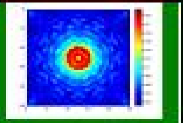
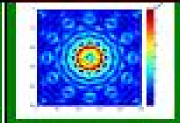
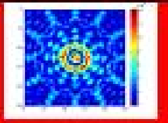
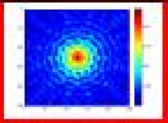
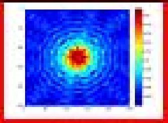
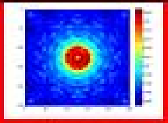
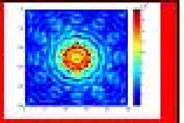
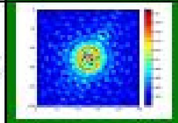
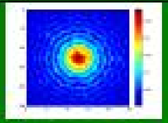
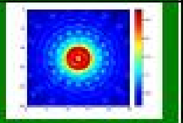
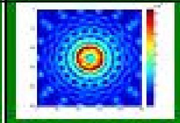
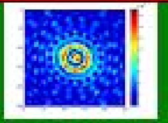
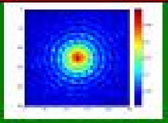
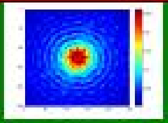
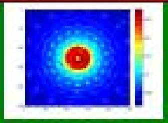
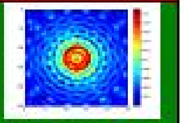
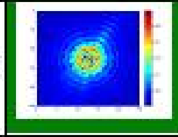
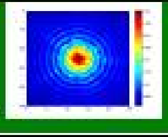
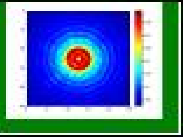
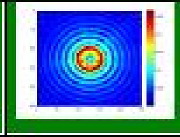
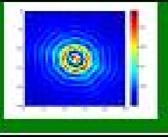
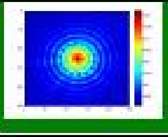
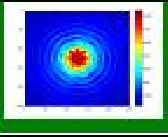
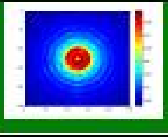
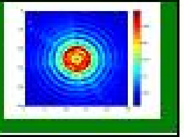
	er=11	er=12	er=13	er=14	er=15	er=16	er=17	er=18	er=19
	N=10.42	N=10.88	N=11.33	N=11.75	N=12.17	N=12.57	N=12.95	N=13.33	N=13.69
	kV(r)a=63	kV(r)a=69	kV(r)a=75	kV(r)a=82	kV(r)a=88	kV(r)a=94	kV(r)a=100	kV(r)a=107	kV(r)a=113
4	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
6	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
9	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
12	🟩 	🟩 	🟩 	🟩 	🟥 	🟥 	🟥 	🟥 	🟥 
18	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
36	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 

Table 6.4. Family of first Born approximation reconstructions for a cylinder with a radius of 120mm (2λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 2 to 10. There are 360 receivers, a resolution of 250x250 on each image with increasing number of sources going down the page for each permittivity value. Calculated minimum degrees of freedom (number of sources) and weak scattering metric are shown for reference.

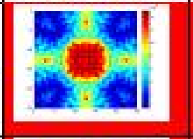
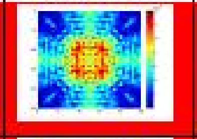
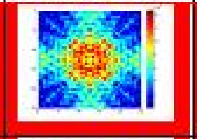
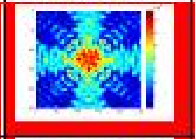
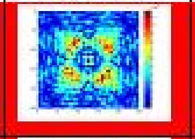
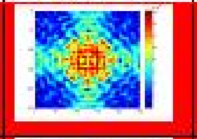
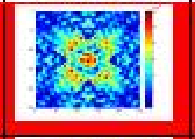
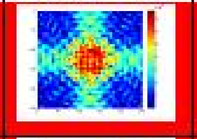
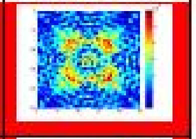
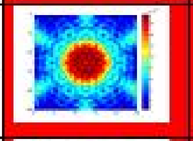
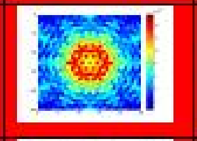
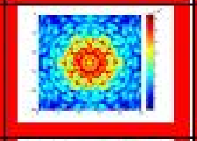
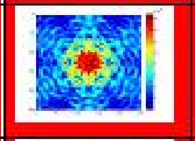
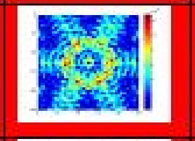
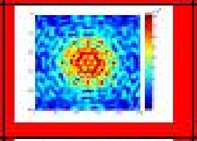
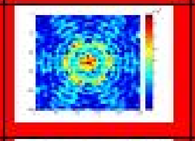
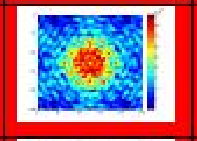
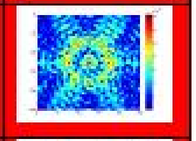
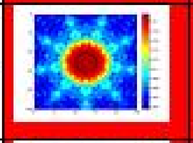
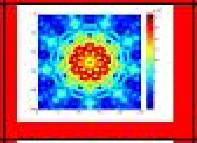
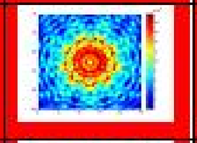
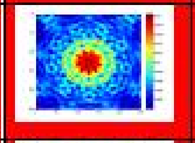
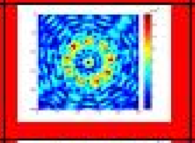
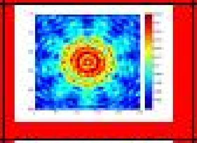
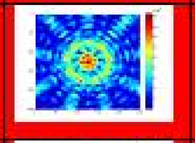
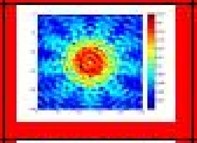
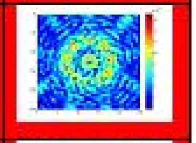
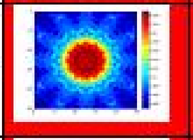
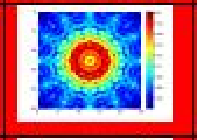
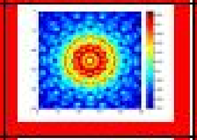
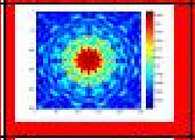
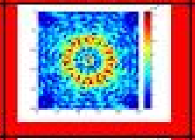
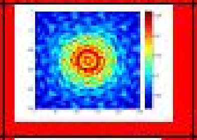
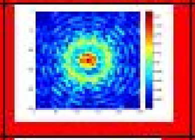
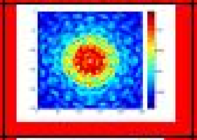
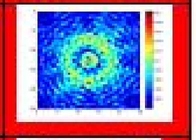
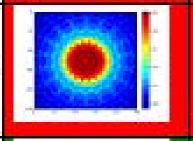
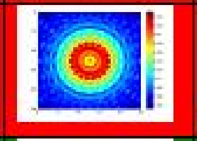
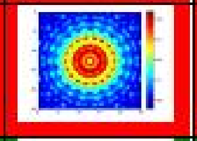
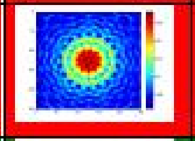
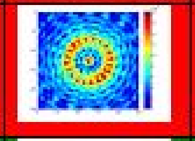
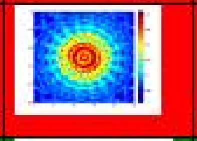
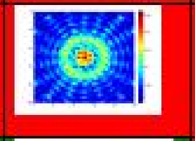
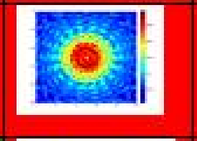
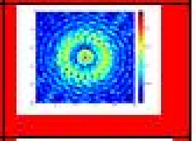
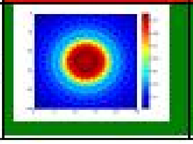
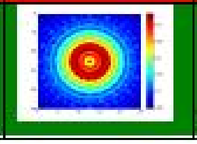
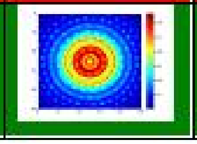
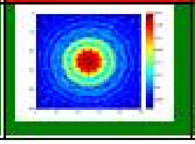
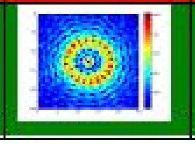
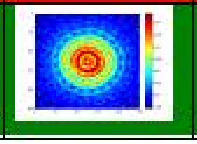
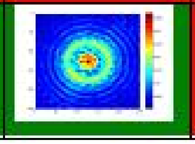
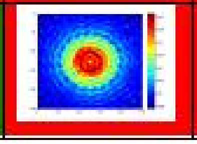
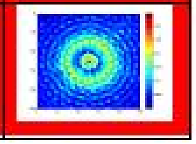
	er=2	er=3	er=4	er=5	er=6	er=7	er=8	er=9	er=10
	N=17.7	N=21.77	N=25.13	N=28.10	N=30.78	N=33.25	N=35.54	N=37.70	N=39.74
	kV(r)a=13	kV(r)a=25	kV(r)a=38	kV(r)a=50	kV(r)a=63	kV(r)a=75	kV(r)a=88	kV(r)a=101	kV(r)a=113
4									
6									
9									
12									
18									
36									

Table 6.5. Family of first Born approximation reconstructions for a pair of cylinders with a radius of 60mm (1λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 1.1 to 1.9. There are 360 receivers, a resolution of 250x250 on each image with increasing number of sources going down the page for each permittivity value. Calculated minimum degrees of freedom (number of sources) and weak scattering metric are shown for reference.

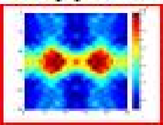
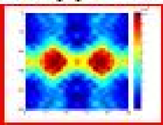
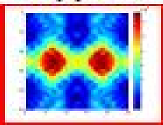
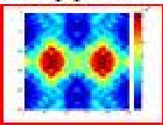
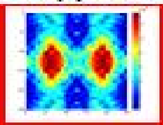
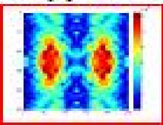
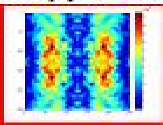
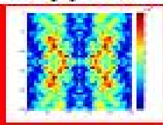
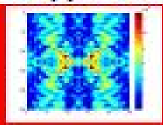
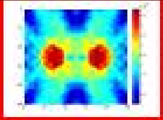
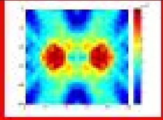
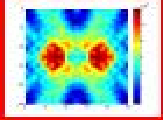
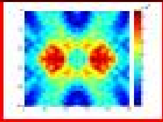
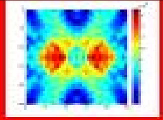
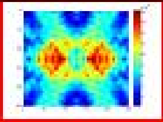
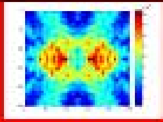
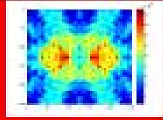
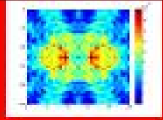
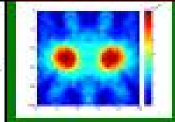
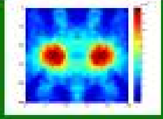
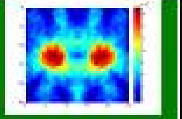
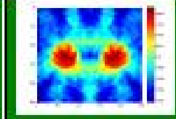
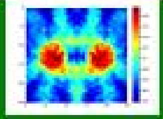
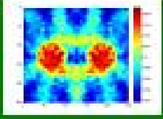
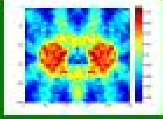
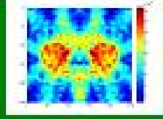
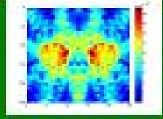
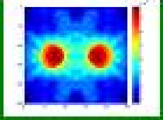
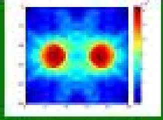
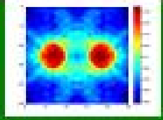
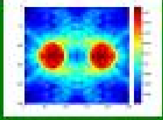
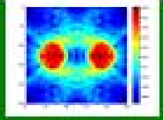
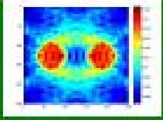
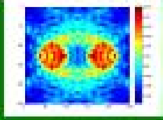
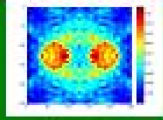
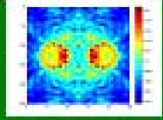
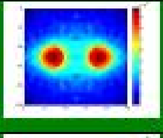
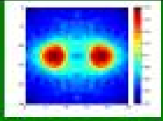
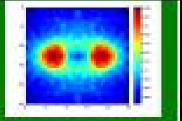
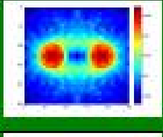
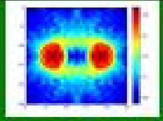
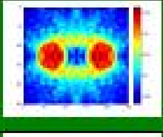
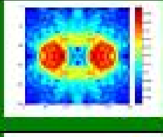
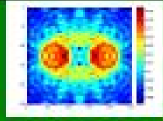
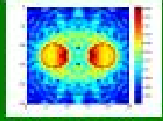
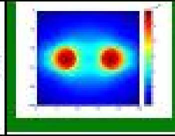
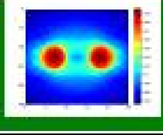
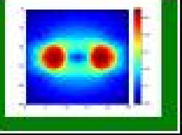
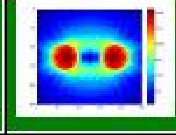
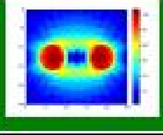
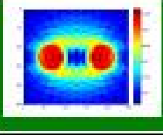
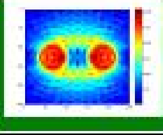
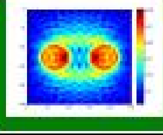
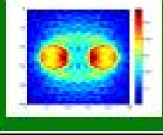
	er=1.1	er=1.2	er=1.3	er=1.4	er=1.5	er=1.6	er=1.7	er=1.8	er=1.9
	N=6.59	N=6.88	N=7.16	N=7.43	N=7.70	N=7.95	N=8.19	N=8.43	N=8.66
	kV(r)a=1.9	kV(r)a=3.8	kV(r)a=5.7	kV(r)a=7.5	kV(r)a=9.4	kV(r)a=11.3	kV(r)a=13.2	kV(r)a=15.1	kV(r)a=17.0
4									
6									
9									
12									
18									
36									

Table 6.6. Family of first Born approximation reconstructions for a pair of cylinders with a radius of 60mm (1λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 2 to 10. There are 360 receivers, a resolution of 250x250 on each image with increasing number of sources going down the page for each permittivity value. Calculated minimum degrees of freedom (number of sources) and weak scattering metric are shown for reference.

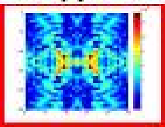
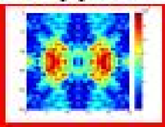
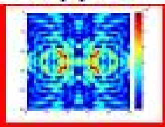
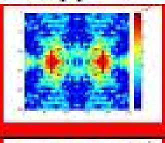
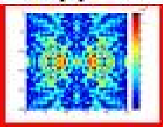
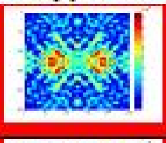
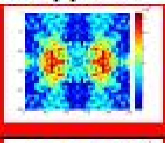
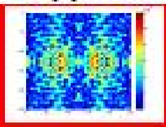
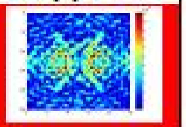
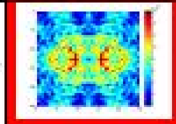
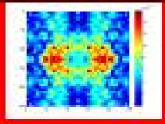
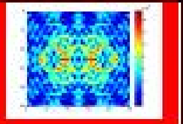
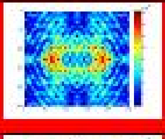
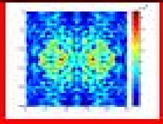
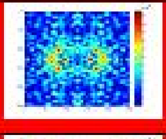
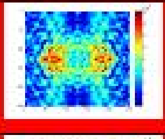
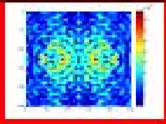
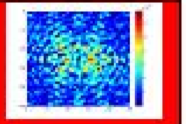
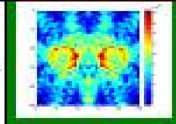
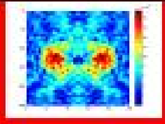
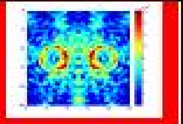
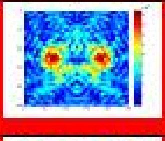
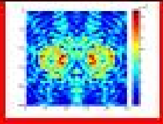
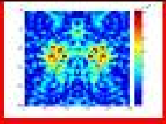
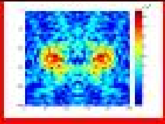
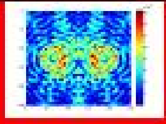
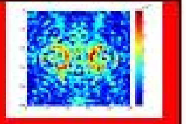
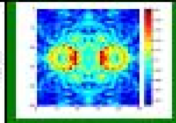
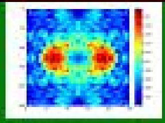
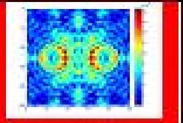
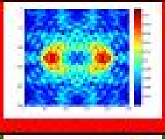
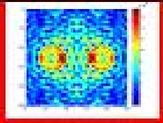
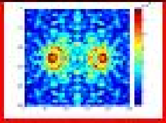
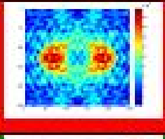
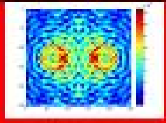
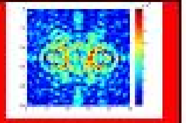
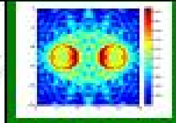
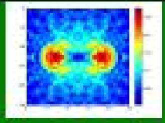
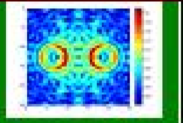
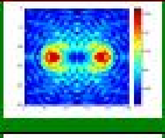
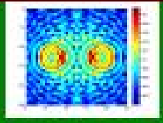
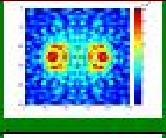
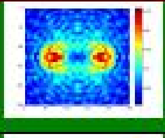
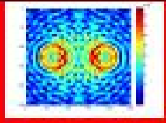
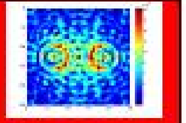
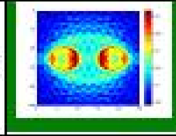
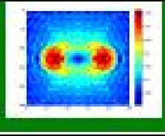
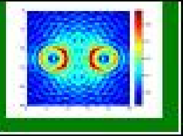
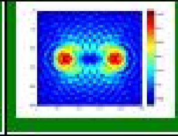
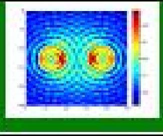
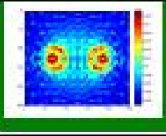
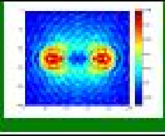
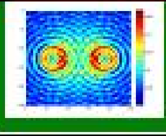
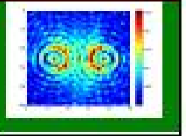
	er=2	er=3	er=4	er=5	er=6	er=7	er=8	er=9	er=10
	N=8.89	N=10.88	N=12.57	N=14.05	N=15.36	N=16.62	N=17.77	N=18.85	N=19.87
	kV(r)a=19	kV(r)a=38	kV(r)a=57	kV(r)a=75	kV(r)a=94	kV(r)a=113	kV(r)a=132	kV(r)a=151	kV(r)a=170
4	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
6	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
9	🟩 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
12	🟩 	🟩 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
18	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟥 	🟥 
36	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 

Table 6.7. Family of first Born approximation reconstructions for a square with sides of 120mm (2λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 1.1 to 1.9. There are 360 receivers, a resolution of 250x250 on each image with increasing number of sources going down the page for each permittivity value. Calculated minimum degrees of freedom (number of sources) and weak scattering metric are shown for reference.

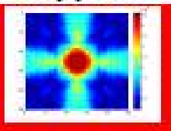
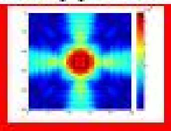
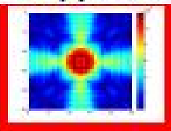
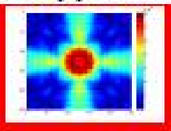
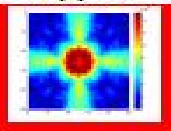
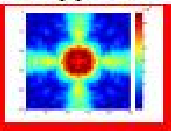
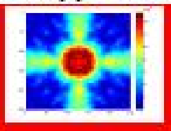
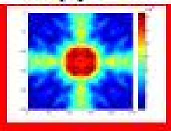
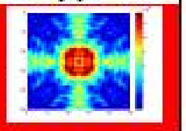
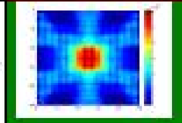
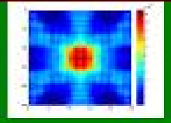
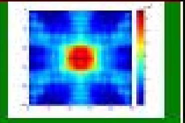
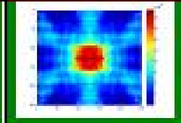
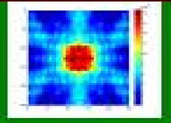
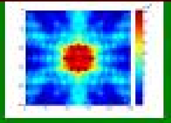
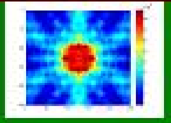
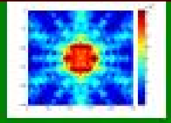
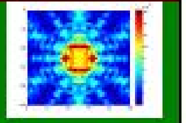
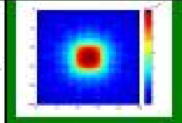
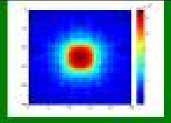
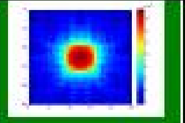
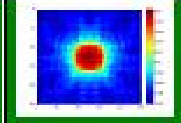
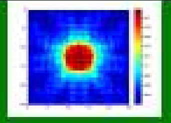
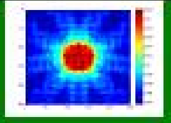
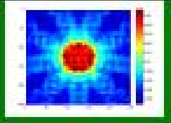
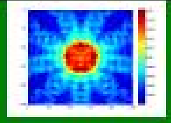
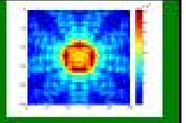
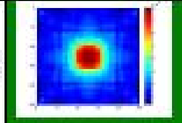
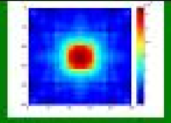
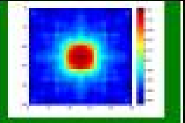
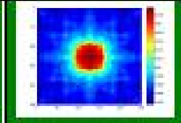
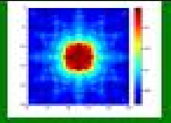
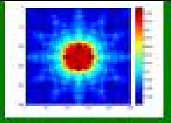
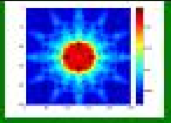
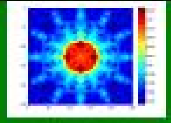
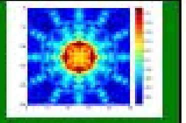
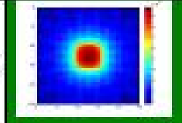
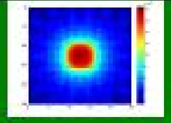
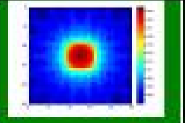
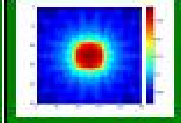
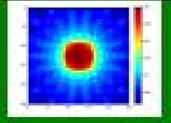
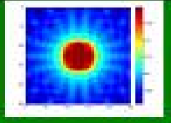
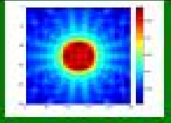
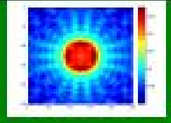
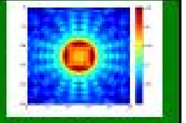
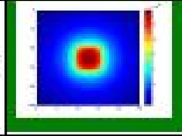
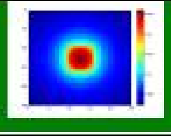
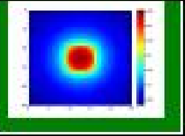
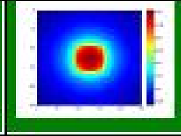
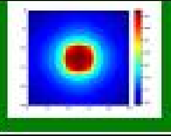
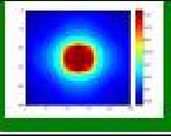
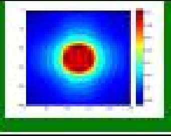
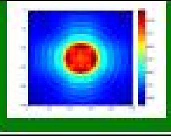
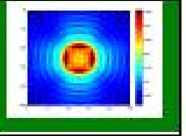
	er=1.1	er=1.2	er=1.3	er=1.4	er=1.5	er=1.6	er=1.7	er=1.8	er=1.9
	N=4.2	N=4.38	N=4.56	N=4.73	N=4.90	N=5.06	N=5.22	N=5.37	N=5.51
	kV(r)a=0.6	kV(r)a=1.3	kV(r)a=1.9	kV(r)a=2.5	kV(r)a=3.1	kV(r)a=3.8	kV(r)a=4.4	kV(r)a=5.0	kV(r)a=5.7
4									
6									
9									
12									
18									
36									

Table 6.8. Family of first Born approximation reconstructions for a pair of squares with sides of 120mm (2λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 1.1 to 1.9. There are 360 receivers, a resolution of 250x250 on each image with increasing number of sources going down the page for each permittivity value. Calculated minimum degrees of freedom (number of sources) and weak scattering metric are shown for reference.

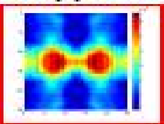
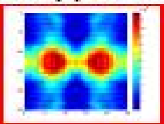
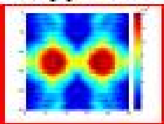
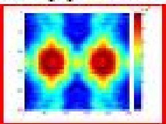
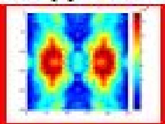
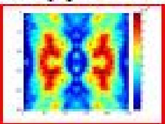
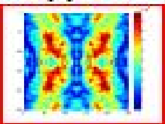
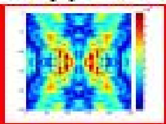
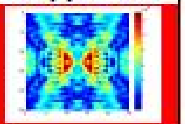
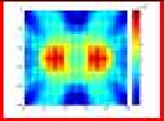
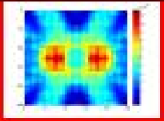
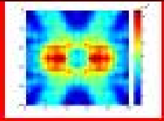
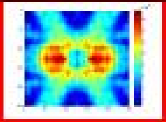
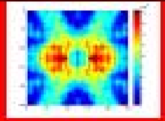
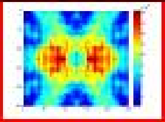
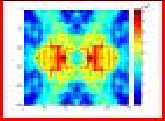
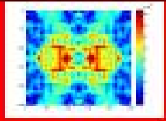
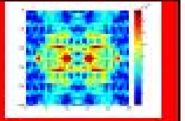
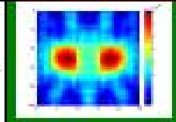
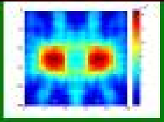
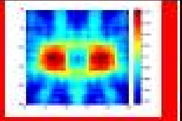
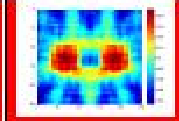
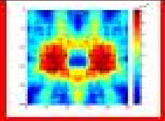
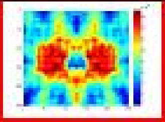
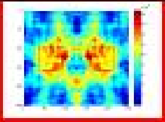
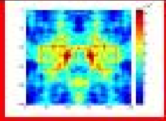
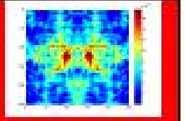
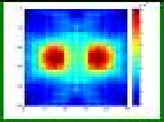
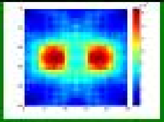
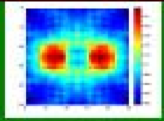
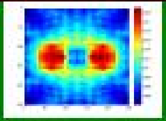
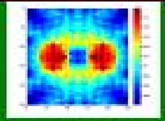
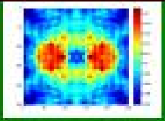
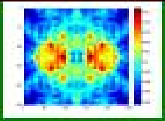
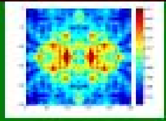
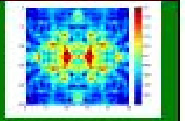
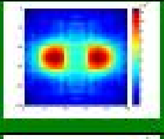
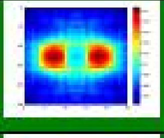
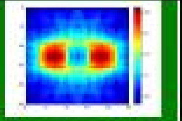
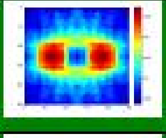
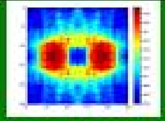
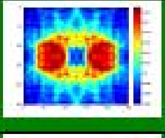
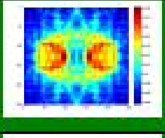
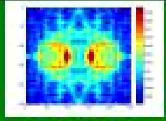
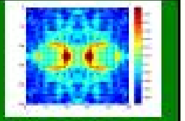
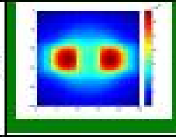
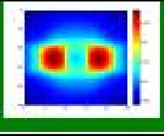
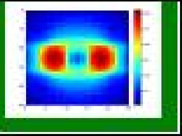
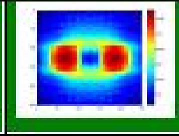
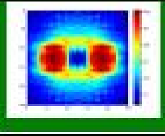
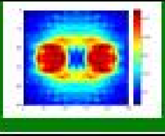
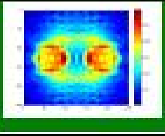
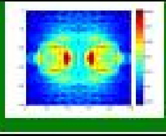
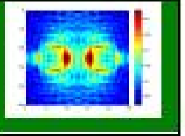
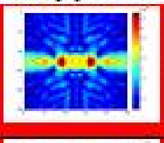
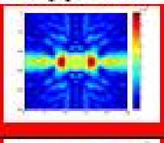
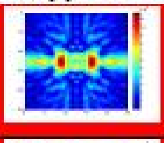
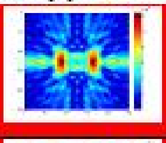
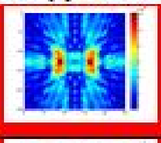
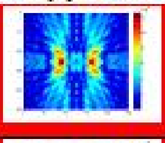
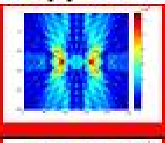
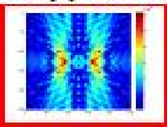
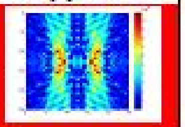
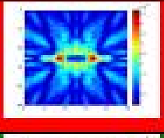
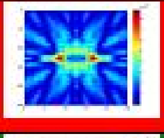
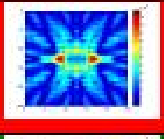
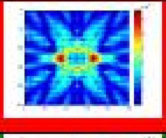
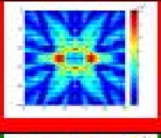
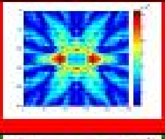
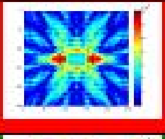
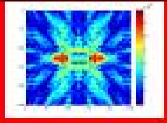
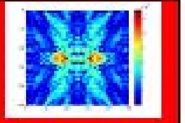
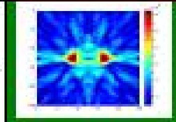
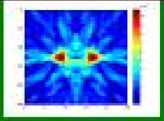
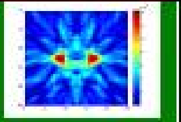
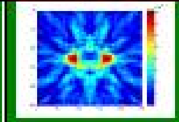
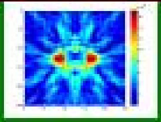
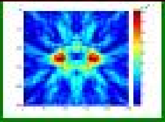
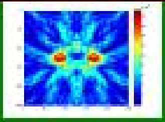
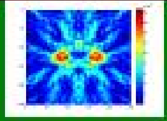
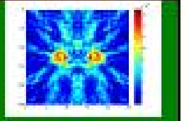
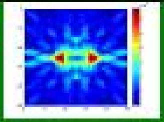
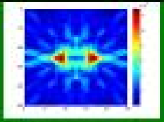
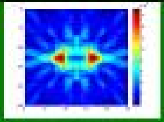
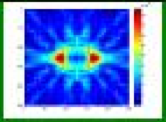
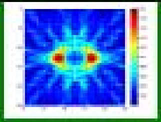
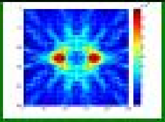
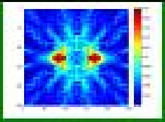
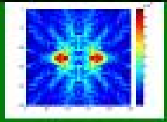
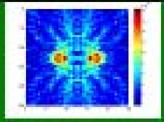
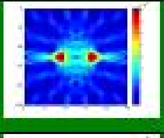
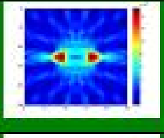
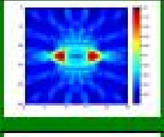
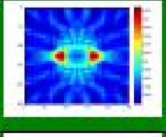
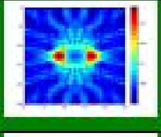
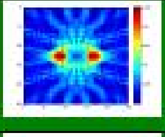
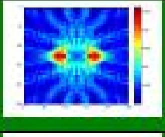
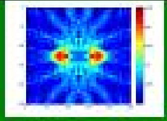
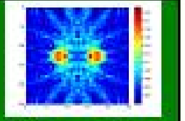
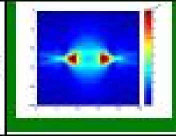
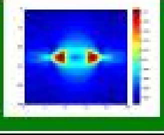
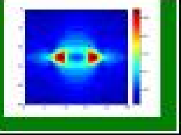
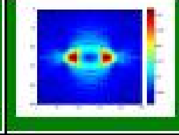
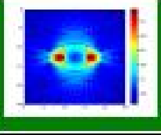
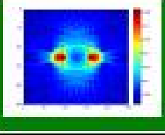
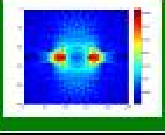
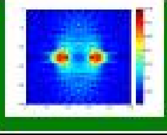
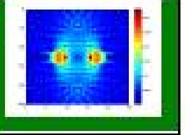
	er=1.1	er=1.2	er=1.3	er=1.4	er=1.5	er=1.6	er=1.7	er=1.8	er=1.9
	N=8.39	N=8.76	N=9.12	N=9.47	N=9.80	N=10.12	N=10.43	N=10.73	N=11.03
	kV(r)a=3.8	kV(r)a=7.5	kV(r)a=11.3	kV(r)a=15.1	kV(r)a=18.9	kV(r)a=22.6	kV(r)a=26.4	kV(r)a=30.2	kV(r)a=33.4
4	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
6	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
9	🟩 	🟩 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
12	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
18	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
36	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 

Table 6.9. Family of first Born approximation reconstructions for a pair of triangles with a base of 120mm (2λ) and a height of 180mm (3λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 1.1 to 1.9. There are 360 receivers, a resolution of 250x250 on each image with increasing number of sources going down the page for each permittivity value. Calculated minimum degrees of freedom (number of sources) and weak scattering metric are shown for reference.

	er=1.1	er=1.2	er=1.3	er=1.4	er=1.5	er=1.6	er=1.7	er=1.8	er=1.9
	N=6.29	N=6.57	N=6.84	N=7.10	N=7.35	N=7.59	N=7.82	N=8.05	N=8.27
	kV(r)a=7.5	kV(r)a=15.1	kV(r)a=22.6	kV(r)a=30.2	kV(r)a=37.7	kV(r)a=45.2	kV(r)a=52.8	kV(r)a=60.3	kV(r)a=67.9
4	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
6	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
9	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
12	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
18	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
36	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 

6.2 Analysis of Requirements for Degrees of Freedom for Receivers

The requirement that the minimum number of degrees of freedom is achieved is applicable to the number of sources and also the number of receivers. This being the case, it seemed appropriate to conduct experiments on the number of receivers much like those done in Section 6.1 for the sources. This was not as simple as it might seem. It was observed that as one continues to reduce the number of receivers, and keep the receivers equally spaced and the same distance from the target, the natural consequence is that the spacing between the receivers is ever increasing. Eventually, as is already known in 1D signal processing [42], this will eventually lead to aliasing issues in the resulting reconstructed image. This is exactly what happened in this case. This was a very significant issue especially dealing with receiver numbers in the 5 to 20 range for the specific examples investigated here. This issue prevented us from following the direct approach used in Section 6.1 for the sources. Before this aspect of the requirements of minimum degrees of freedom could be sufficiently examined, the issue of aliasing had to be addressed. One approach utilized in 1D signal processing to address aliasing in under-sampled signals is to randomly space the samples in lieu of using equally spaced samples [42]. This is a common technique that when employed eliminates the effects of aliasing, and as long as the original signal strength or signal-to-noise ratio is high enough, the original signal will be evident in the presence of the spatial noise introduced by reducing and randomizing the receiver locations. This process is illustrated in Figure 6.1 below taken from [42]. This same method is the technique that was implemented in 2 dimensions in the current algorithm code in an attempt to disperse the aliasing as done in 1 dimension. This will of course introduce some element of noise to the reconstructed images, but the signal-to-noise ratio should be sufficient to produce a useful image for examining the effects of degrees of freedom on the number of receivers.

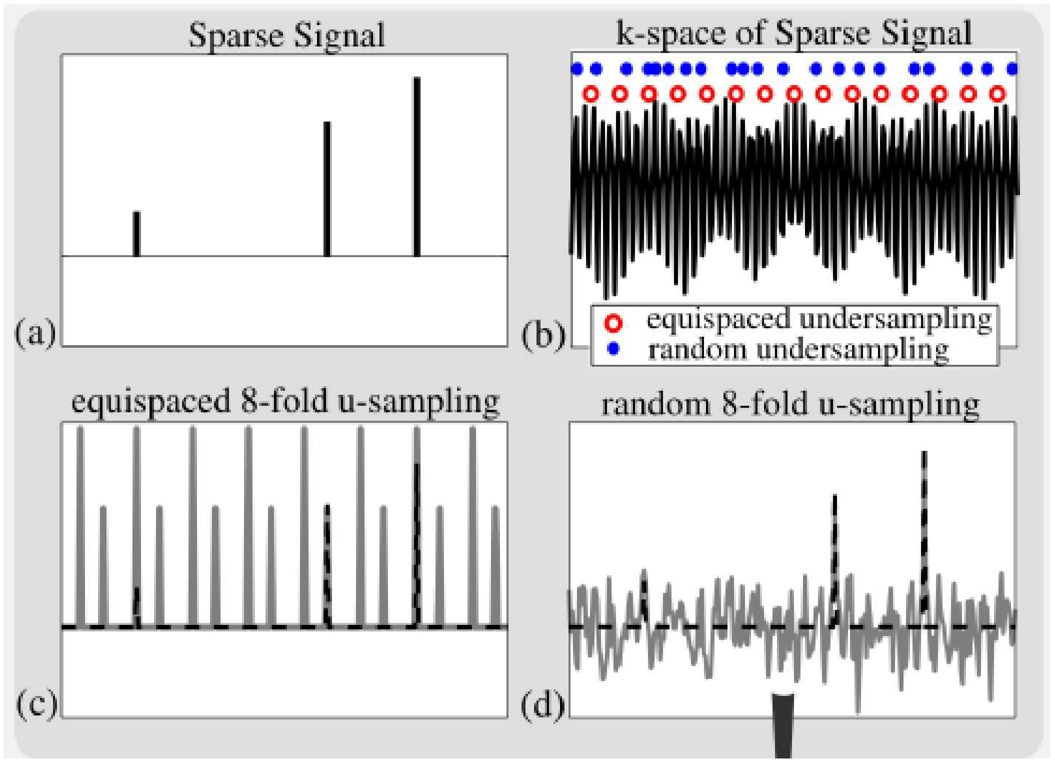


Figure 6.1. (a) Sparse signal to be sampled. (b) The k space representation of original signal in (a) with equispaced undersampling and random undersampling sample locations shown. (c) Resulting reconstruction of original sampled signal using equispaced samples. (d) Resulting reconstruction of original sampled signal using random samples.

Since the total number of receivers used in this model is 360, any multiple of 360 can be used for the number of receivers and still maintain the equal spacing of the receivers. This being the case, the code was modified such that for any undersampled quantity of receivers used, i.e., less than 360, the user has the ability to use the random generator in MATLAB to have the receiver at any location between the chosen locations to be selected in a Gaussian distributed manner centered around the original receiver location. This in a sense creates, for lack of a better term, what we define here to be a

Noise Bandwidth (NBW) around the chosen receiver location. For example, if 36 receivers are chosen, then the maximum noise bandwidth would be 360/36=10. This means that the receiver location could be any location within +/- 5 degrees around the original receiver location. Naturally, as fewer receivers are used, the NBW can increase if desired.

Since the consequences of doing this will be a function of frequency, this approach was initially tested for 3 different frequencies to observe the effects. The three incident frequencies used are 8GHz, 5GHz, and 2GHz. The simplest case was modeled first, that being a circle or cylinder with a radius of λ respective of the incident frequency. In conjunction with testing this possible remedy for the aliasing, the degree of freedom criteria was examined as well. For each case in this series of tests the minimum number of degrees of freedom is calculated to be $\pi(n)$ and for these tests n=1.225 so this results in N= $\pi(1.225)$=3.85. The results for these series of tests are shown in Table 6.10 and it is evident from these images that the random spacing technique is effective.

With the aliasing issue now addressed to some extent, the degrees of freedom experiments were then run for the targets comprised of 2 circles, 1 square, 2 squares, and 2 triangles as before for the tests with sources. These images are located in Tables 6.11 through 6.14. As already mentioned, though the aliasing is gone, there is now a noise element added to the images from the randomized detector spacing. This being the case, the effect of the degrees of freedom on the reconstructed images is not as pronounced as it was for the experiments involving the sources, but there is evidence in the images that suggests that the requirement of minimum degrees of freedom on the number of receivers is valid as well. This complements the series of experiments performed in Section 6.1.

Table 6.10. Family of first Born approximation reconstructions for a cylinder with a radius of 1λ illuminated by various incident frequencies. The target permittivity is 1.5. There are 360 receivers, a resolution of 250x250 on each image with increasing number of receivers going down the page. Green indicates degrees of freedom satisfied. Red indicated that the number of degrees of freedom is insufficient. Yellow indicates that the number of available degrees of freedom is marginal.

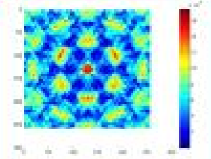
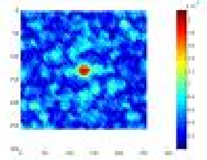
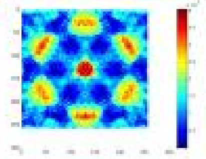
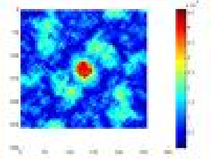
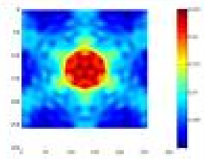
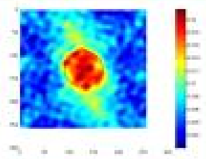
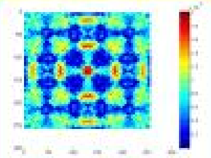
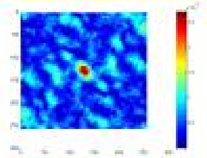
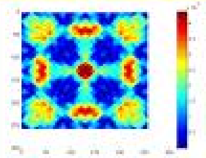
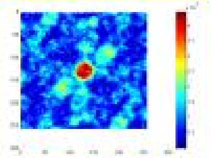
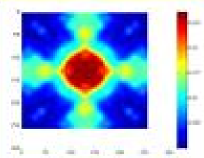
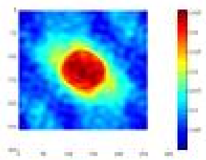
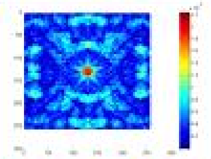
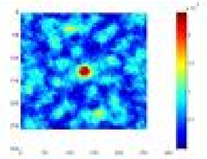
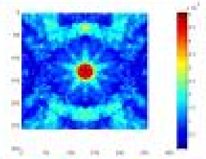
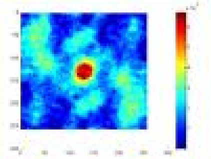
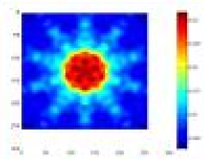
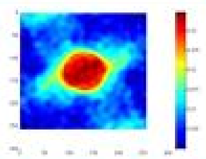
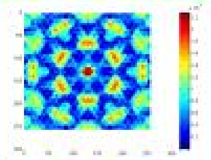
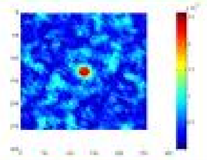
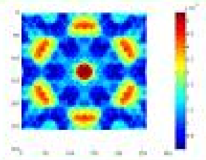
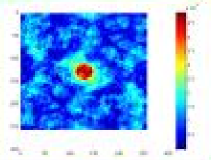
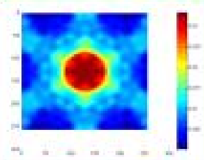
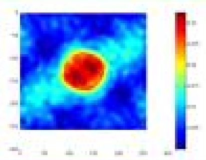
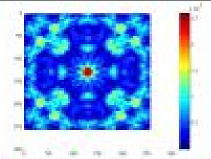
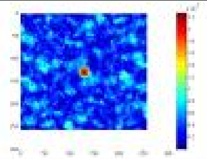
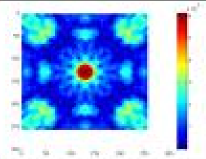
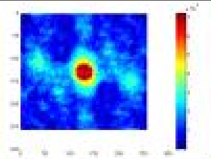
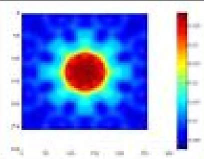
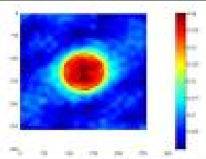
Rec#	8GHz NBW=0	8GHz NBW=360/Rec#	5GHz NBW=0	5GHz NBW=360/Rec#	2GHz NBW=0	2GHz NBW=360/Rec#
3	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
4	🟨 	🟨 	🟨 	🟨 	🟨 	🟨 
5	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
6	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
8	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 

Table 6.11: Family of first Born approximation reconstructions for a pair of cylinders with a radius of 60mm (1λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 1.1 to 1.9. There are 360 sources, a resolution of 250x250 on each image with increasing number of receivers going down the page for each permittivity value. Calculated minimum degrees of freedom (number of receivers) are shown for reference.

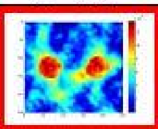
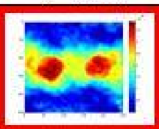
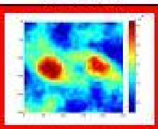
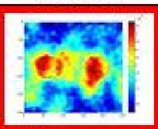
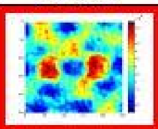
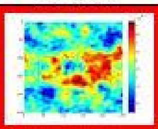
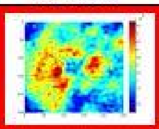
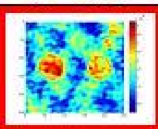
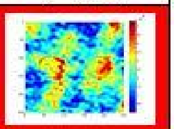
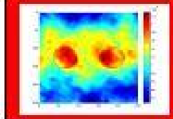
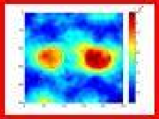
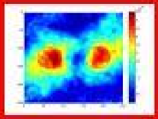
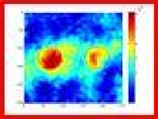
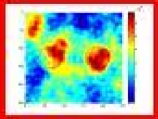
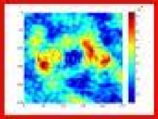
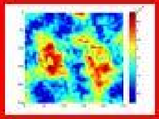
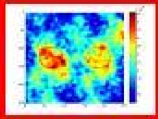
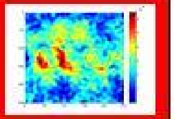
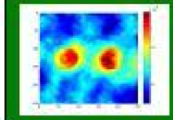
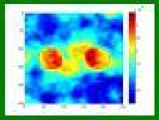
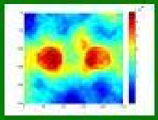
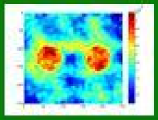
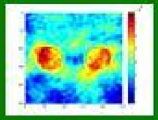
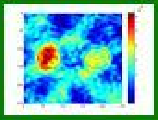
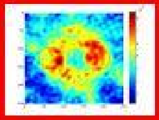
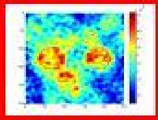
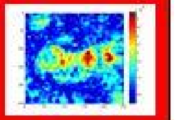
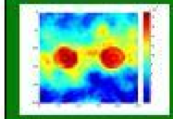
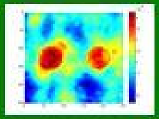
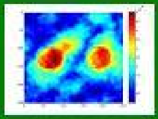
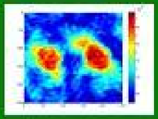
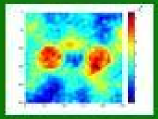
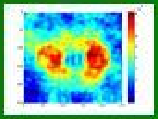
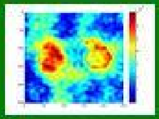
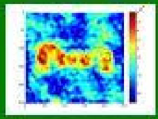
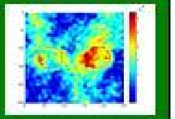
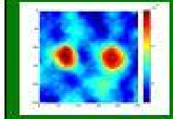
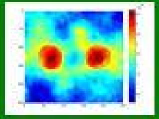
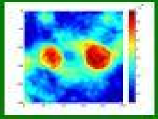
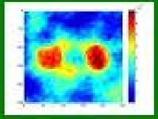
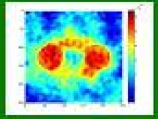
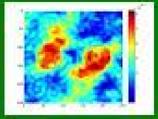
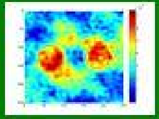
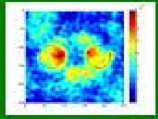
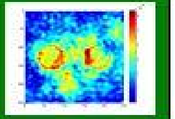
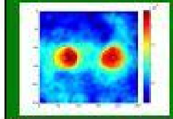
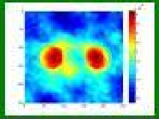
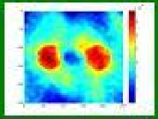
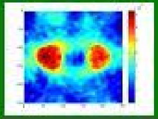
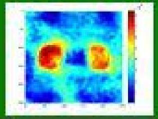
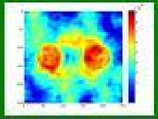
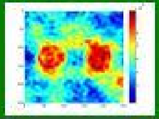
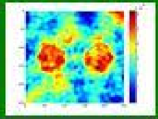
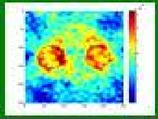
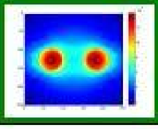
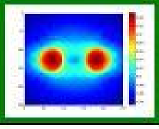
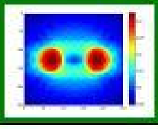
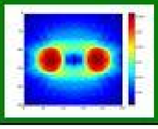
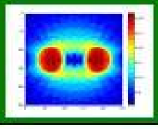
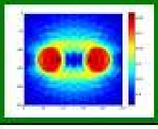
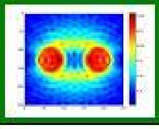
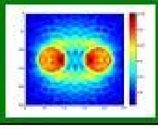
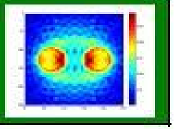
	er=1.1	er=1.2	er=1.3	er=1.4	er=1.5	er=1.6	er=1.7	er=1.8	er=1.9
	N=6.59	N=6.88	N=7.16	N=7.43	N=7.70	N=7.95	N=8.19	N=8.43	N=8.66
5	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
6	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
8	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟥 	🟥 	🟥 
9	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
10	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
12	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
360	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 

Table 6.12: Family of first Born approximation reconstructions for one square with a sides of 120mm (2λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 1.1 to 1.9. There are 360 sources, a resolution of 250x250 on each image with increasing number of receivers going down the page for each permittivity value. Calculated minimum degrees of freedom (number of receivers) are shown for reference.

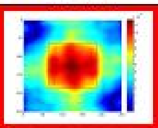
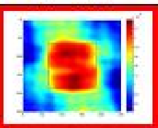
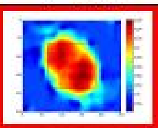
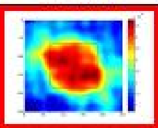
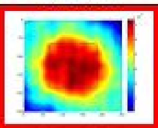
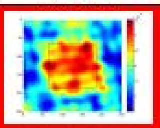
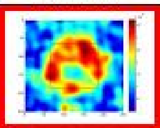
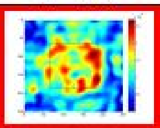
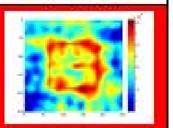
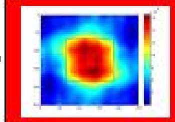
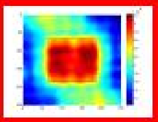
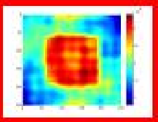
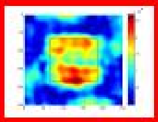
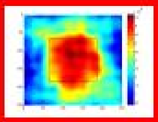
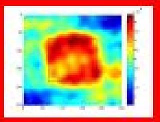
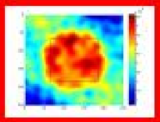
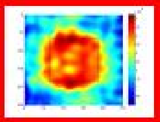
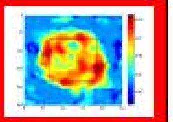
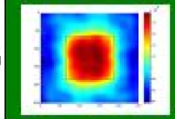
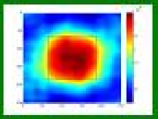
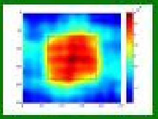
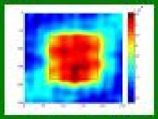
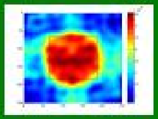
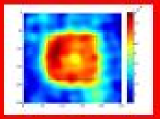
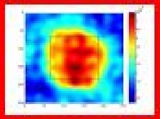
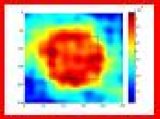
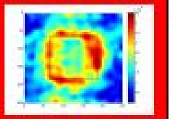
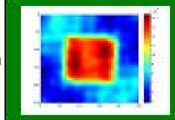
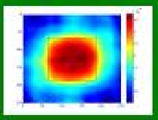
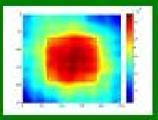
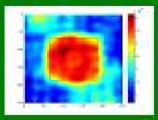
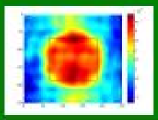
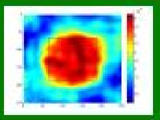
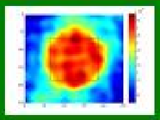
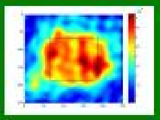
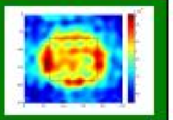
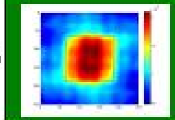
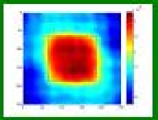
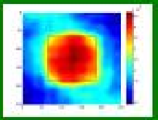
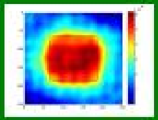
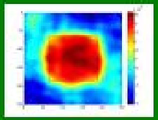
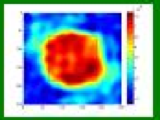
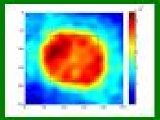
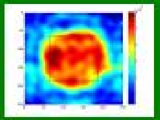
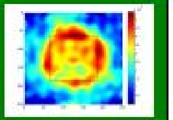
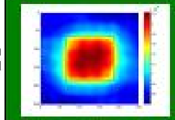
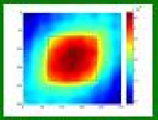
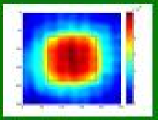
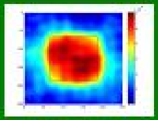
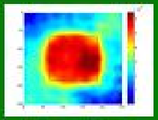
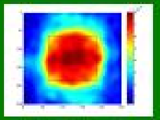
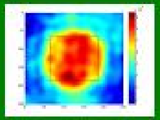
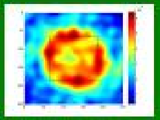
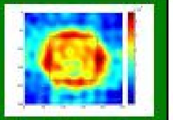
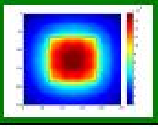
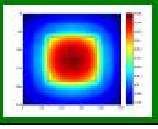
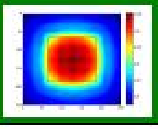
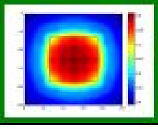
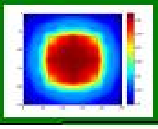
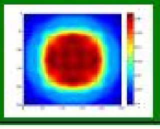
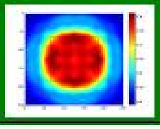
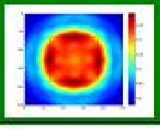
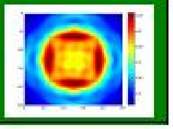
	er=1.1	er=1.2	er=1.3	er=1.4	er=1.5	er=1.6	er=1.7	er=1.8	er=1.9
	N=4.2	N=4.38	N=4.56	N=4.73	N=4.90	N=5.06	N=5.22	N=5.37	N=5.51
3	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
4	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
5	🟩 	🟩 	🟩 	🟩 	🟩 	🟥 	🟥 	🟥 	🟥 
6	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
8	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
10	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
360	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 

Table 6.13: Family of first Born approximation reconstructions for a pair of squares with sides of 120mm (2λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 1.1 to 1.9. There are 360 sources, a resolution of 250x250 on each image with increasing number of receivers going down the page for each permittivity value. Calculated minimum degrees of freedom (number of receivers) are shown for reference.

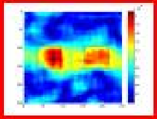
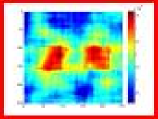
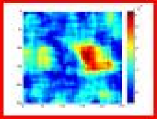
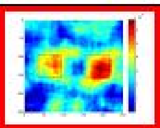
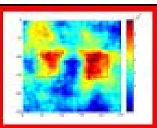
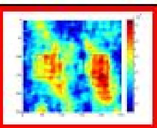
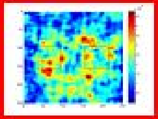
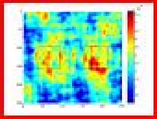
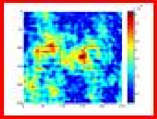
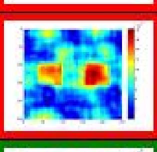
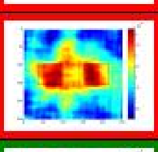
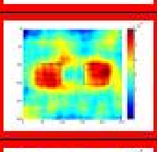
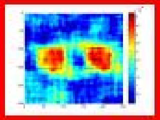
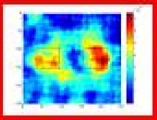
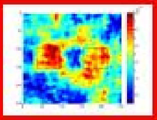
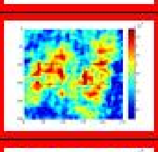
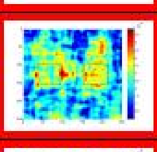
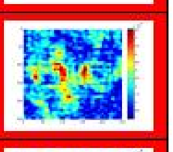
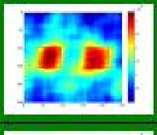
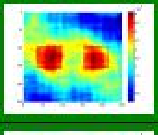
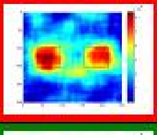
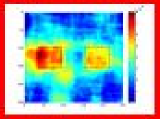
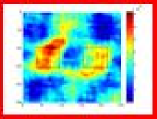
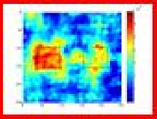
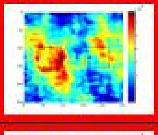
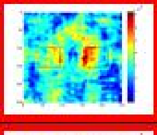
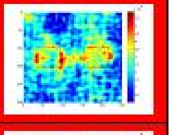
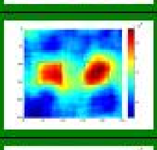
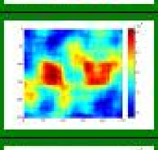
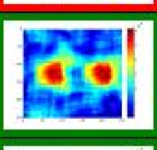
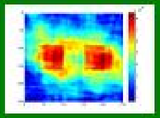
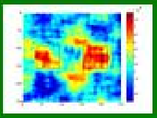
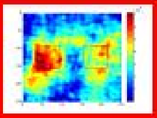
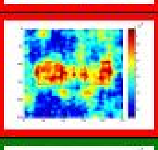
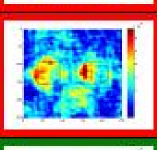
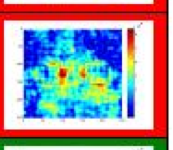
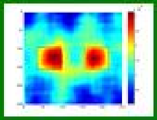
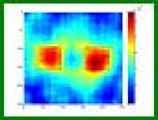
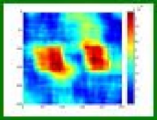
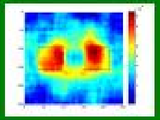
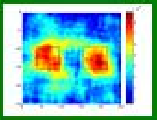
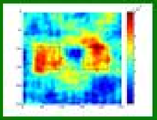
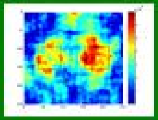
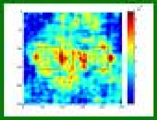
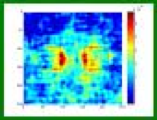
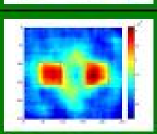
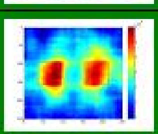
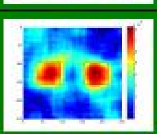
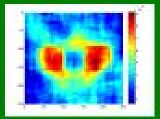
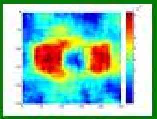
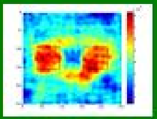
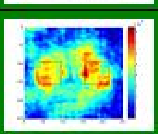
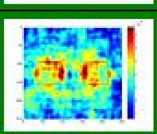
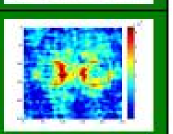
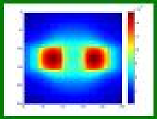
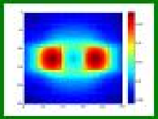
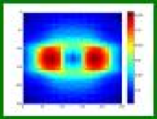
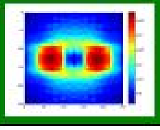
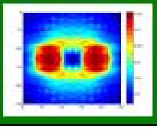
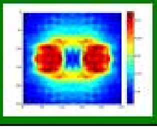
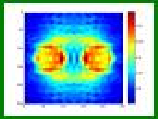
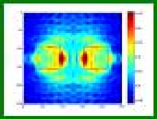
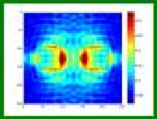
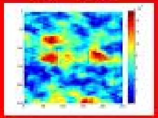
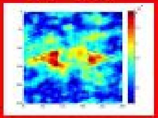
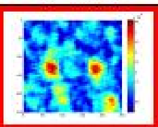
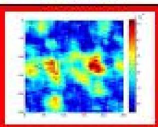
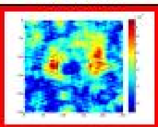
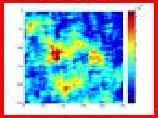
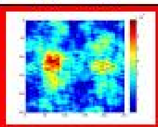
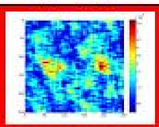
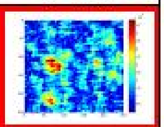
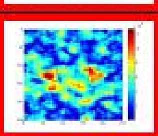
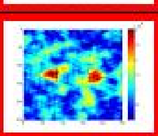
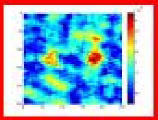
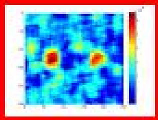
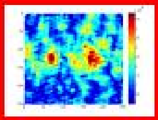
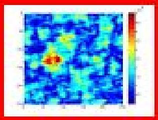
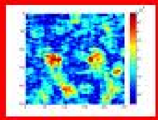
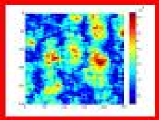
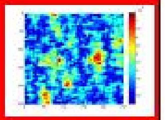
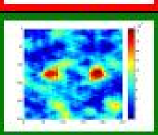
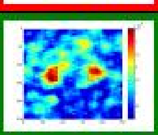
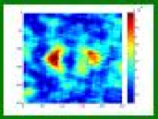
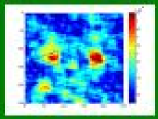
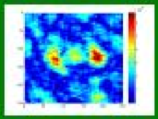
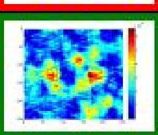
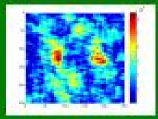
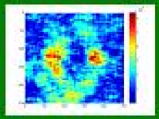
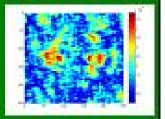
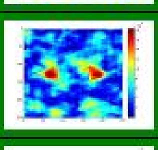
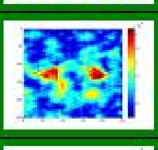
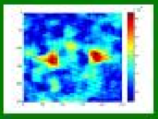
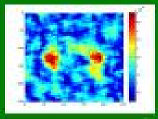
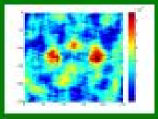
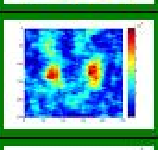
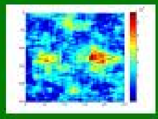
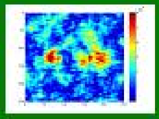
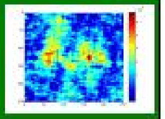
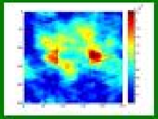
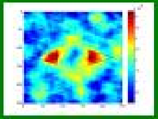
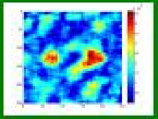
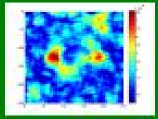
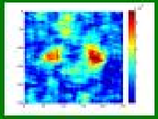
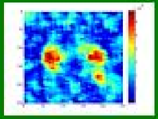
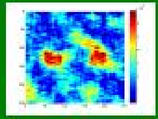
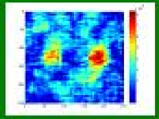
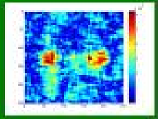
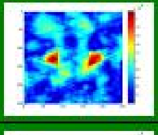
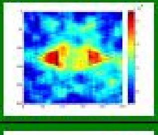
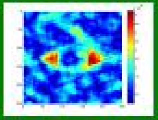
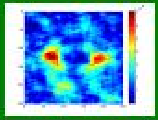
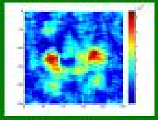
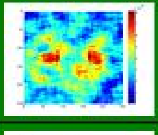
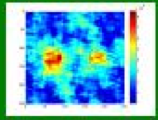
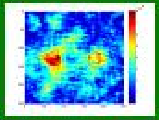
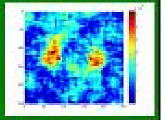
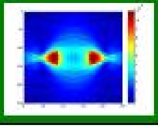
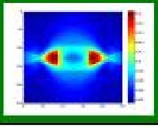
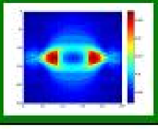
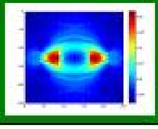
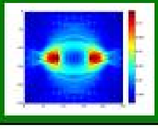
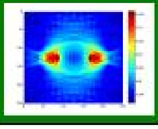
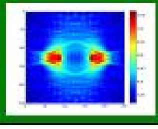
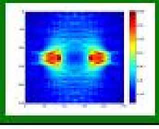
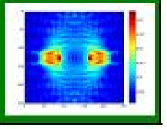
	er=1.1	er=1.2	er=1.3	er=1.4	er=1.5	er=1.6	er=1.7	er=1.8	er=1.9
	N=8.39	N=8.76	N=9.12	N=9.47	N=9.80	N=10.12	N=10.43	N=10.73	N=11.03
6	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
8	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
9	🟩 	🟩 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
10	🟩 	🟩 	🟩 	🟩 	🟩 	🟥 	🟥 	🟥 	🟥 
12	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
15	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
360	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 

Table 6.14: Family of first Born approximation reconstructions for a pair of triangles with heights of 180mm (3λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 1.1 to 1.9. There are 360 sources, a resolution of 250x250 on each image with increasing number of receivers going down the page for each permittivity value. Calculated minimum degrees of freedom (number of receivers) are shown for reference.

	er=1.1	er=1.2	er=1.3	er=1.4	er=1.5	er=1.6	er=1.7	er=1.8	er=1.9
	N=6.29	N=6.57	N=6.84	N=7.10	N=7.35	N=7.59	N=7.82	N=8.05	N=8.27
5	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
6	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 	🟥 
8	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
9	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
10	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
12	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 
360	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 	🟩 

6.3 Analysis of Relationship Between Born and Lorenz-Mie Q Factor

It is obvious in Tables 6.2, through 6.4, the figures for various cylinders, that there is a cyclic pattern to the quality of the Born reconstruction images with increasing scattering strength. Since it was demonstrated in the previous two sections that the minimum degrees of freedom have been satisfied, this pattern of behavior must have another explanation. In considering cylindrical targets, it is evident that this circle or cylinder can be analyzed as a 2 dimensional Lorenz-Mie scatterer as discussed in Section 2.3. This being the case, it would logically follow that as this cylinder is taken through the different scenarios of varying size and permittivity, that the incident wave inside the target should be cycling through various points of resonance. For Lorenz-Mie scatterers, this is characterized by the following "Q" factor equation which can be regarded as a measure of the scattering cross-section of the cylinder, and which increases at resonant frequencies:

$$Q = 2 - \frac{4}{p}\sin p + \frac{4}{p^2}\left(1 - \cos p\right) \tag{6.1}$$

where *p* in the above equation is defined as

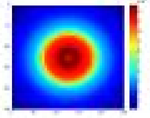
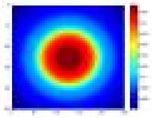
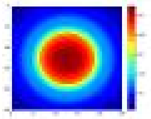
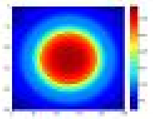
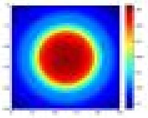
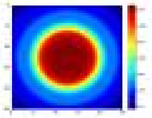
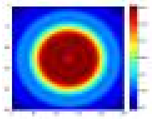
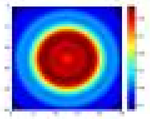
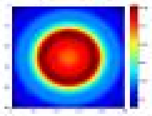
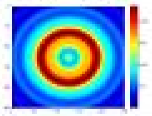
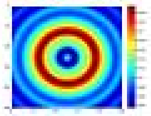
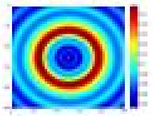
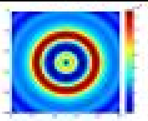
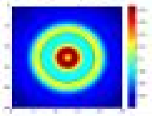
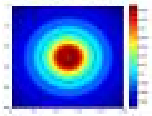
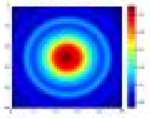
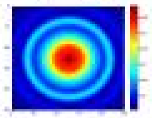
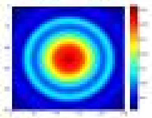
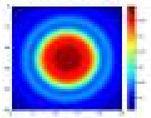
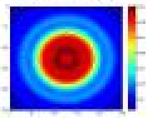
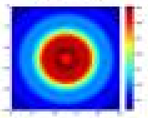
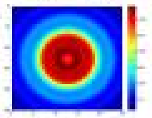
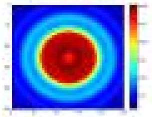
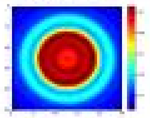
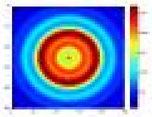
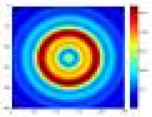
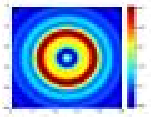
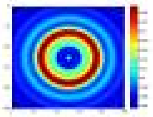
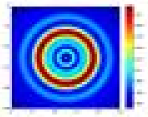
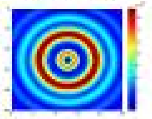
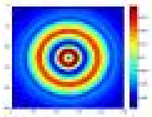
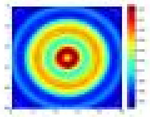
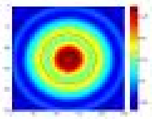
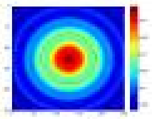
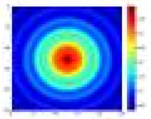
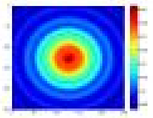
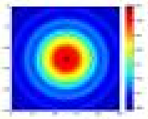
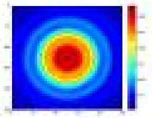
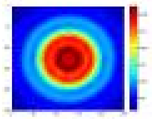
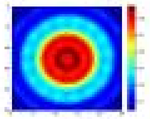
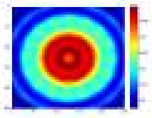
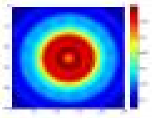
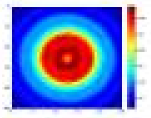
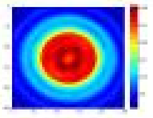
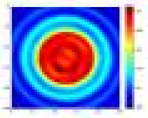
$$p = 4\pi r\left(n-1\right)/\lambda \tag{6.2}$$

These two equations are defined almost entirely in terms of physical parameters of the target. These parameters are the radius "r", the wavelength of the incident field "λ", and the index of refraction "*n*", which is also defined as

$$n = \sqrt{\varepsilon_r \mu_r} \tag{6.3}$$

where the relative permeability is generally equal to 1. The following tables show families of images that demonstrate the effects of Q on Born reconstructions.

Table 6.15. Family of first Born approximation reconstructions for a cylinder with a radius of 60mm (1λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 1.1 to 5.9. There are 360 receivers, a resolution of 250x250 on each image with increasing permittivity going from left to right and down the page. The permittivity is shown underneath each target.

							
er=1.1	er=1.2	er=1.3	er=1.4	er=1.5	er=1.6	er=1.7	er=1.8
							
er=1.9	er=2.1	er=2.2	er=2.3	er=2.4	er=2.5	er=2.6	er=2.7
							
er=2.8	er=2.9	er=3.1	er=3.2	er=3.3	er=3.4	er=3.5	er=3.6
							
er=3.7	er=3.8	er=3.9	er=4.1	er=4.2	er=4.3	er=4.4	er=4.5
							
er=4.6	er=4.7	er=4.8	er=4.9	er=5.1	er=5.2	er=5.3	er=5.4
							
er=5.5	er=5.6	er=5.7	er=5.8	er=5.9			

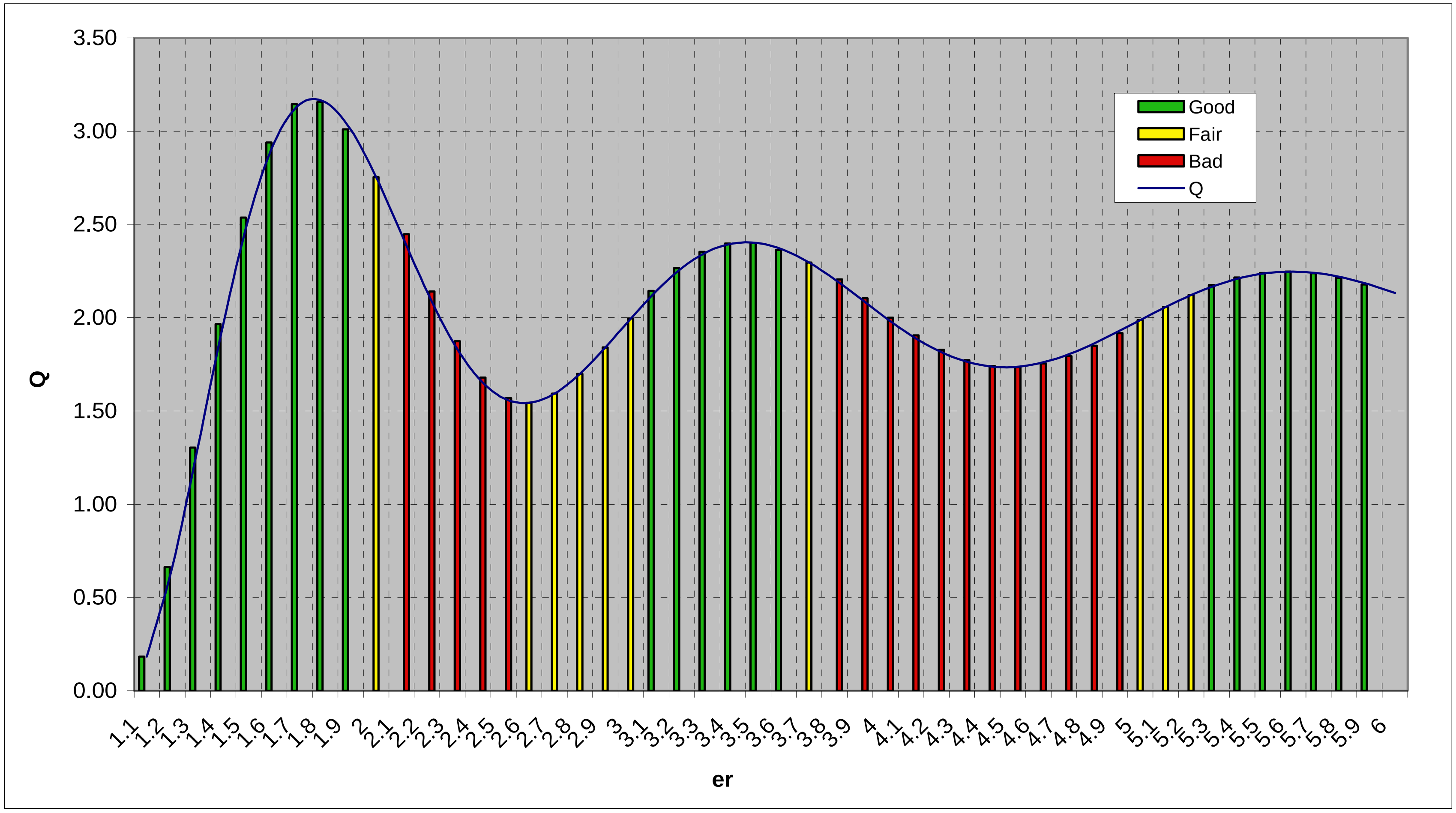
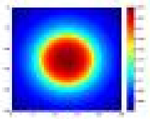
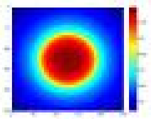
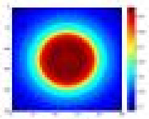
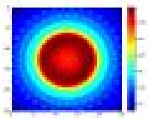
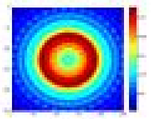
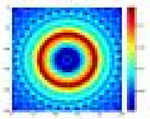
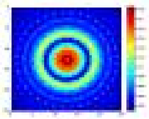
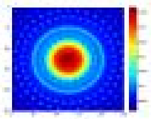
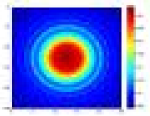
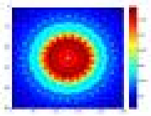
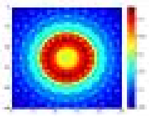
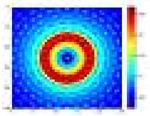
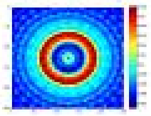
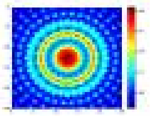
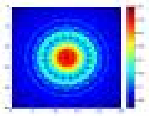
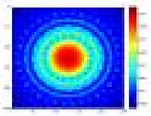
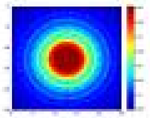
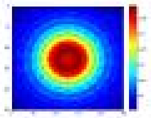
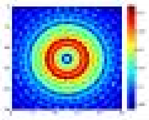
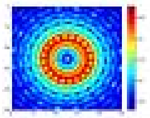
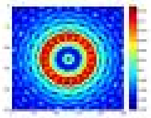
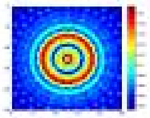
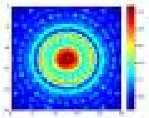
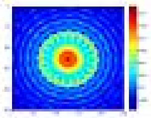
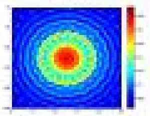
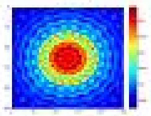
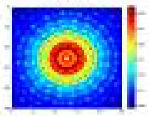
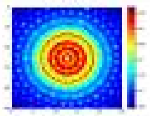
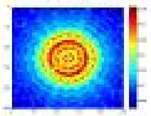
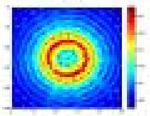
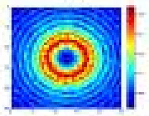
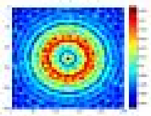
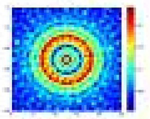
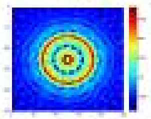
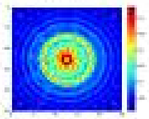
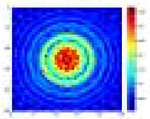
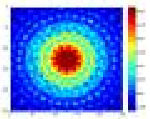
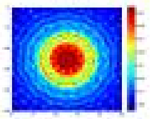
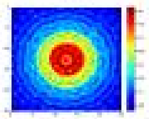
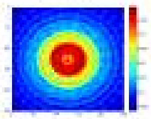
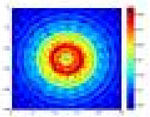
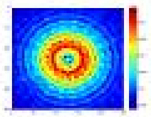
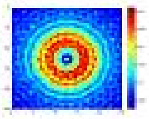
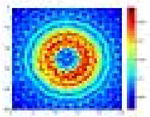
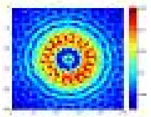


Figure 6.2. Graph showing the plot of the Lorenz-Mie scattering efficiency Q factor for the same conditions as the images in Table 6.15. Super-imposed on this graph is a column for each image which indicates the relative quality of the Born reconstruction with green indicating good, yellow indicating fair, and red indicating poor. This graph demonstrates that when Q is “high” or increasing the image reconstruction is good and when Q is “low” or decreasing then the image is poor.

Table 6.16. Family of first Born approximation reconstructions for a cylinder with a radius of 120mm (2λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 1.1 to 5.9. There are 360 receivers, a resolution of 250x250 on each image with increasing permittivity going from left to right and down the page. The permittivity is shown underneath each target.

							
1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8
							
1.9	2.1	2.2	2.3	2.4	2.5	2.6	2.7
							
2.8	2.9	3.1	3.2	3.3	3.4	3.5	3.6
							
3.7	3.8	3.9	4.1	4.2	4.3	4.4	4.5
							
4.6	4.7	4.8	4.9	5.1	5.2	5.3	5.4
							
5.5	5.6	5.7	5.8	5.9			

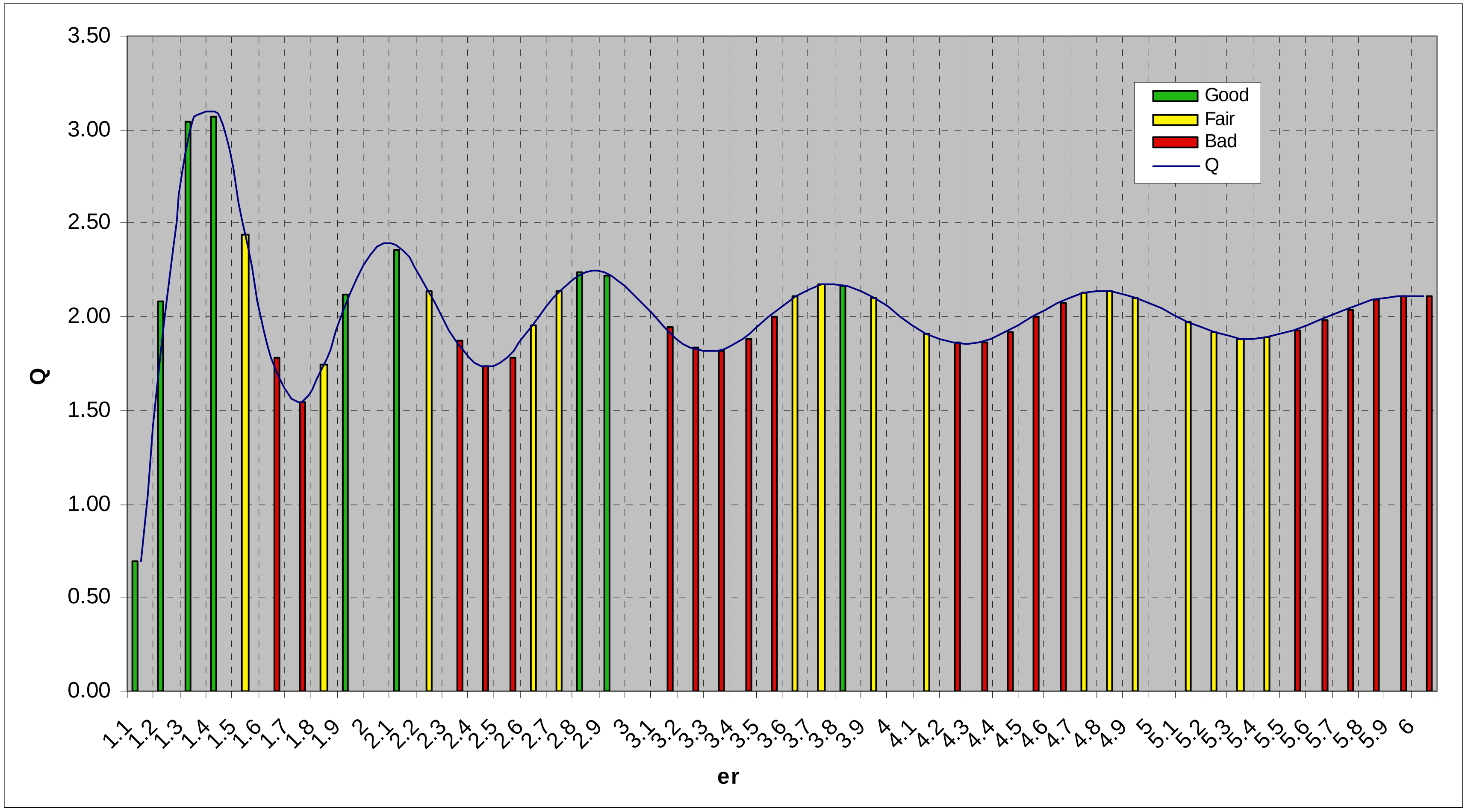
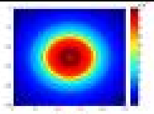
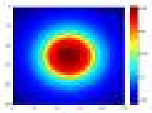
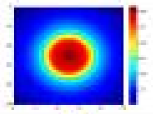
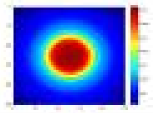
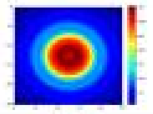
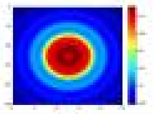
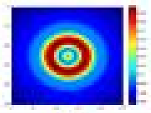
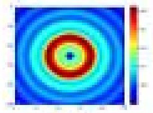
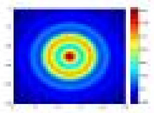
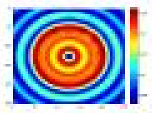
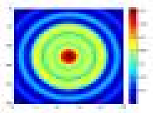
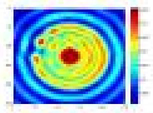
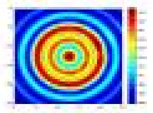
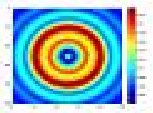
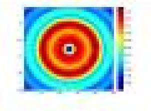
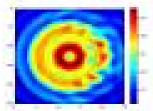
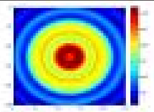
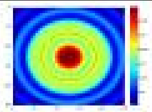
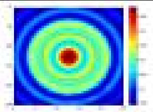
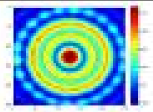
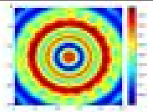
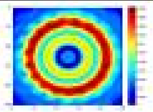
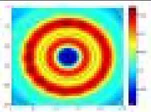
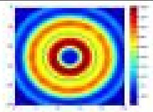
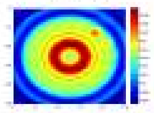
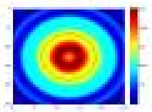
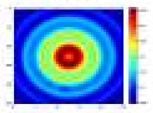
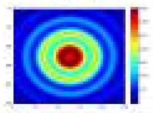
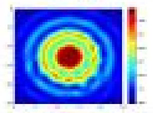
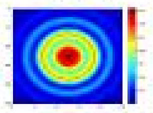
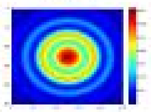
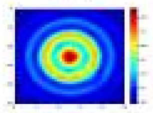
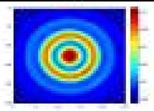
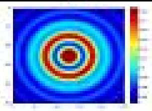
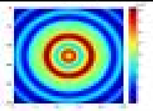
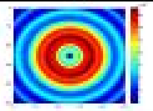
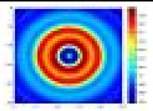
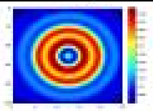
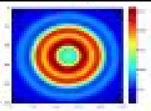
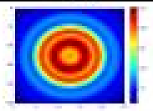
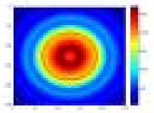
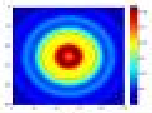
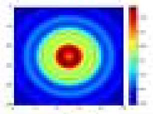
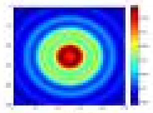
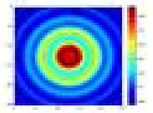


Figure 6.3. Graph showing the plot of the Lorenz-Mie scattering efficiency Q factor equation for the same conditions as the images in Table 6.16. Super-imposed on this graph is a column for each image which indicates the relative quality of the Born reconstruction with green indicating good, yellow indicating fair, and red indicating poor. This graph demonstrates that when Q is “high” or increasing the image reconstruction is good and when Q is “low” or decreasing then the image is poor.

Table 6.17. Family of first Born approximation reconstructions for a cylinder with a radius of 60mm (1λ) illuminated by a source with a frequency of 5GHz (λ=60mm) for a permittivity range of 0.9 to -3.9. There are 360 receivers, a resolution of 250x250 on each image with decreasing permittivity going from left to right and down the page. The permittivity is shown underneath each target.. The permeability for each image above is -1.

							
0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
							
0.1	-0.1	-0.2	-0.3	-0.4	-0.5	-0.6	-0.7
							
-0.8	-0.9	-1.1	-1.2	-1.3	-1.4	-1.5	-1.6
							
-1.7	-1.8	-1.9	-2.1	-2.2	-2.3	-2.4	-2.5
							
-2.6	-2.7	-2.8	-2.9	-3.1	-3.2	-3.3	-3.4
							
-3.5	-3.6	-3.7	-3.8	-3.9			

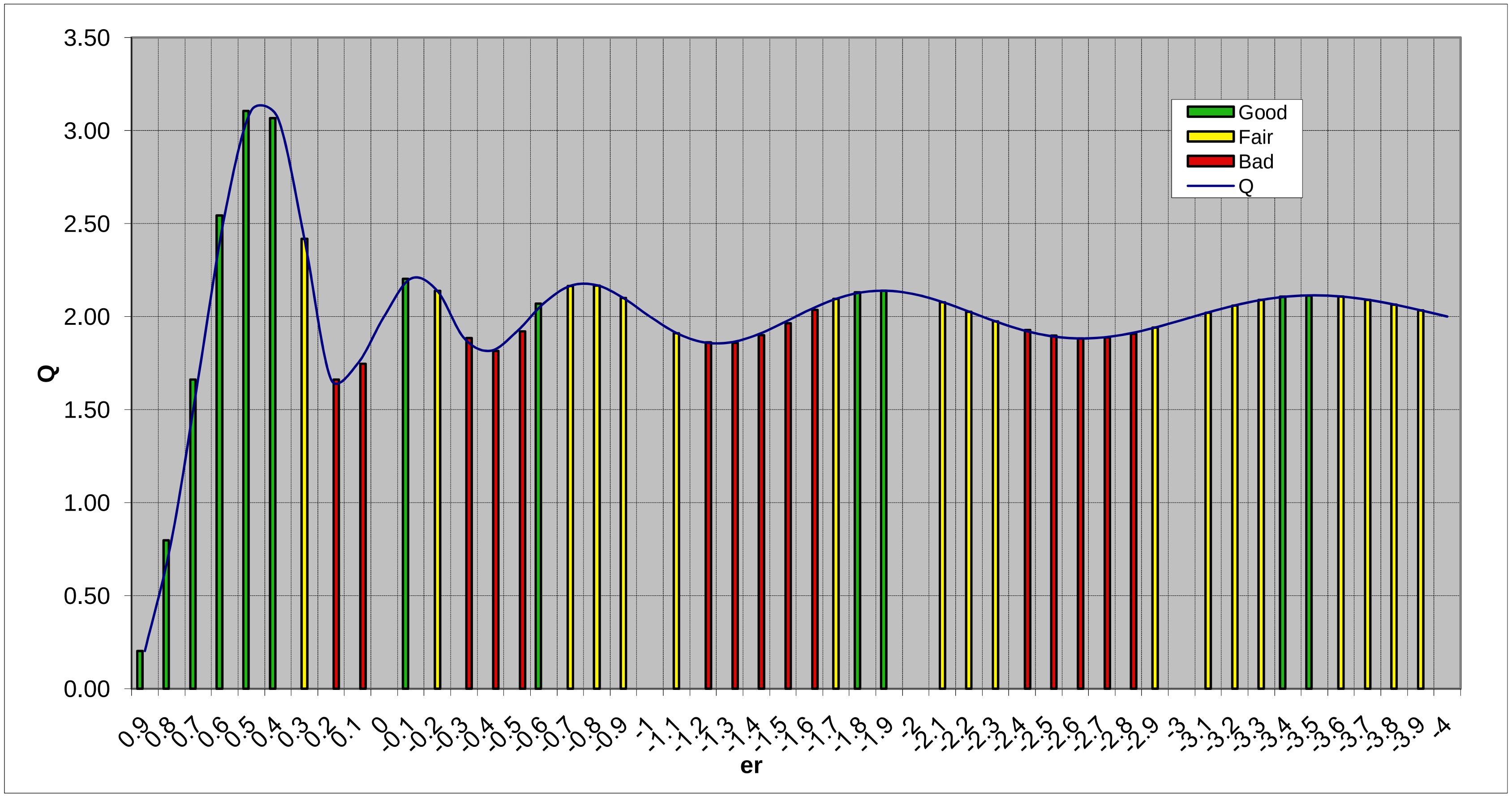


Figure 6.4. Graph showing the plot of the Lorenz-Mie scattering efficiency Q factor equation for the same conditions as the images in Table 6.17. Super-imposed on this graph is a column for each image which indicates the relative quality of the Born reconstruction with green indicating good, yellow indicating fair, and red indicating poor. This graph demonstrates that when Q is "high" or increasing the image reconstruction is good and when Q is "low" or decreasing the image is poor.

One very special case of the Lorenz-Mie model and the scattering cross section indicated by Q is that when the permittivity is negative. This is recently of special interest for work currently being done in the meta-material fields as researchers strive to produce materials with negative index of refraction. While this is a challenge in the physical realm, it can be simulated rather easily in the virtual realm. Table 6.17 shows the results for a cylinder as the permittivity is varied from 1 to -4. It should be noted that if the permittivity is allowed to go negative, this causes problems for the Q equation as "*p*" is now an imaginary number and as is well known this corresponds to a very lossy or evanescent wave case. Thus when the permittivity is negative and the permeability is positive, the Q relationship breaks down and one would expect no resonances are predicted. If this condition is simulated using the methods above, the reconstructed images for all of the values in the range of permittivities look like the image in Figure 6.5, which is very poor. To correctly model a negative index material, the permeability value is also set to -1 when the permittivity value is negative. This allows for the negative root to be selected for "*p*" which yields a real valued representation for Q. This gives consistent results with those of the simulation as seen in Figure 6.4.

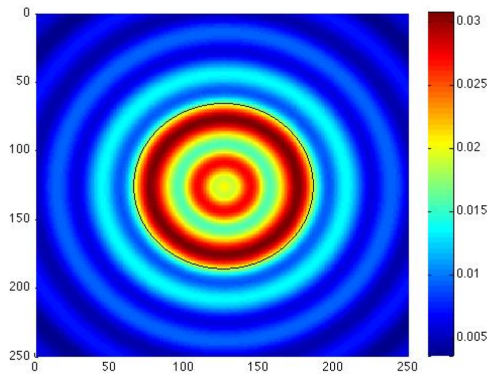


Figure 6.5. The resulting reconstructed image for a 2-dimensional cylinder with a positive value of 1 for the permeability and a varying negative values for the permittivity. All images look virtually the same regardless of the (negative) value for the permittivity.

6.4 Analysis of Effects of Independent Source Data Processing

It has been shown in previous work that improvements can be made to the reconstructed image produced by using the Born approximation method by performing filtering in the cepstrum domain [6]. Previously in applying this method, the filtering was applied to the V<Ψ> image data from the combined sources. In retrospect, while the expectation was that the field term might appear noise-like, a more rigorous approach is described here. The basis of the derivation in Section 3, in particular Eqs. 3.2 through 3.5, is that it is for a single incident wave or direction. Since this is the case, it seems logical that the processing of the data should better be carried out on a source by source basis and then recombined later in the process. When the independent source data is combined at the beginning, this gives the opportunity for the information relating to the unwanted Ψ terms to be combined and possibly modulated in such a way that parts of its spectrum may be even more difficult to locate and/or remove.

This being the case, the cepstrum filtering method in [6] was re-implemented on a source by source basis, in lieu of a combined source method. To accomplish this, the Ewald circle of data for each independent source was derived separately and the Born image for each source was computed as illustrated in Table 6.18 for two different target sets, a circle and two triangles. The data for each source was then transformed into the cepstrum domain and filtered accordingly as before in [6], again as illustrated in Table 6.19 for these two different targets sets. The question then arises at to the correct step in the reconstruction process to recombine the independent data into a cumulative data set from which the entire image could be recovered. There are two opportunities to do this. The first is in the cepstrum domain directly after filtering. The second is to transform each of the independent filtered source data back into the image domain and recombine there. Both of these scenarios were implemented and examined.

This total process was performed on 4 unique target sets to provide a broad comparison of the general performance of these methods. In Table 6.20, four target sets, with various permittivities were modeled and filtered as described above. In Table 6.20 it is obvious that all of the cepstrum filtered images show improvement over the image obtained using just the Born method, both in regards to image boundaries and noise reduction. In our analysis of these methods, it has been observed that as the permittivity of a target is increased, the apparent size of the target is artificially inflated in relation to the actual target size. This can be seen in the images in Table 6.20. It appears in the filtering processes where the sources are independently filtered that this size inflation is decreased. Moreover, it appears that the images from the process of having the independent source data combined in the cepstrum domain seems to perform slightly better. Additionally, the independent source filtered images exhibit a scaling issue as compared to the Born image and the cepstrum filtered Born images. The image with the data combined in cepstrum space has a magnitude or scale that decreases as the number of sources is increased, while conversely the image with the data combined in image space has a magnitude that increases as the number of sources is increased. It will be shown in a later section that a scaling factor, dependent on the number of sources, can be applied to adjust or improve these inappropriately scaled values. Taking this all into account, and analyzing the images in Table 6.20, it appears that in general, the method of processing the independent source data separately followed by combining these data sets in the cepstrum domain appears to perform better than the other methods examined.

Table 6.18. Sequence of images showing the individual source Ewald circles and the corresponding Born images for each source. The composite of the Ewald circles and the composite Born reconstructions are also shown for a circle and for two triangles.

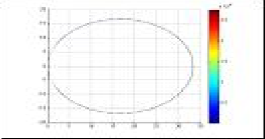
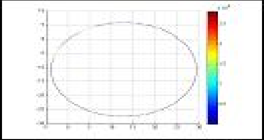
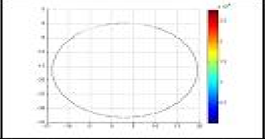
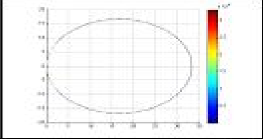
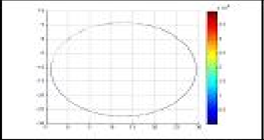
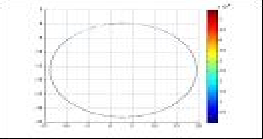
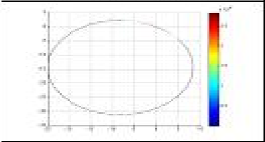
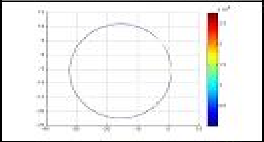
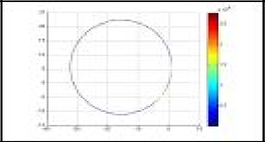
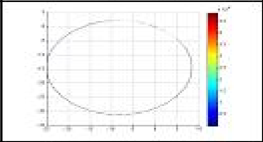
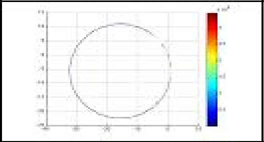
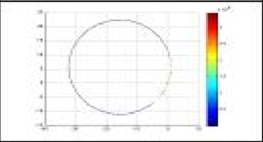
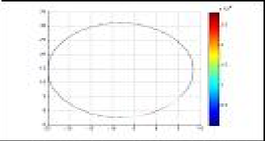
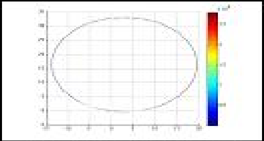
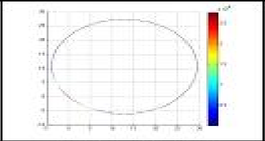
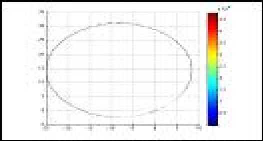
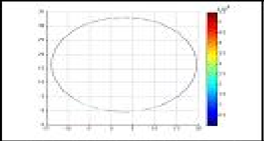
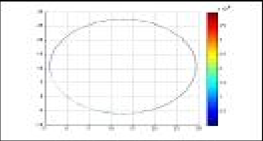
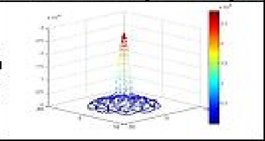
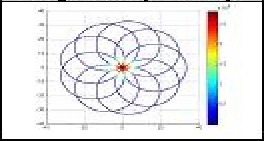
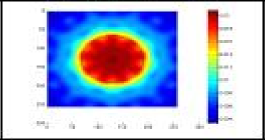
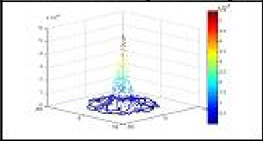
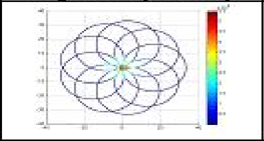
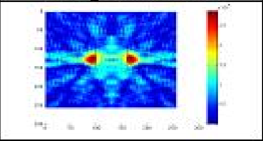
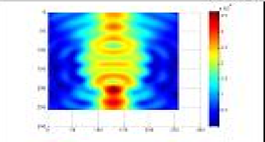
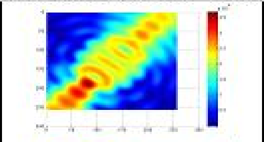
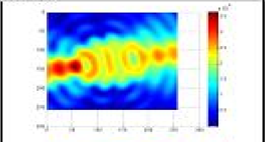
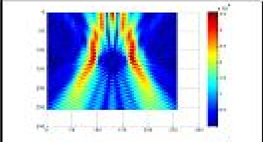
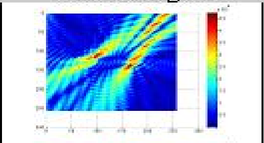
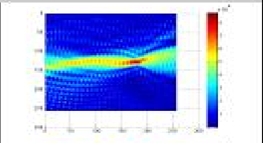
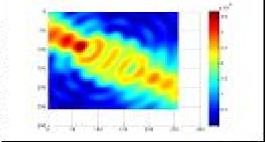
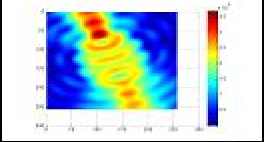
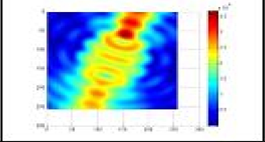
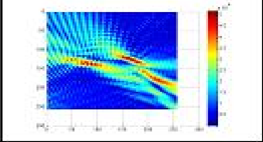
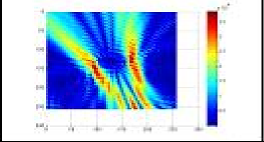
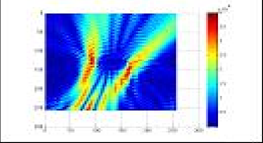
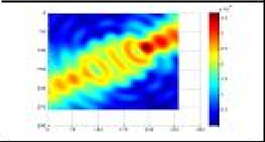
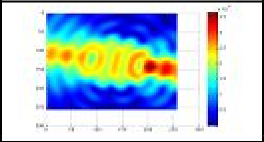
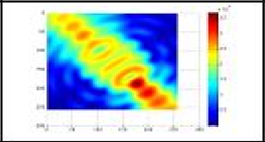
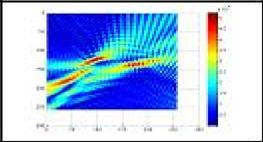
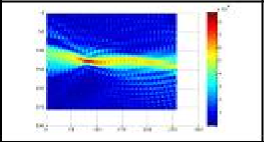
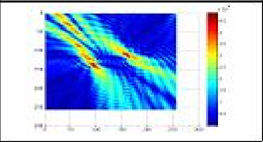
Ewald Circles for Each of the 9 Sources						
						
						
	Side View (Ewalds)	Top View (Ewalds)	Composite Born	Side View (Ewalds)	Top View (Ewalds)	Composite Born
Composite Images						
	One Circle with Radius = 1 lambda			Two Triangles		
Corresponding Born Image for Each Ewald Circle Above						
						
						

Table 6.19. Sequence of images showing the individual Born images for each source and the corresponding Cepstrum filtered image. The resulting Cepstrum composite images are also shown for the Born images, sources combined in Cepstrum space and Image Space.

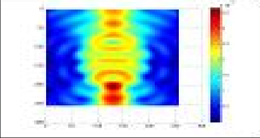
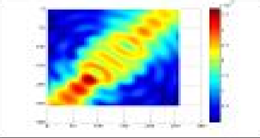
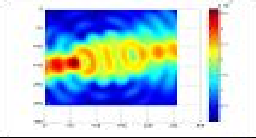
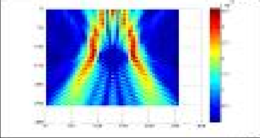
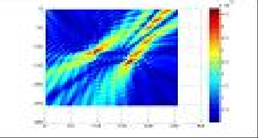
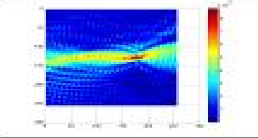
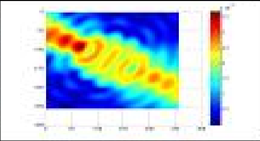
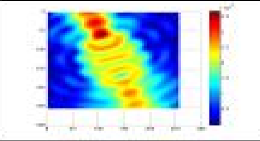
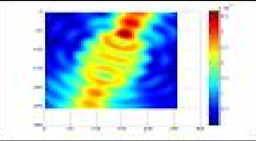
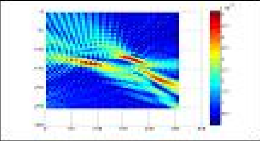
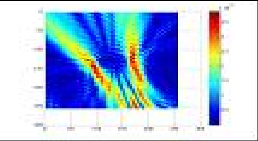
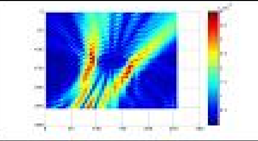
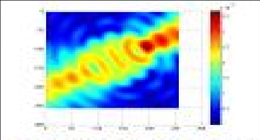
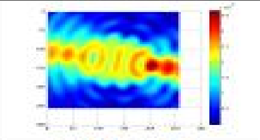
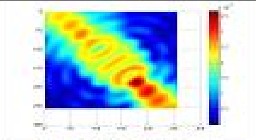
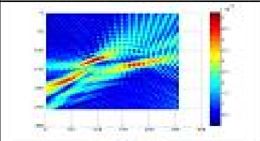
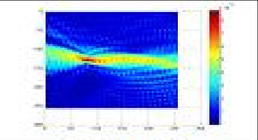
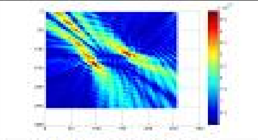
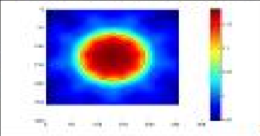
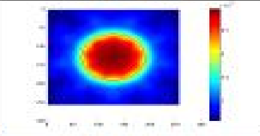
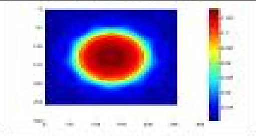
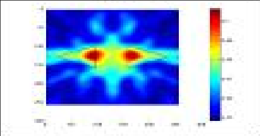
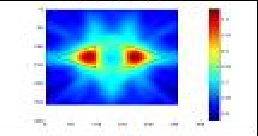
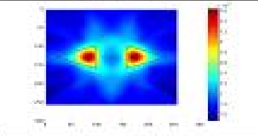
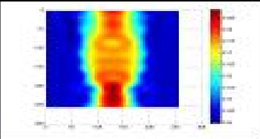
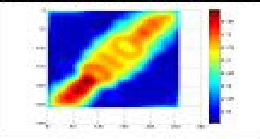
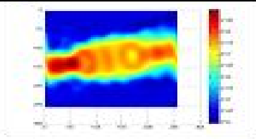
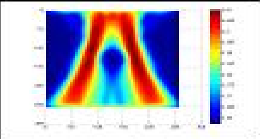
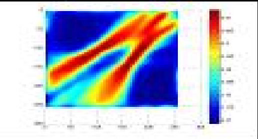
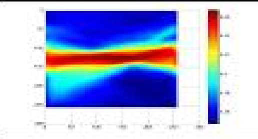
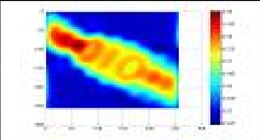
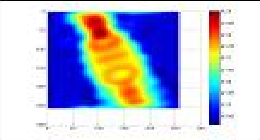
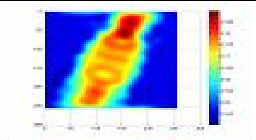
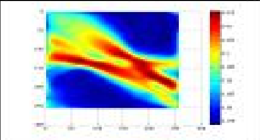
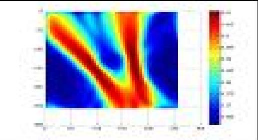
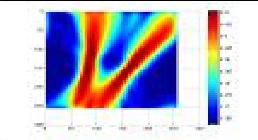
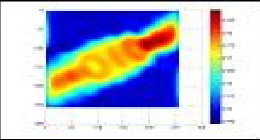
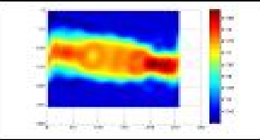
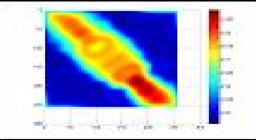
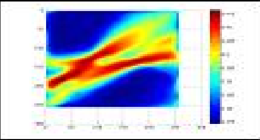
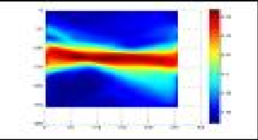
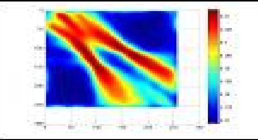
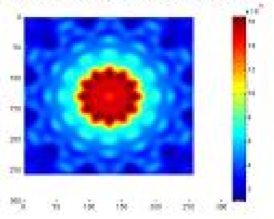
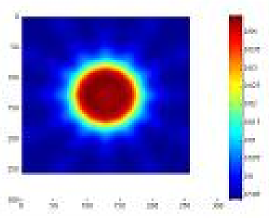
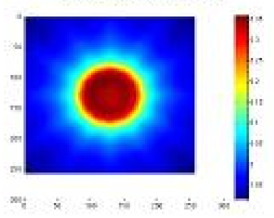
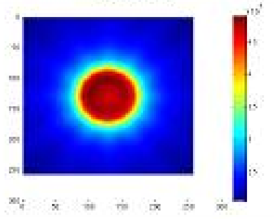
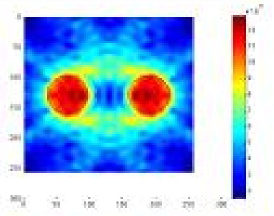
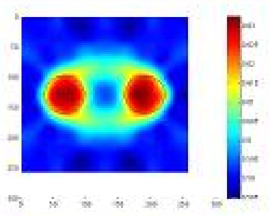
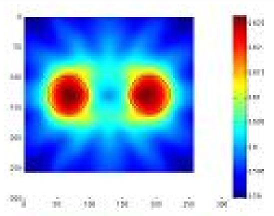
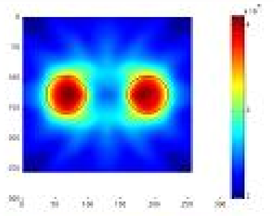
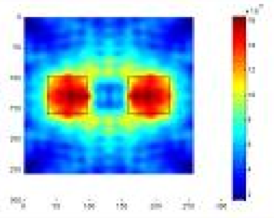
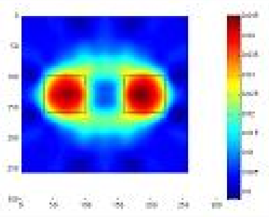
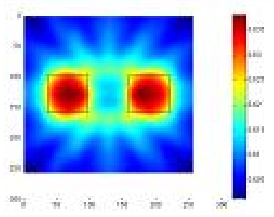
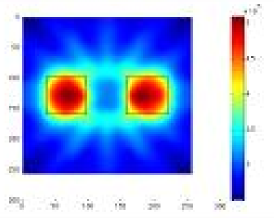
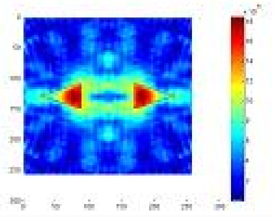
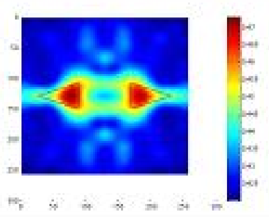
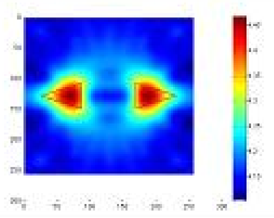
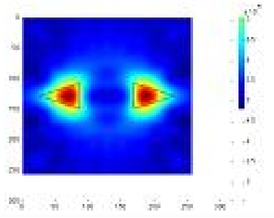
Corresponding Born Image for Each Ewald Circle Above						
						
						
	Born Reconstruction	**Image Space**	**Cepstrum Space**	**Born**	**Image Space**	**Cepstrum Space**
Composite Images						
	One Circle with Radius = 1 lambda			**Two Triangles**		
Corresponding Cepstrum Filtered Image for Each Born Image Above						
						
						

Table 6.20. A comparison of reconstructed images for various targets using the Born method, cepstrum filtering of the Born reconstructions, and processing sources independently and recombining in images space, and cepstrum space.

	Born Reconstruction with 12 sources @ 5GHz and 120 Receivers	Cepstrum of Born Reconstruction	Processed Independent Sources Combined in Image Space	Processed Sources Combined in Cepstrum Space
1 Circle Radius = λ er=1.8				
2 Circles Radius = λ er =1.5				
2 Squares Sides =2 λ er=1.4				
2 Triangles Height =3 λ er=1.5				

6.5 Analysis of Effects of Modified Filters in Cepstrum Space

While it is known and understood that in the cepstral filtering method, it is necessary to use some type of low pass filter to eliminate or attenuate the Ψ component in the data, there has been almost no research conducted to determine the optimal parameters for this filter. This is partly because there is little information for one to intuitively know where the "signal" data and "noise" are located in the cepstrum space. The cepstrum domain does not have a linear relationship to the spectral domain and so some experimentation is necessary to gain a better understanding of this domain to help to create the optimum filter for better image reconstruction. It was demonstrated in [6] that a Gaussian type filter performed reasonably well in filtering in the cepstrum domain and this was characterized by the following equation:

$$Filter = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-\left[x^2+y^2\right]}{2\sigma^2}} \tag{6.4}$$

The σ term in Eq. 6.4 dictates how wide the filter will be, which we can refer to as its bandwidth. Conventional filter theory would suggest that it is desirable to have the filter bandwidth to be as wide as possible to allow as much of the desirable signal spectrum to pass as possible, while at the same time narrow enough to significantly attenuate the un-desirable signal (or noise) as much as possible. This assumption is based on the noise having mostly higher spatial frequencies than those associated with V(r) which need not be necessarily the case, especially in the cepstrum domain. That being said, it seemed a logical place to start to vary the width or σ term and observe the effects, if any, on the performance of the various filtering methods described above.

Table 6.21. A comparison of reconstructed images for triangle targets using the combined source cepstrum filtering of the Born images, and processing sources independently and recombining in the image space, and also the cepstrum space for increasing σ. The triangles have a height of 3λ and ε_r=1.5.

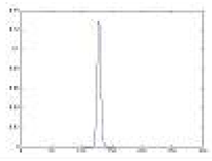
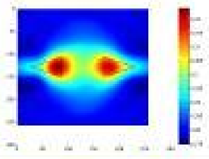
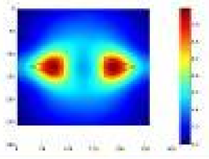
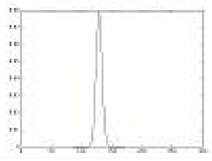
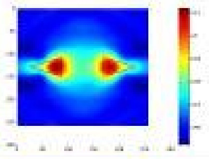
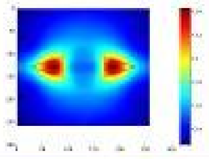
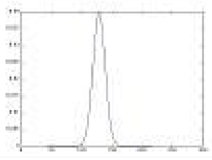
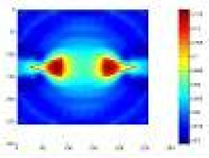
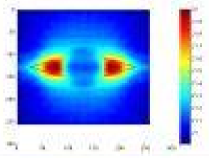
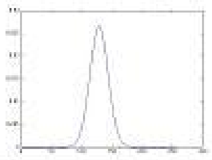
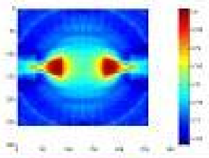
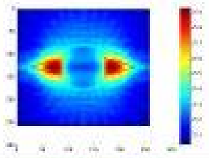
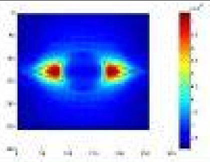
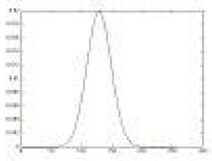
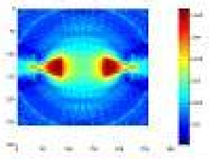
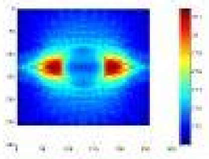
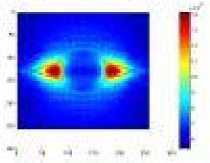
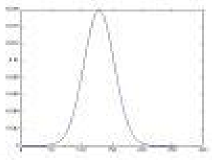
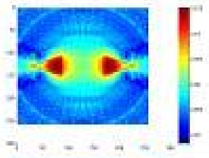
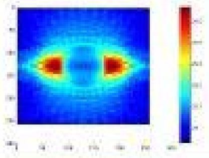
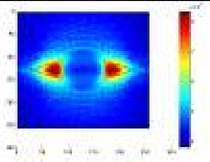
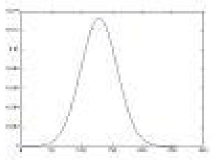
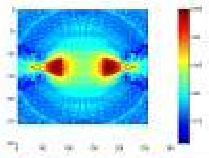
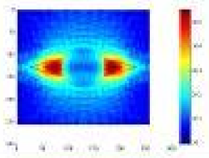
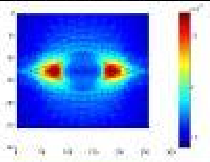
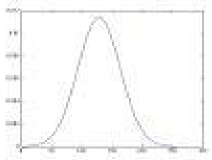
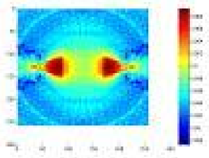
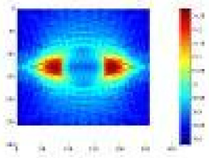
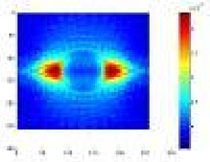
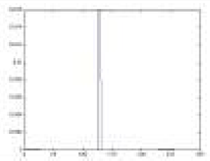
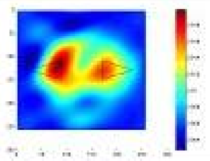
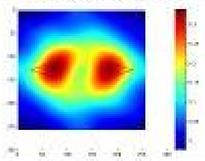
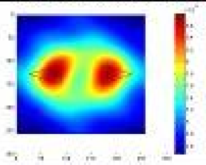
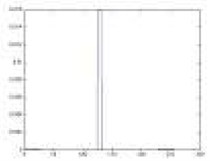
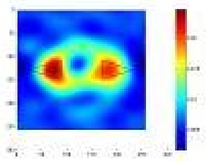
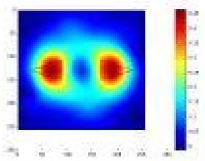
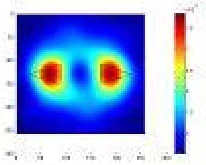
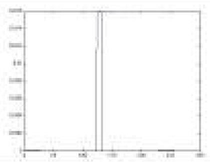
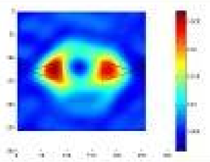
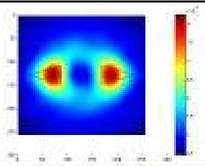
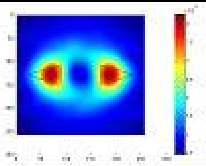
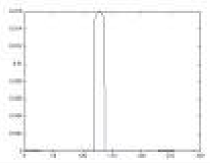
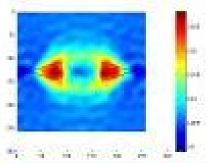
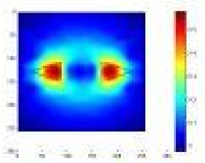
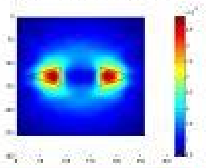
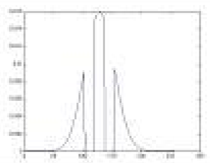
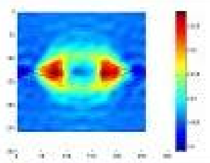
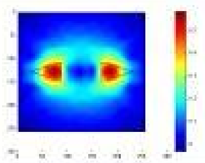
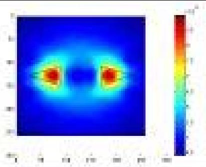
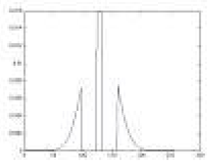
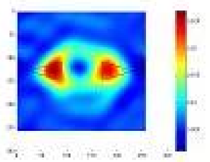
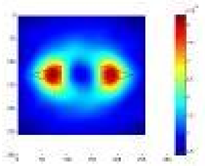
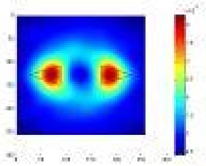
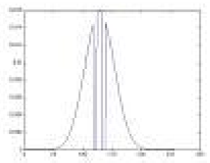
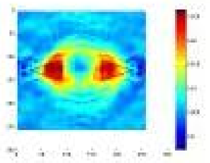
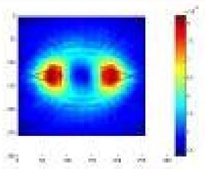
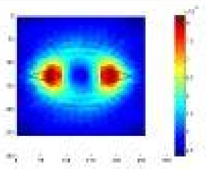
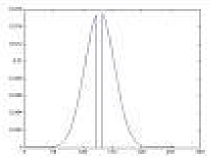
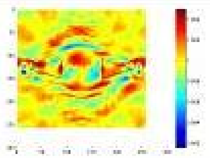
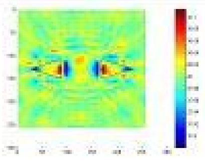
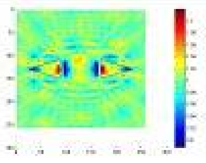
	Filters	36 Combined Source	Ind. Sources in Image Space	Ind. Sources in Cepstrum Space
3				No Valid Output for Reconstructed Image
5				No Valid Output for Reconstructed Image
10				No Valid Output for Reconstructed Image
15				
20				
25				
30				
35				

Table 6.21 contains a family of images that show how each of the cepstrum filtering methods mentioned in the previous section performs using a Gaussian filter described by Eq. 6.4 with gradually increasing width or σ term.

It is clear from the images in Table 6.21 that there is no significant gain or benefit to have a width for σ greater than about 10 for the particular family of targets considered here. It should also be noted that there was no valid reconstructed image output for the "combined in cepstrum space" filtering method for values of σ of 3 and 10. It will be shown later that this is due to the peak value of the filter which can be corrected by scaling. This being said, it should be noted that this form of Gaussian filter has a maximum value at the center that varies inversely as σ varies. The larger σ is the smaller the maximum of the filter is. It will be shown later that this plays a very significant role in the filtering process in the cepstrum domain.

Now that it is known that the majority, if not all, of the "good" or desirable part of the spectrum lies within σ=10 of the center of the filter, the next question is whether there is any location within this region that has "bad" or un-desirable data. To investigate this, a series of modified Gaussian filters were used to try and further characterize the mapping of data in the cepstrum domain. To do this, a Gaussian filter with a σ value set to 30 was strategically "notched" with gaps of varying widths and locations to see if there was any location "under" the Gaussian filter that is more important than any other for the "good" or the "bad" data. The results of this filter experiment and its effects on the various types of cepstrum filtering discussed previously is shown in Table 6.22.

Table 6.22. A comparison of reconstructed images for triangle targets using the combined source cepstrum filtering of the Born images, and processing sources independently and recombining in the image space, and also the cepstrum space for various filters. The triangles have a height of 3λ and ε_r=1.5.

Gap	Filters	Combined Source	Ind. Sources in Image Space	Ind. Sources in Cepstrum Space
3 to 100				
4 to 100				
5 to 100				
10 to 100				
10 to 25				
5 to 30				
5 to 10				
0 to 5				

It is evident again in Table 6.22 as in the previous table that the important signal data is located within the $\sigma = 10$ range. There does not seem to be any benefit, in general, to trying to include any data from the outer limits of the cepstrum domain. It is demonstrated clearly from the last set of images in Table 6.22, where the $\sigma = 0$ to $\sigma = 5$ range is filtered out that this is where the most important information is since the reconstructions for this condition are very poor and the original target image is undetermined even with 36 /360 degrees of freedom for sources/receivers respectively..

Now that the optimal value for sigma has been investigated and evaluated for the Gaussian filter, we will now investigate the peak value. It was observed in Table 6.21 that the "combined cepstrum" method did not produce any results for 3 of the scenarios presented. It was assumed that this was due to the fact that the pertinent "harmonics" were either outside the passband, or they were excessively attenuated due to the reduced peak that is inherent to the Gaussian function. Experimentation was done on the peak value of the Gaussian by using simple scaling techniques and it was discovered that the issue was not that the Gaussian peak was too low, but in fact it was too high. The fact is that it was observed that there is an inverse relationship between the peak value of the Gaussian filter and the overall range in magnitude of the resulting image using the "combined cepstrum" method. Moreover, it was observed through simulations that the minimum constraint for the peak of the filter to insure that the "combined cepstrum" method produced a reconstructed image is that the peak must be equal to or less than 1/Ns, where Ns is the total number of sources. It was also observed that, in general, to scale the magnitude of the reconstructed image produced by the "combined cepstrum" method to be in the same general range as the magnitude of the image produced using the

Born reconstruction, the filter needed to be scaled by an additional value of 2/3. The peak values for each number of sources are shown in Figure 6.6. Also in this Figure the trend line for this process is drawn and compared with the ideal trend line of 1/Ns which demonstrates the conclusion of 1/Ns for the minimum constraint for the Gaussian filter for the "combined cepstrum" method. In addition, the line for the "2/3" adjusted line for magnitude adjustment is shown as well.

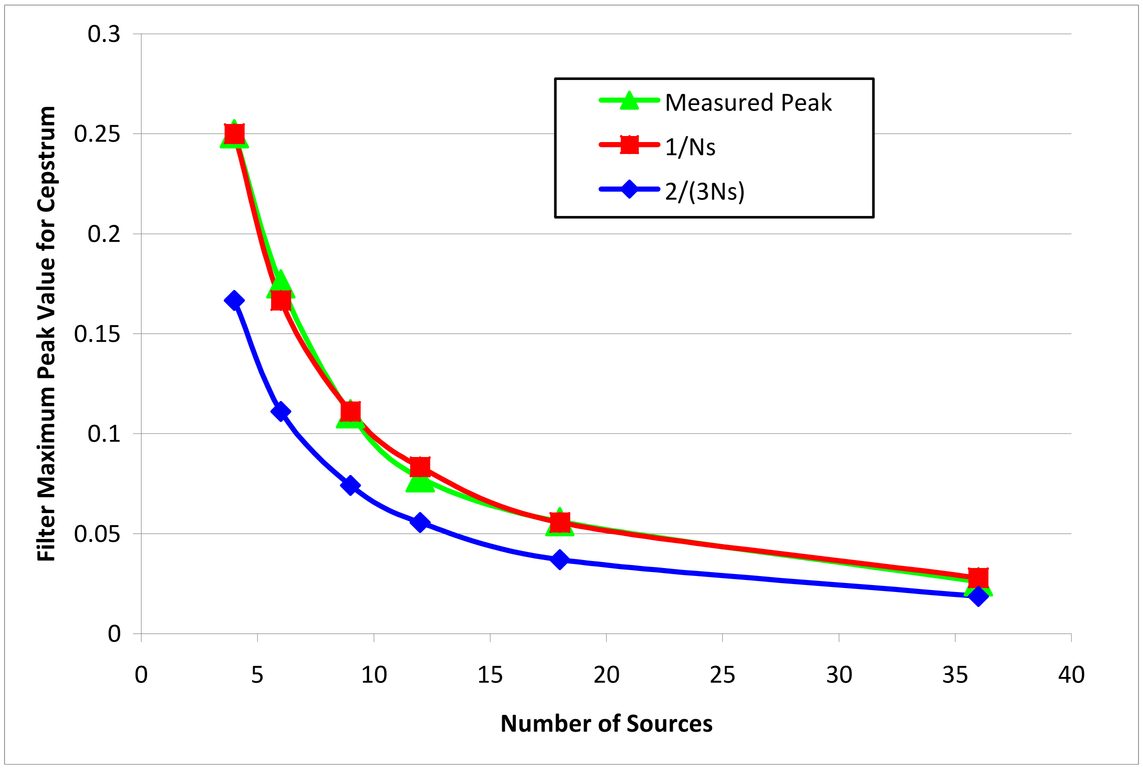
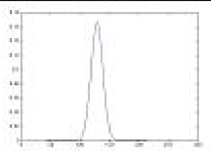
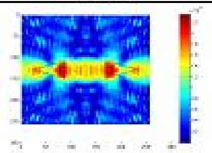
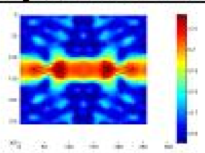
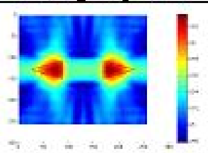
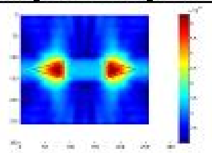
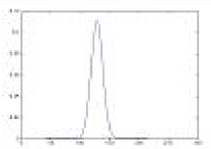
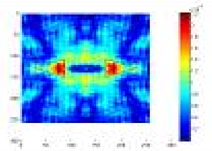
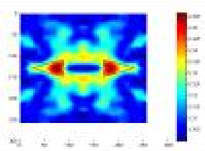
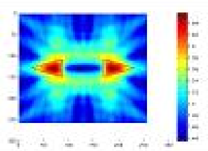
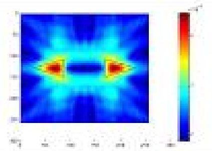
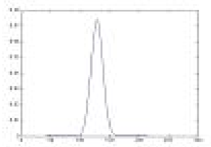
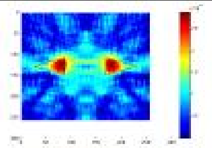
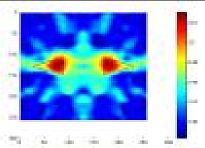
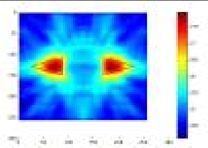
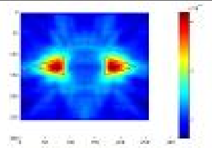
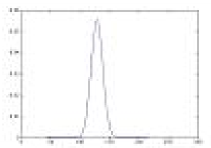
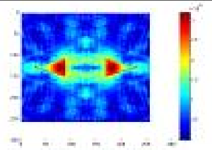
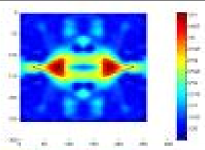
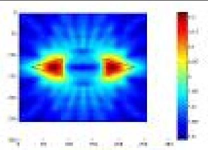
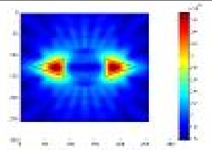
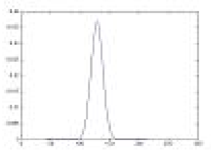
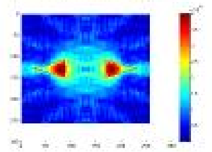
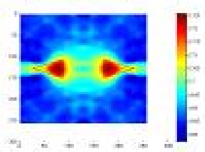
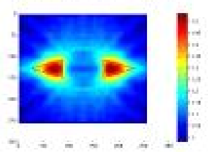
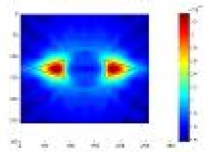
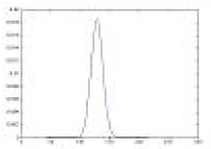
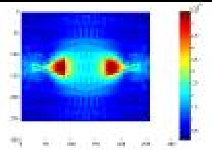
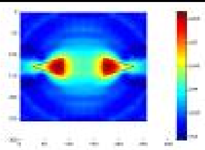
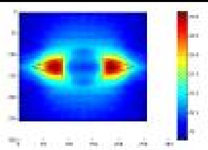
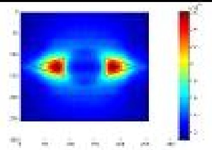


Figure 6.6. Filter peak values versus the number of sources used required causing the "combined cepstrum" method to produce a reconstructed image.

Using the "optimized" value for sigma, and the new peak scaling factor of 2/(3Ns), the process was run again for the two triangles target set for various number of sources using each of the processing methods for comparison of performance. The results of this experiment are shown in Table 6.18. The "optimized combined cepstrum" method appears to consistently perform the best out of all of the methods considered.

Table 6.23. A comparison of reconstructed images for triangle targets using the combined source cepstrum filtering of the Born images, and processing sources independently and recombining in images space, and cepstrum space for “optimized” filters. Images were reconstructed using 36 sources (5GHz) and, 360 receivers all equally spaced. The triangles have a height of 3λ and ε_r=1.5.

Ns	Filter	Born	Cepstrum of Born	Image Space	Cepstrum Space
4					
6					
9					
12					
18					
36					

The final aspect of investigating filtering or removal of unwanted parts of the spectrum or data consists of trying to remove or subtract a cepstrum representation of the incident field, Ψ_{inc}. The reasoning behind this, as discussed in Section 5.3 is that in the cepstrum domain after the reference has been added we have a representation of $V\Psi$ in the form of Eq. 5.12 as follows

$$\approx \log\left(\frac{\text{V}}{R}\right)+\frac{\langle\Psi\rangle}{R'} \tag{6.5}$$

As discussed in Section 5.13 it is thought that by subtracting a weighted ceptrum representation of Ψ_{inc} we would then have

$$\approx \log\left(\frac{\text{V}}{R}\right)+\frac{\langle\Psi\rangle}{R'}-\kappa\langle\Psi_{inc}\rangle \tag{6.6}$$

which as stated before if the factor for "κ" can be found that is close to $1/R'$, and $<\Psi_{\text{inc}}>$ is a close approximation to $<\Psi>$ then improvement in the resulting reconstructed image can be expected.

This is what was implemented in the MATLAB code for both the combined source method and the independent source processing method combined in the cepstrum domain. The effect on the combined source method was observed first. It appears that with a weighting factor of about .4, the reconstructed image does seem to fill more of the original target boundary. Examples of this are shown in Tables 6.24 and 6.25. The independent source method was observed next and while there was minimal change in the image itself, for a weighting factor of 1 a significant improvement in the scale of the reconstructed image was noted as shown in Tables 6.26 through 6.29.

Table 6.24. A comparison of reconstructed images for triangle targets with dimensions of 3λ using the combined source cepstrum filtering of the Born images for three different permittivity values. These images compare the results of subtracting the cepstrum version of the incident field times a weighting factor of .4 (-.4xΨ_{inc}).

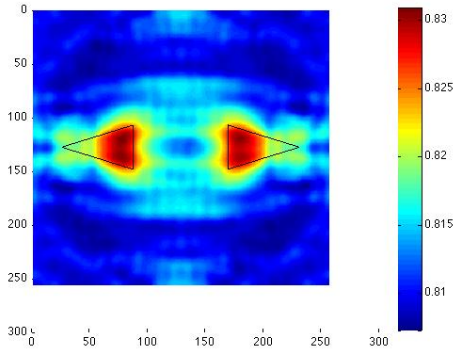
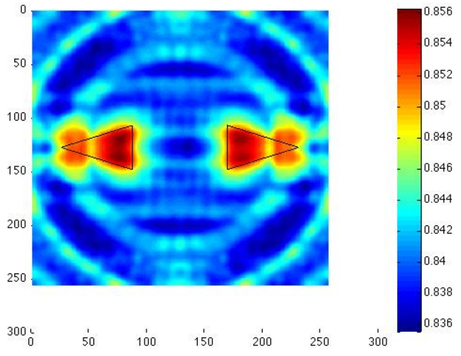
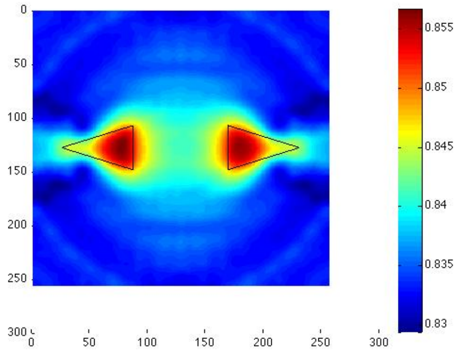
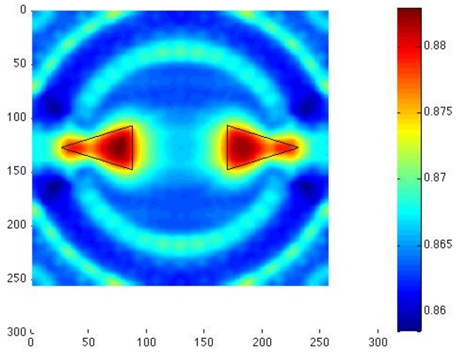
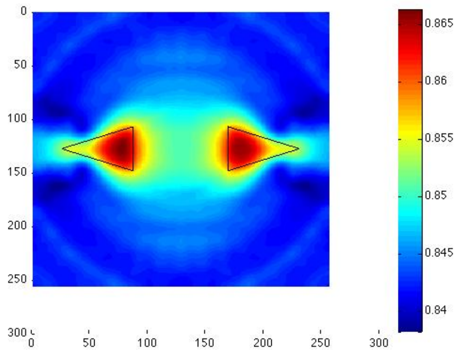
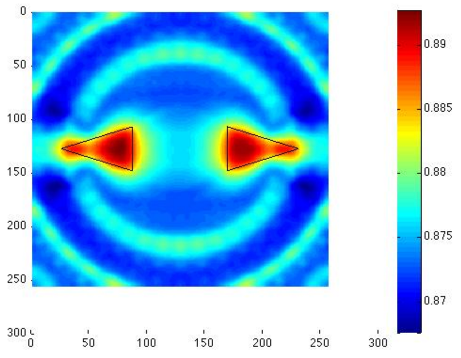
	Reconstructed image by processing combined sources in the Fourier Domain and then filtered in the cepstrum domain	Reconstructed image by processing combined sources in the Fourier Domain and then filtered in the cepstrum domain while subtracting .4xΨ_{inc}
er=1.1		
er=1.5		
er=1.9		

Table 6.25. A comparison of reconstructed images for square targets with sides of 2λ using the combined source cepstrum filtering of the Born images for three different permittivity values. These images compare the results of subtracting the cepstrum version of the incident field times a weighting factor of .4 (-.4xΨ_{inc}).

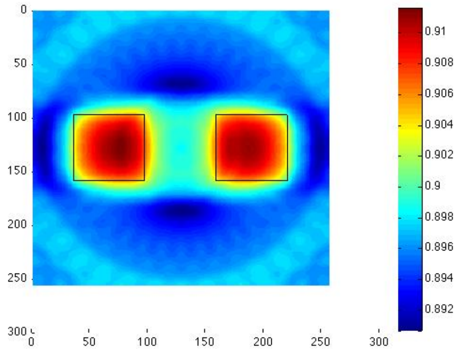
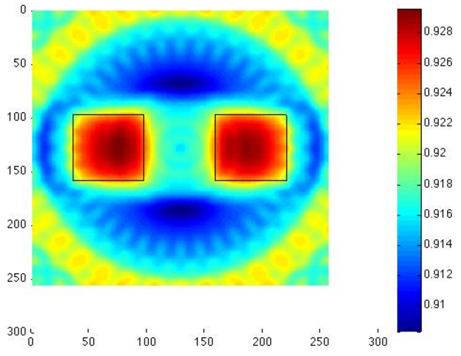
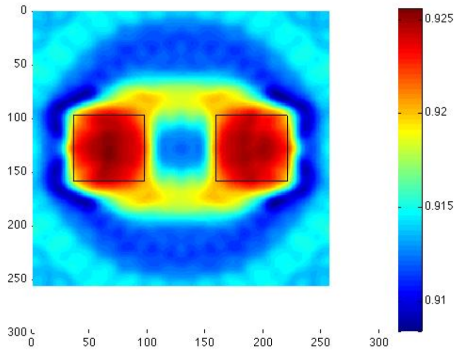
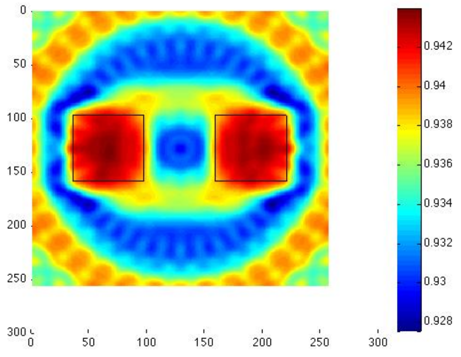
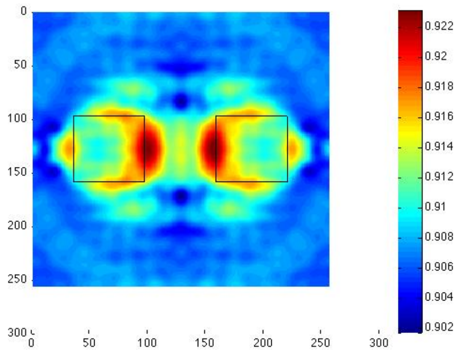
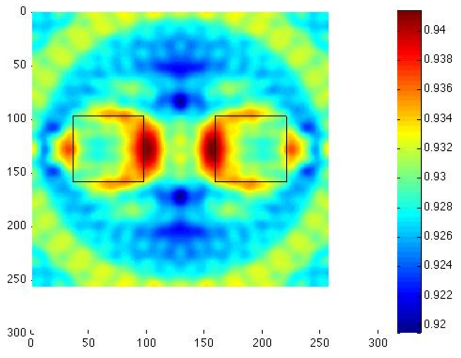
	Reconstructed image by processing combined sources in the Fourier Domain and then filtered in the cepstrum domain	Reconstructed image by processing combined sources in the Fourier Domain and then filtered in the cepstrum domain while subtracting .4xΨ_{inc}
er=1.1		
er=1.5		
er=1.9		

Table 6.26. A comparison of reconstructed images for a square target with sides of 2λ using the method of processing each source separately and then combining them in cepstrum space after filtering for three different permittivity values. These images compare the results of subtracting the cepstrum version of the incident (Ψ_{inc}). The most notable change being the change in scale.

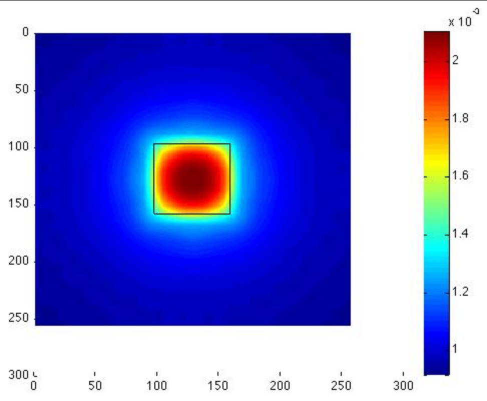
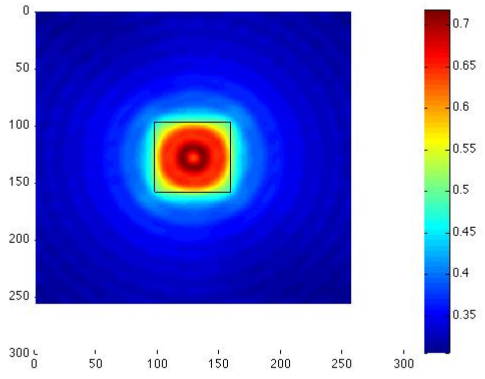
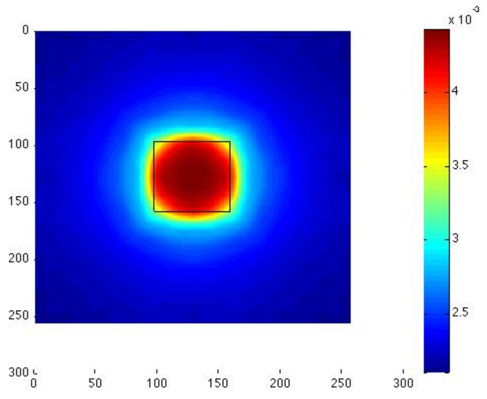
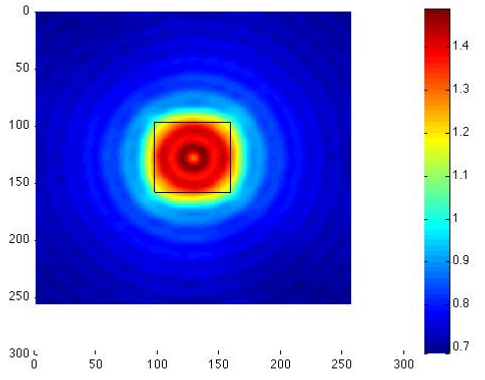
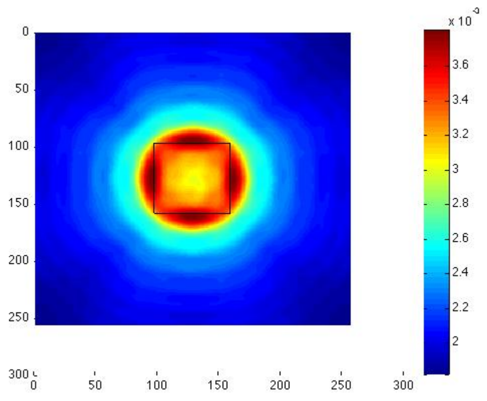
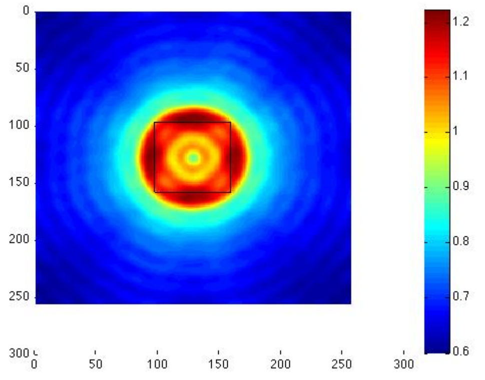
	Reconstructed image processing each source separately and combining in the cepstrum domain	Reconstructed image processing each source separately and combining in the cepstrum domain while subtracting Ψ_{inc}
er=1.1		
er=1.5		
er=1.9		

Table 6.27. A comparison of reconstructed images for square targets with sides of 2λ in length using the method of processing each source separately and then combining them in cepstrum space after filtering, repeated for three different permittivity values. These images compare the results of subtracting the cepstrum version of the incident (Ψ_{inc}). The most notable change being the change in scale.

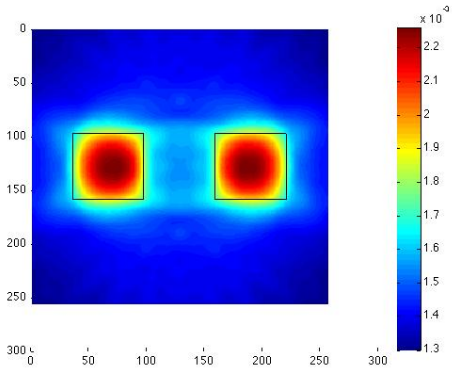
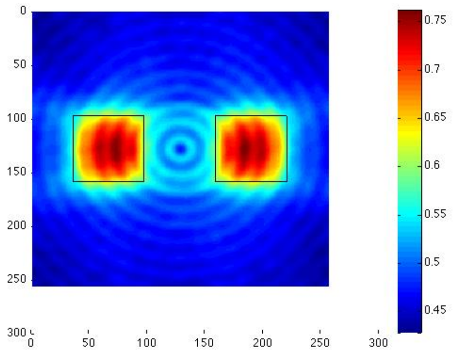
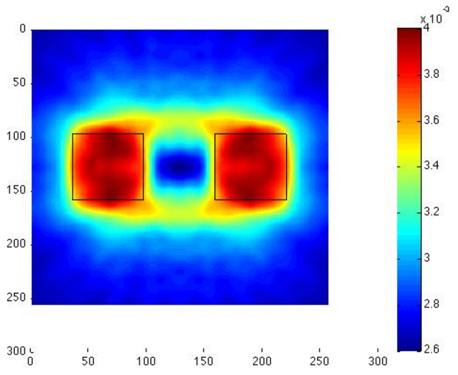
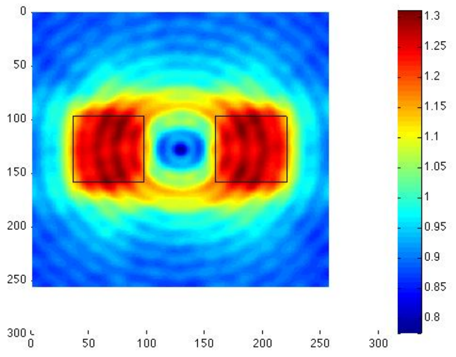
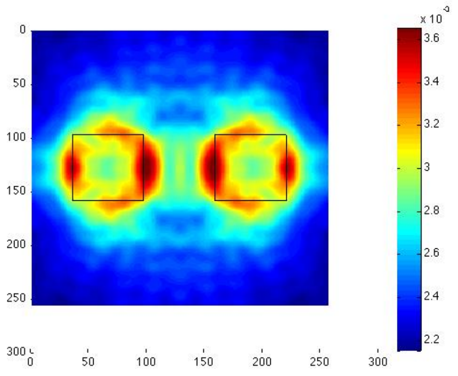
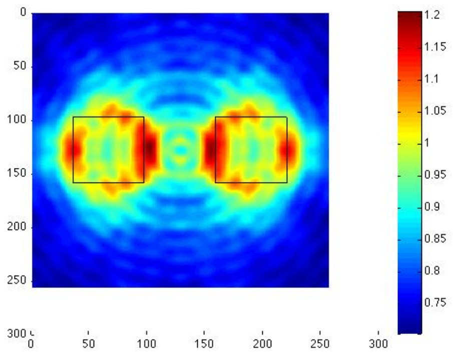
	Reconstructed image processing each source separately and combining in the cepstrum domain	Reconstructed image processing each source separately and combining in the cepstrum domain while subtracting Ψ_{inc}
er=1.1		
er=1.5		
er=1.9		

Table 6.28. A comparison of reconstructed images for circular targets with radius of λ using the method of processing each source separately and then combining them in cepstrum space after filtering for three different permittivity values. These images compare the results of subtracting the cepstrum version of the incident (Ψ_{inc}). The most notable change being the change in scale.

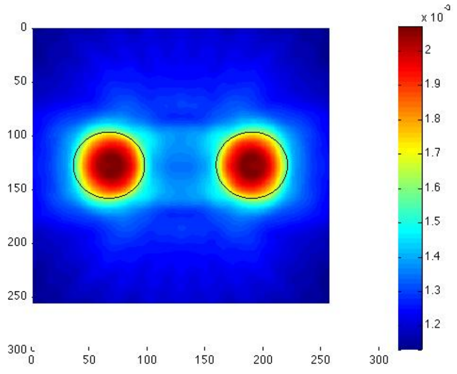
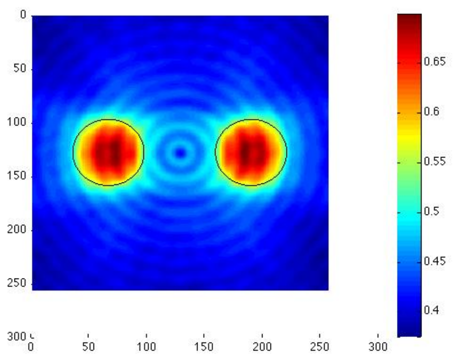
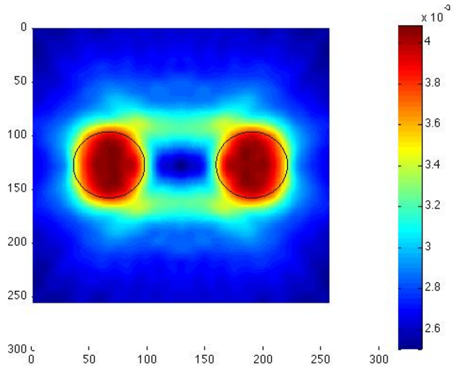
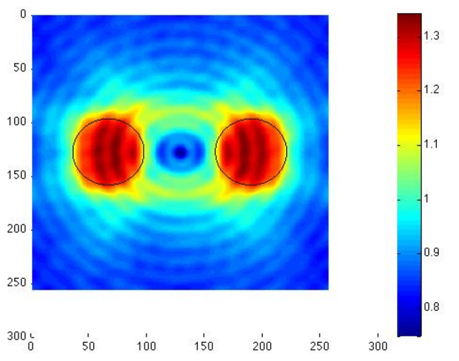
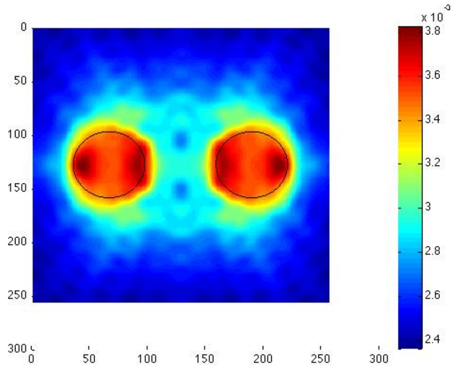
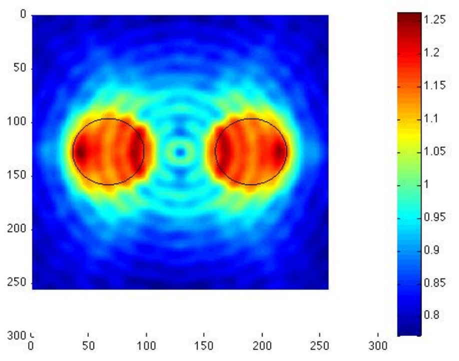
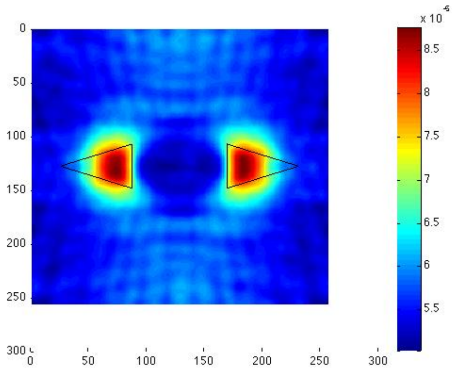
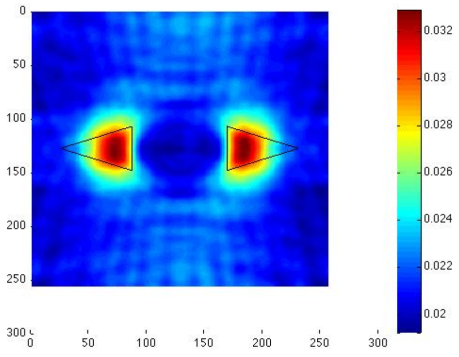
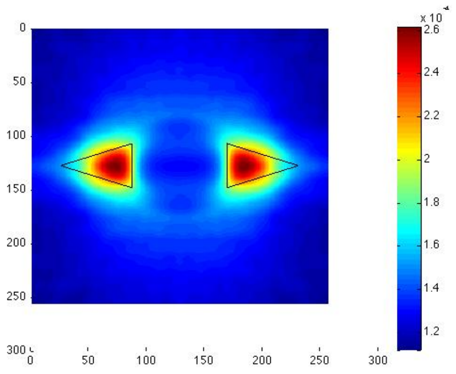
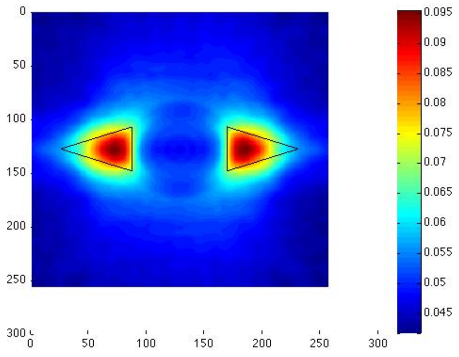
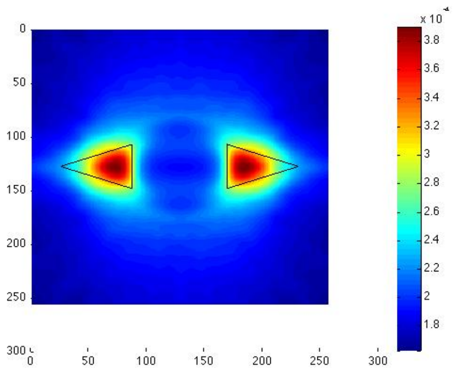
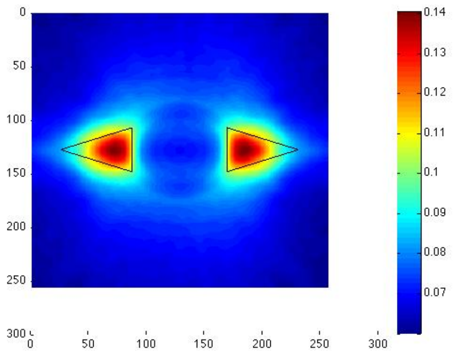
	Reconstructed image processing each source separately and combining in the cepstrum domain	Reconstructed image processing each source separately and combining in the cepstrum domain while subtracting Ψ_{inc}
er=1.1		
er=1.5		
er=1.9		

Table 6.29. A comparison of reconstructed images for triangular targets with dimensions of 3λ using the method of processing each source separately and then combining them in cepstrum space after filtering for three different permittivity values. These images compare the results of subtracting the cepstrum version of the incident (Ψ_{inc}). The most notable change being the change in scale.

	Reconstructed image processing each source separately and combining in the cepstrum domain	Reconstructed image processing each source separately and combining in the cepstrum domain while subtracting Ψ_{inc}
er=1.1		
er=1.5		
er=1.9		

6.6 Analysis of the Effects of Random Under-Sampling

As briefly mentioned in Section 6.2, aliasing can become a significant issue as the number of receivers is reduced and the spacing between the receivers is increased. The issue becomes critical as the spacing between the receivers equals or exceeds the wavelength of the incident field. Also mentioned in Section 6.2 was that the use of random spacing for the receivers as they are reduced in number, can be very effective in dealing with the aliasing [42] and provide a means to reduce the number of receivers needed to effectively sample the source [43] [44].

For the model setup for this research, the receiver spacing from the center of the target area is 760mm. It was observed at the output of the cepstrum filtering method earlier that aliasing started to appear once the number of receivers dropped below 90 for an incident frequency of 5GHz. The wavelength of the incident frequency of 5GHz is 60mm. In general, the spacing between equispaced receivers in 2 dimensions is calculated by dividing the circumference by the number of receivers or

$$Spacing = \frac{Circumference}{\#\mathrm{Re}cievers} = \frac{2\pi r}{90}$$

which for the case mentioned above, the spacing is 53mm. This is still smaller than the wavelength, which corresponds to the highest spatial frequency, but as the receivers are reduced in number further, this spacing increases to be greater than the wavelength, hence the increased likelihood of aliasing. For the frequency ranges used in this research, the aliasing threshold can be determined rather straightforwardly using the above relationship and comparing it to the wavelength of the incident frequency in use. Table 6.30 shows how the number of receivers, their associated receiver spacing and wavelengths of incident frequencies compare and it shows the aliasing threshold for each.

Table 6.30. Comparison of receiver spacing and incident frequency wavelength to illustrate the aliasing threshold limits for each scenario.

# of Receivers	Spacing (m)	λ_{inc} (m)	f_{inc} (GHz)
360	0.013	0.030	10
180	0.027	0.033	9
		0.038	8
120	0.040	0.043	7
		0.050	6
90	0.053	0.060	5
72	0.066	0.075	4
60	0.080	0.100	3
45	0.106		
40	0.119		
36	0.133	0.150	2
30	0.159	0.300	1
24	0.199		
20	0.239		
18	0.265		
15	0.318		
12	0.398		
10	0.478		

Using the data from Table 6.30, simulations were run for each incident frequency in the range of the aliasing threshold limit predicted using the relationship described above. The results of this experiment are shown in Table 6.31 and clearly show the accuracy of predicting when aliasing will occur. Tables 6.32 through 6.34 also show examples of how applying the random receiver spacing method introduced in Section 2 addresses this aliasing for the images processed using the cepstrum method for a circle using various incident frequencies. In addition Tables 6.35 through 6.39 show anti-aliasing results for various target types demonstrating that the result of applying this technique to undersampled images is quite favorable in general.

Table 6.31. A comparison of reconstructed images for a circular target with radius = λ using the Born approximation, and processing sources independently and recombining in cepstrum space. 36 Sources are used with varying numbers of receivers equally and spaced to show the effects of aliasing after the number of receivers drops below the aliasing limit. Green indicates above the aliasing limit, Yellow indicates at the threshold of the aliasing limit, and Red indicates below the aliasing limit.

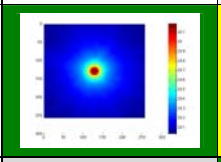
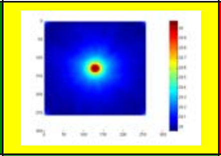
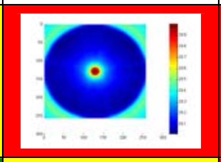
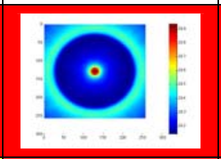
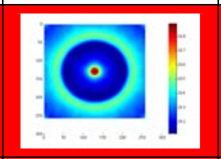
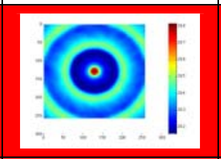
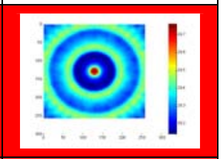
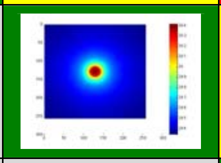
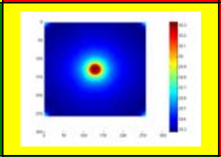
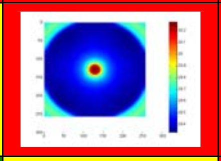
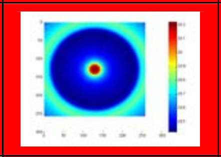
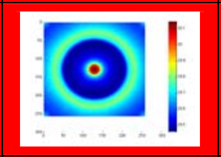
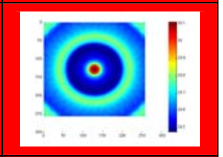
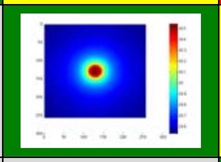
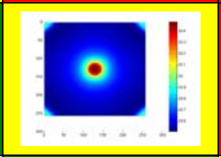
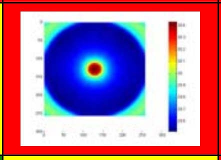
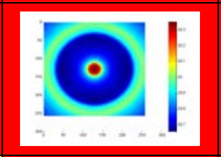
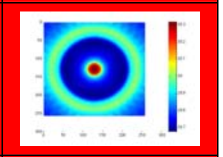
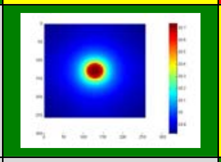
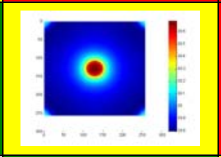
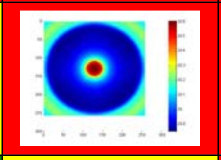
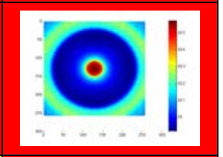
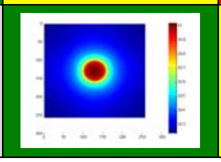
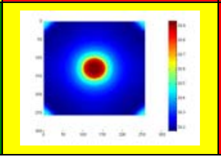
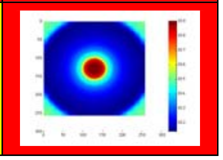
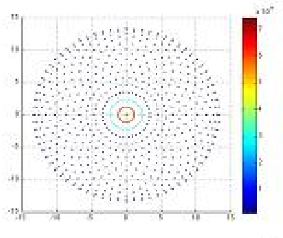
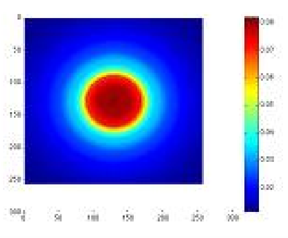
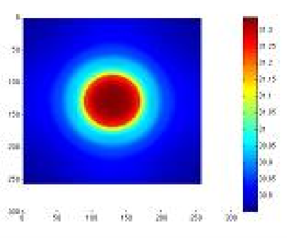
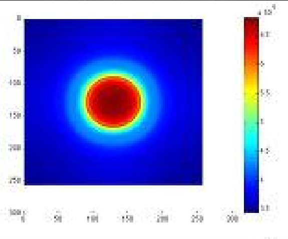
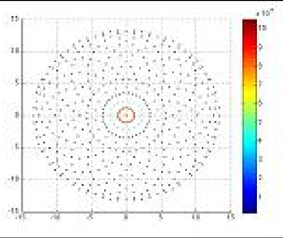
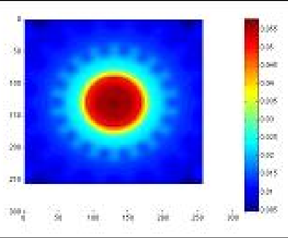
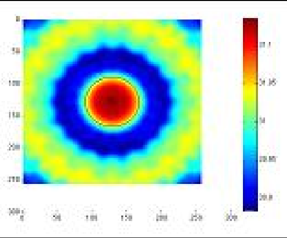
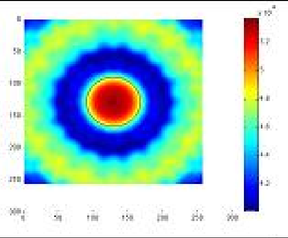
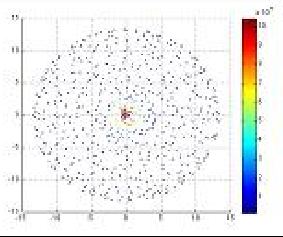
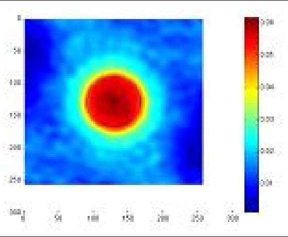
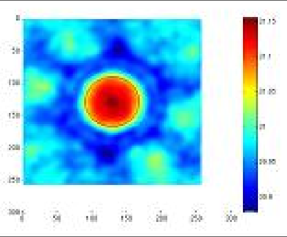
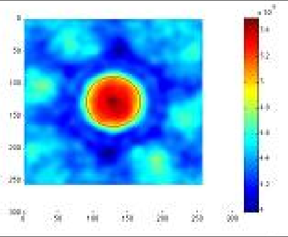
		Number of Receivers						
		180	120	90	72	60	45	40
Incident Frequency	8GHz	🟩 	🟨 	🟥 	🟥 	🟥 	🟥 	🟥 
	6 GHz		🟩 	🟨 	🟥 	🟥 	🟥 	🟥 
	5 GHz			🟩 	🟨 	🟥 	🟥 	🟥 
	4 GHz				🟩 	🟨 	🟥 	🟥 
	3 GHz					🟩 	🟨 	🟥 

Table 6.32. A comparison of reconstructed images for a circular target with radius = λ for an incident frequency of 2GHz using the Born approximation, and processing sources independently. 36 Sources are used with varying numbers of receivers equally and randomly spaced to show the effects of the anti-aliasing technique of random receiver spacing.

	Combined Ewald Circles	Born	Image Space	Cepstrum Space

er=1.5
Receivers = 36
Noise BW = 0
				

er=1.5
Receivers = 18
Noise BW = 0
				

er=1.5
Receivers = 18
Noise BW = 16
				

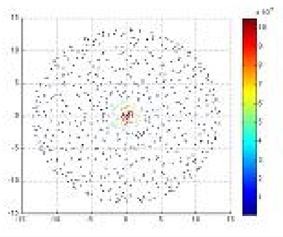
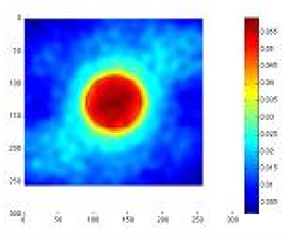
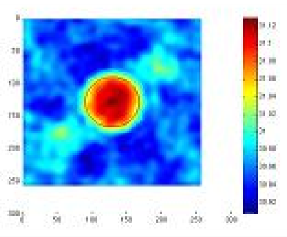
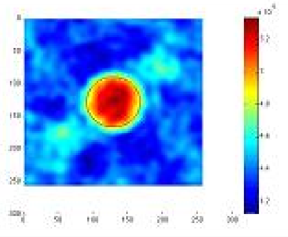
er=1.5
Receivers = 18
Noise BW = 20
				

Table 6.33. A comparison of reconstructed images for a circular target with radius = λ for an incident frequency of 5GHz using the Born approximation, and processing sources independently. 36 Sources are used with varying numbers of receivers equally and randomly spaced to show the effects of the anti-aliasing technique of random receiver spacing.

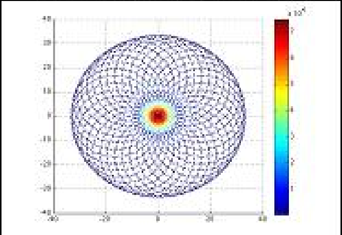
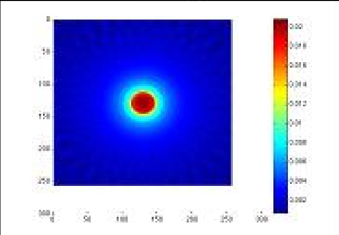
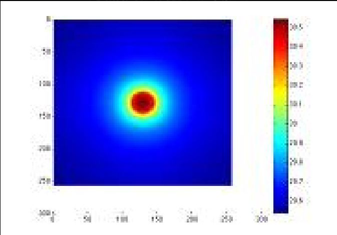
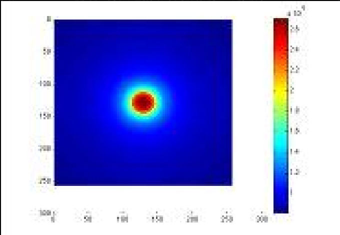
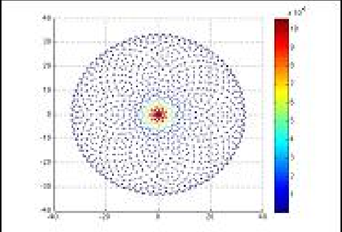
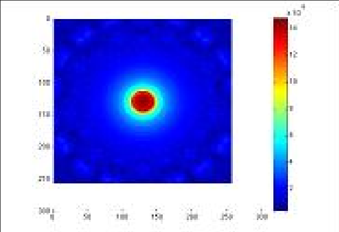
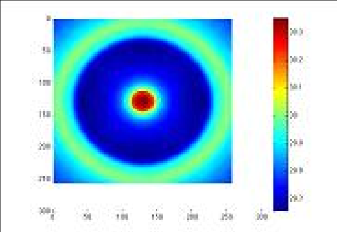
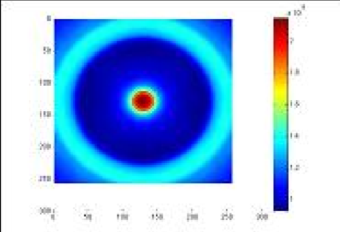
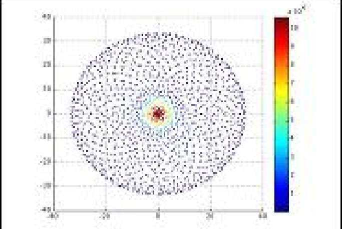
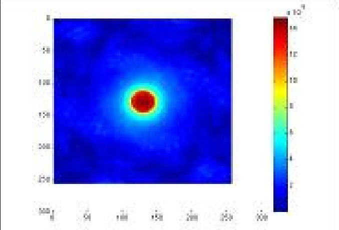
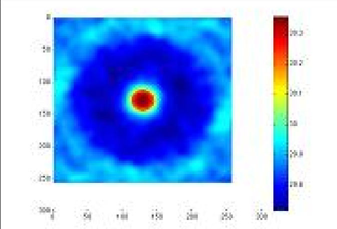
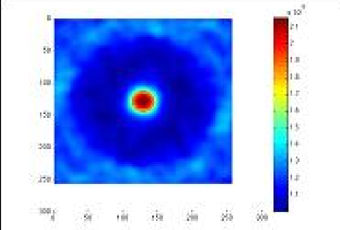
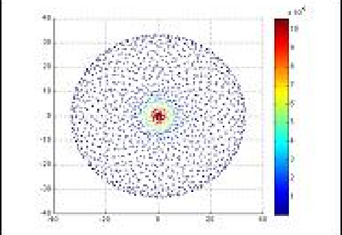
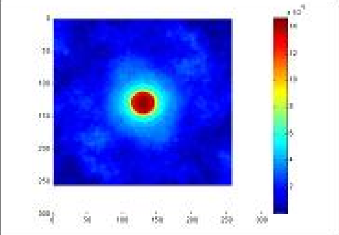
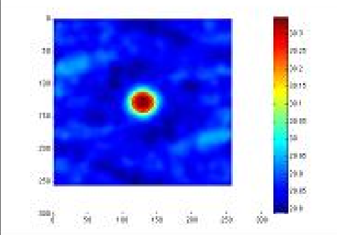
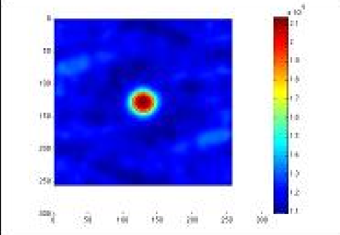
	Combined Ewald Circles	Born	Image Space	Cepstrum Space
er=1.5 Receivers = 90 Noise BW = 0				
er=1.5 Receivers = 45 Noise BW = 0				
er=1.5 Receivers = 45 Noise BW = 4				
er=1.5 Receivers = 45 Noise BW = 6				

Table 6.34. A comparison of reconstructed images for a circular target with radius = λ for an incident frequency of 8GHz using the Born approximation, and processing sources independently. 36 Sources are used with varying numbers of receivers equally and randomly spaced to show the effects of the anti-aliasing technique of random receiver spacing.

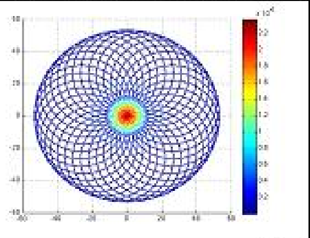
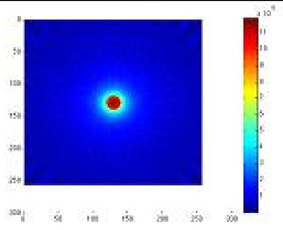
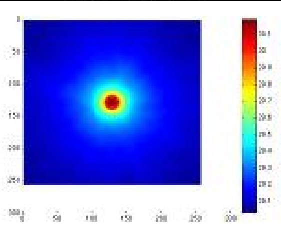
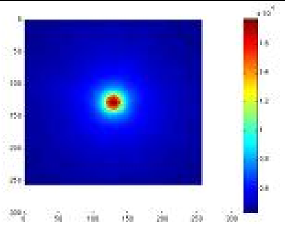
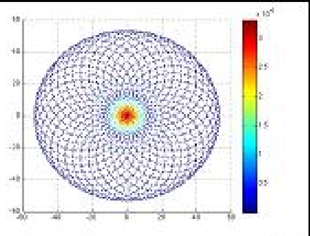
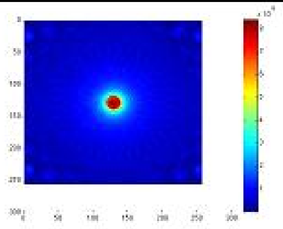
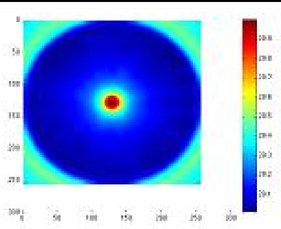
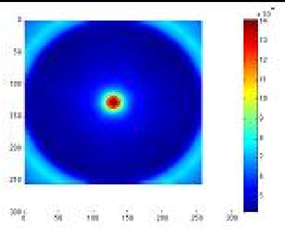
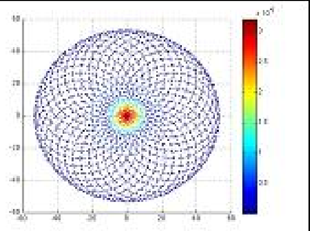
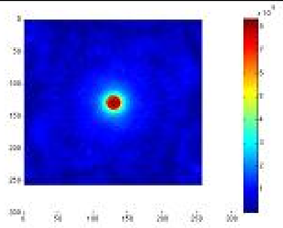
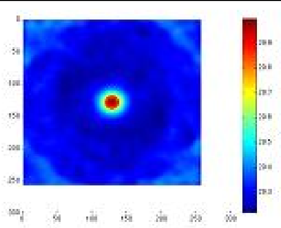
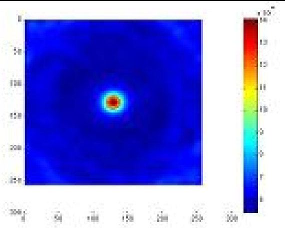
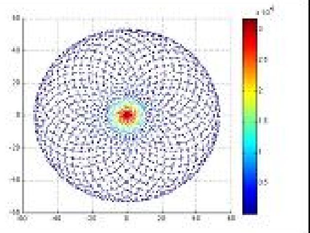
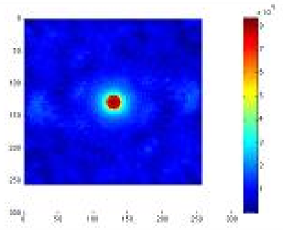
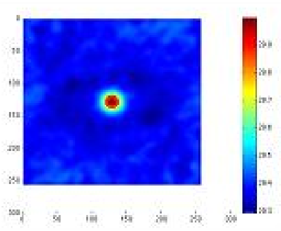
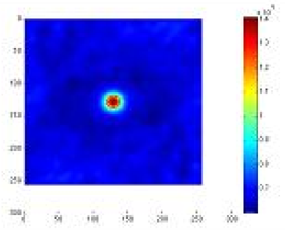
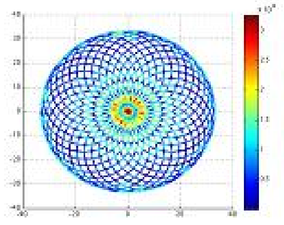
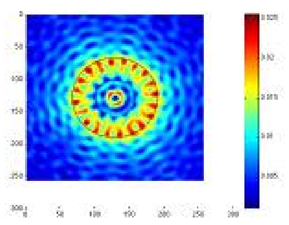
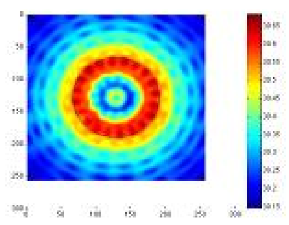
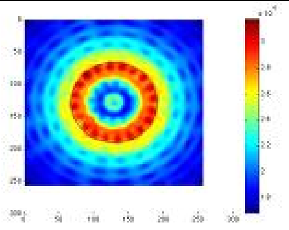
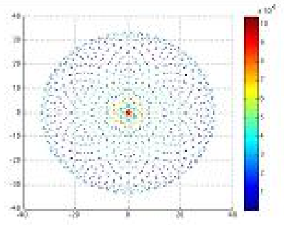
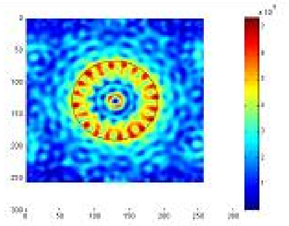
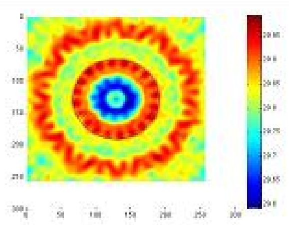
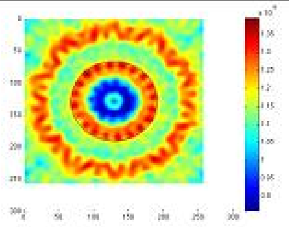
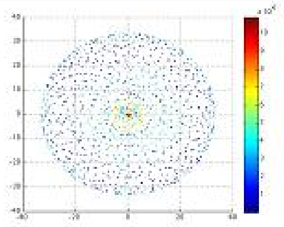
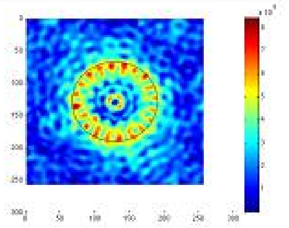
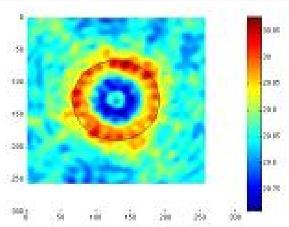
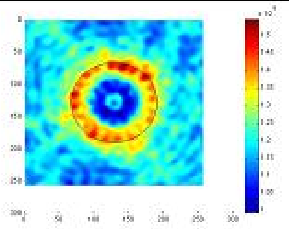
	Combined Ewald Circles	Born	Image Space	Cepstrum Space
er=1.5 Receivers = 180 Noise BW = 0				
er=1.5 Receivers = 90 Noise BW = 0				
er=1.5 Receivers = 90 Noise BW = 2				
er=1.5 Receivers = 90 Noise BW = 4				

Table 6.35. A comparison of reconstructed images for a circular target with radius = 2 λ and increased permittivity using the Born approximation, and processing sources independently and recombining in image space, and cepstrum space. 36 Sources are used with varying numbers of receivers equally and randomly spaced to show the effects of aliasing and anti-aliasing techniques.

	Combined Ewald Circles	Born	Image Space	Cepstrum Space

er=6
Receivers = 360
Noise BW = 0
				

er=6
Receivers = 30
Noise BW = 0
				

er=6
Receivers = 30
Noise BW = 12
				

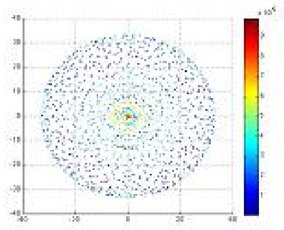
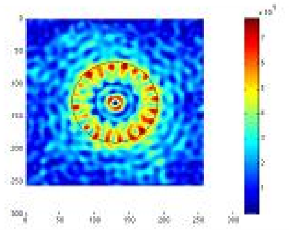
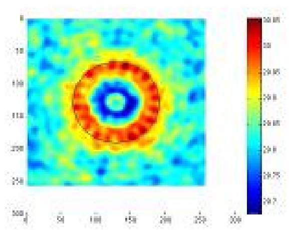
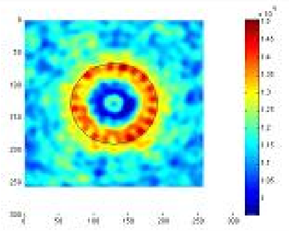
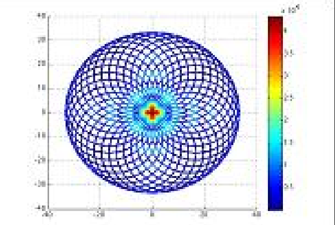
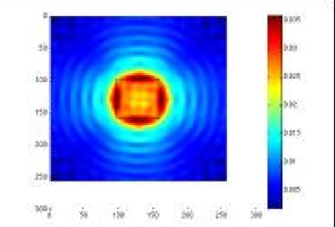
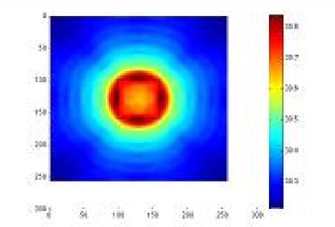
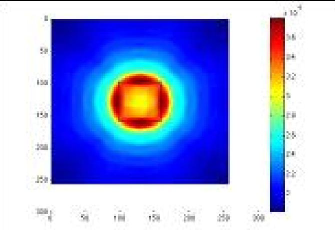
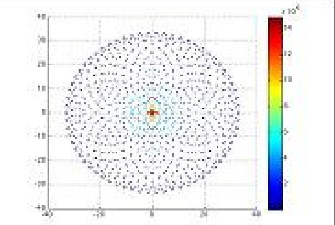
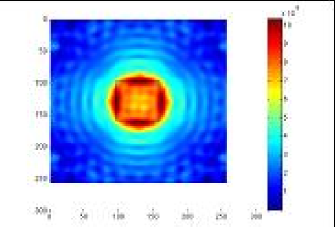
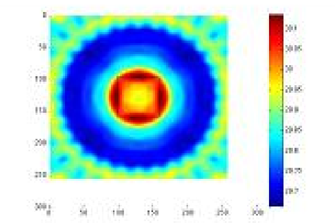
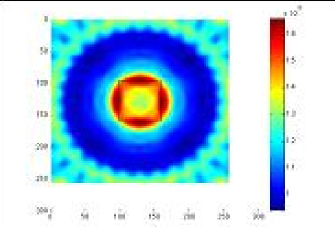
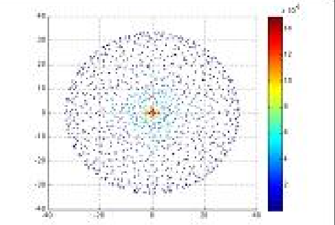
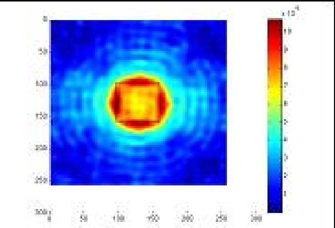
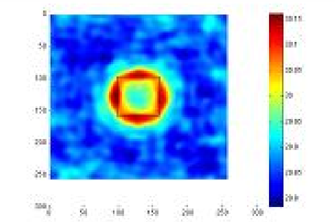
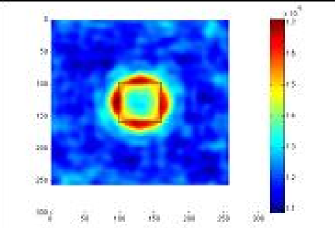
er=6
Receivers = 36
Noise BW = 10
				

Table 6.36. A comparison of reconstructed images for a square target with sides = 2 λ using the Born approximation, and processing sources independently and recombining in images space, and cepstrum space. 36 Sources are used with varying numbers of receivers equally and randomly spaced to show the effects of aliasing and anti-aliasing techniques.

	Combined Ewald Circles	Born	Image Space	Cepstrum Space

er=1.9
Receivers = 360
Noise BW = 0
				

er=1.9
Receivers = 30
Noise BW = 0
				

er=1.9
Receivers = 30
Noise BW = 12
				

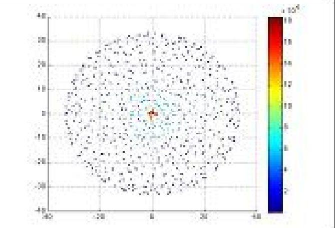
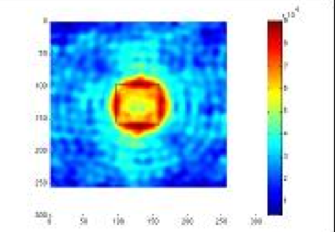
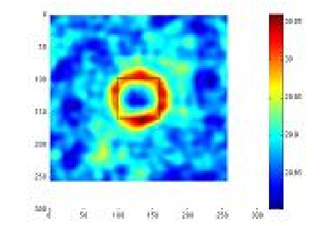
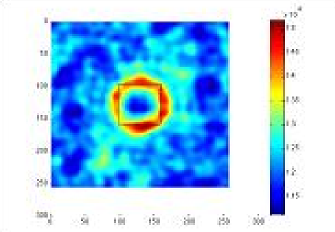
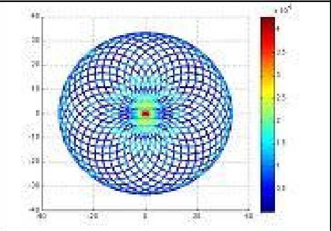
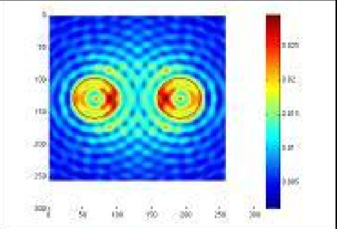
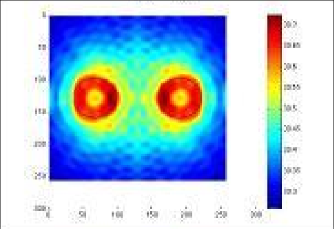
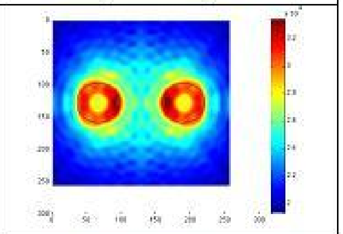
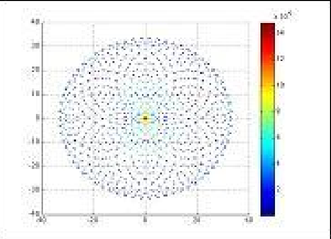
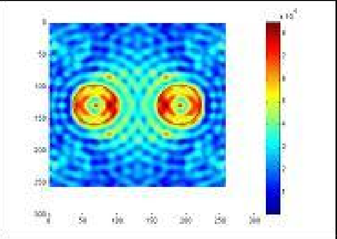
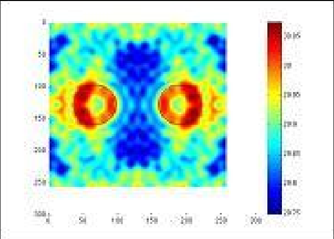
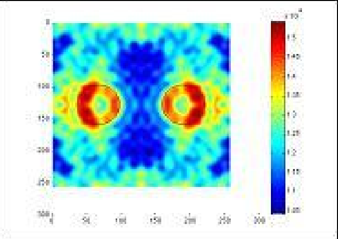
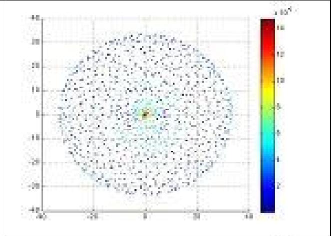
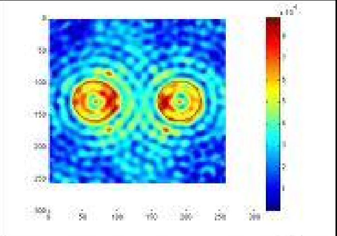
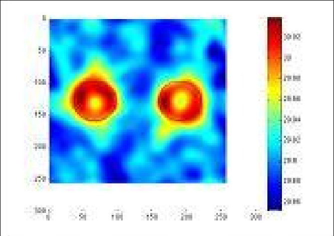
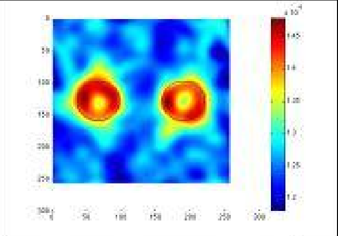
er=1.9
Receivers = 20
Noise BW = 18
				

Table 6.37. A comparison of reconstructed images for a 2 circle targets each with radius = λ using the Born approximation, and processing sources independently and recombining in images space, and cepstrum space. 36 Sources are used with varying numbers of receivers equally and randomly spaced to show the effects of aliasing and anti-aliasing techniques.

	Combined Ewald Circles	Born	Image Space	Cepstrum Space

er=6
Receivers = 360
Noise BW = 0
				

er=6
Receivers = 30
Noise BW = 0
				

er=6
Receivers = 30
Noise BW = 12
				

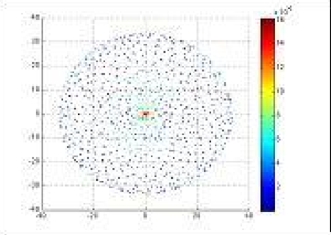
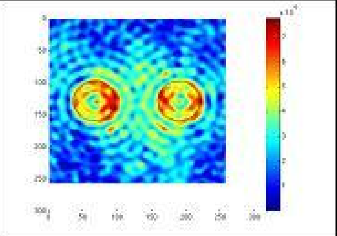
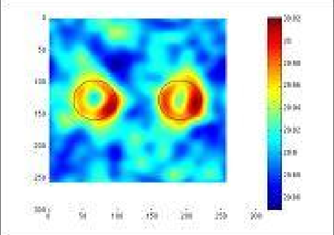
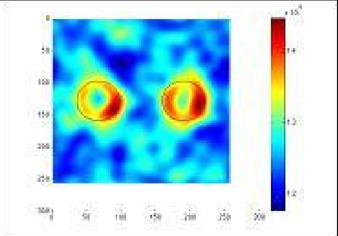
er=6
Receivers = 24
Noise BW = 14
				

Table 6.38. A comparison of reconstructed images for a 2 square targets each with sides = 2λ using the Born approximation, and processing sources independently and recombining in images space, and cepstrum space. 36 Sources are used with varying numbers of receivers equally and randomly spaced to show the effects of aliasing and anti-aliasing techniques.

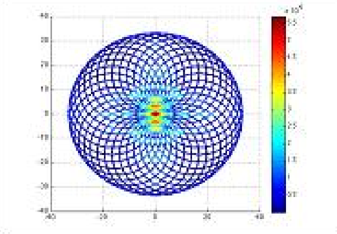
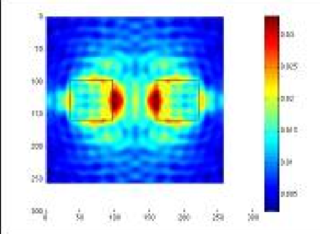
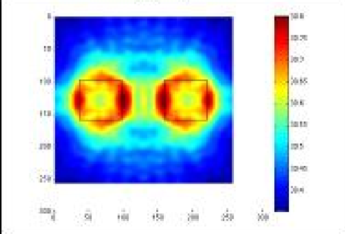
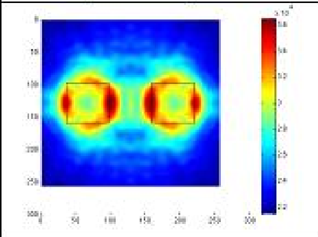
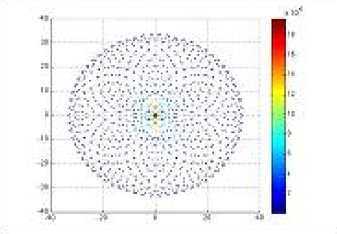
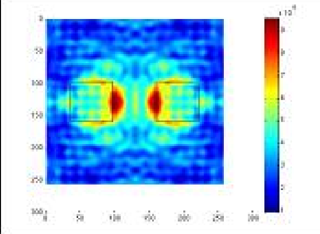
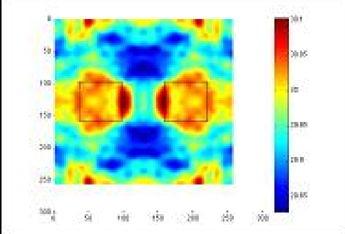
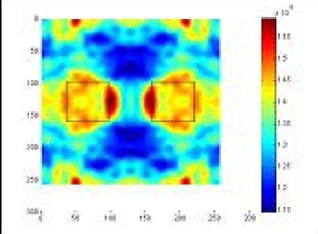
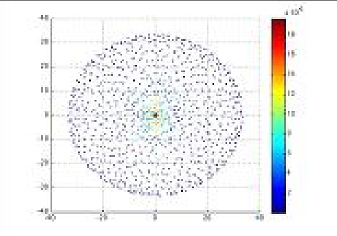
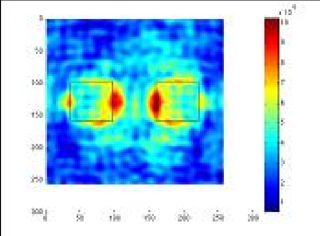
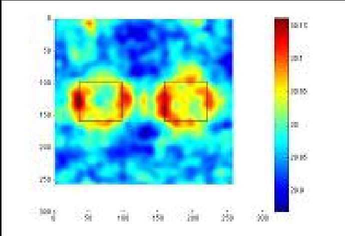
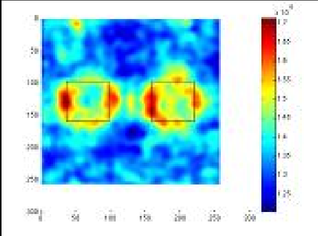
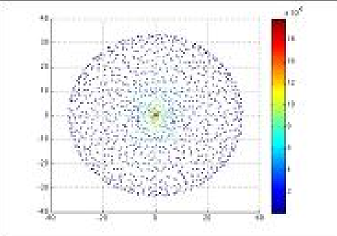
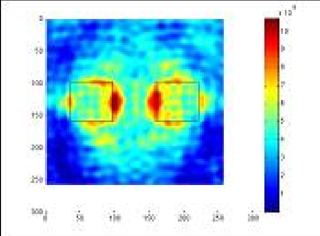
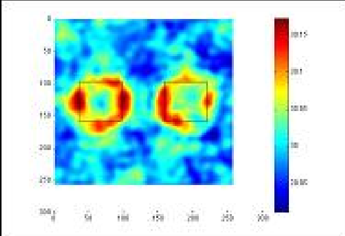
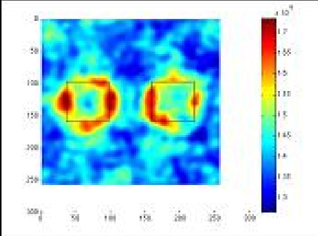
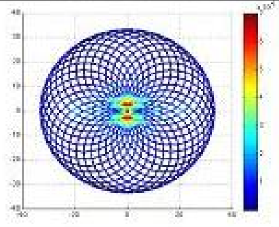
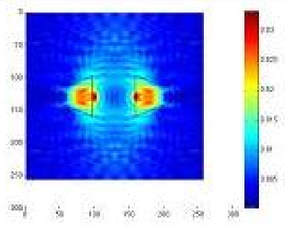
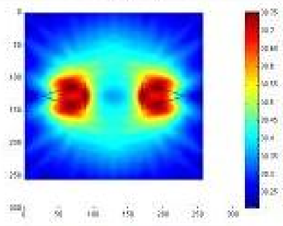
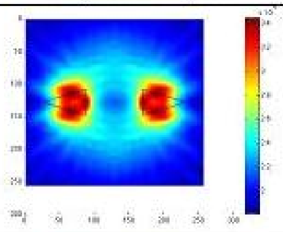
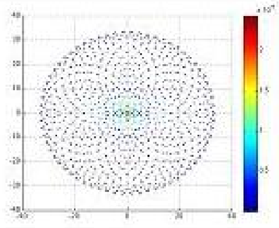
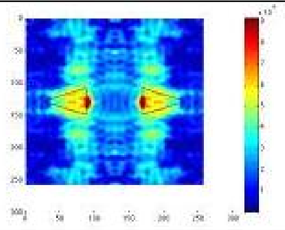
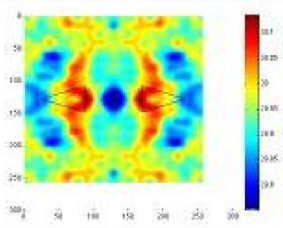
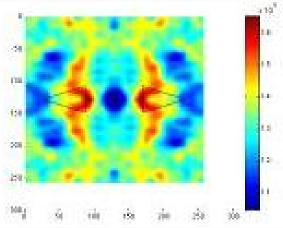
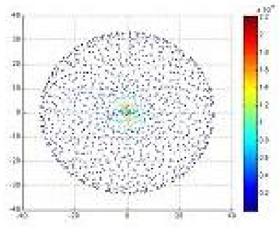
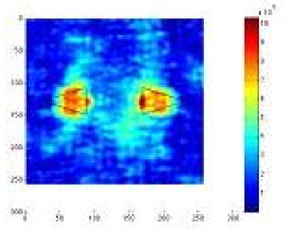
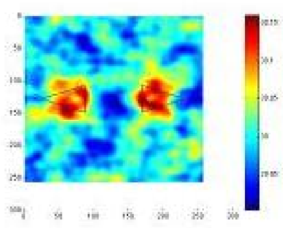
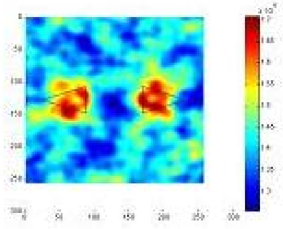
	Combined Ewald Circles	Born	Image Space	Cepstrum Space
er=1.9 Receivers = 360 Noise BW = 0				
er=1.9 Receivers = 30 Noise BW = 0				
er=1.9 Receivers = 30 Noise BW = 8				
er=1.9 Receivers = 36 Noise BW = 10				

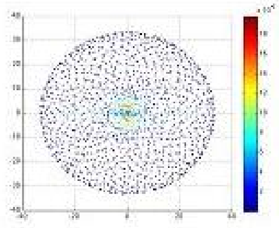
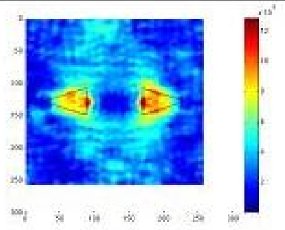
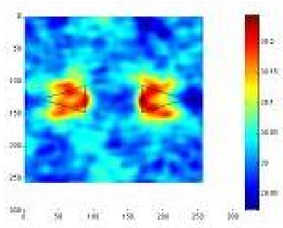
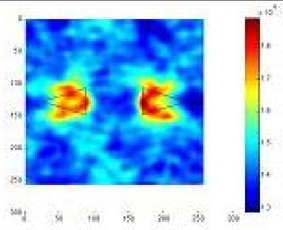
Table 6.39. A comparison of reconstructed images for a 2 triangle targets each with height = 3λ using the Born approximation, and processing sources independently and recombining in images space, and cepstrum space. 36 Sources are used with varying numbers of receivers equally and randomly spaced to show the effects of aliasing and anti-aliasing techniques.

	Combined Ewald Circles	Born	Image Space	Cepstrum Space

er=1.9
Receivers = 360
Noise BW = 0
				

er=1.9
Receivers = 36
Noise BW = 0
				

er=1.9
Receivers = 36
Noise BW = 10
				

er=1.9
Receivers = 45
Noise BW = 8
				

CHAPTER 7: CONCLUSIONS

The main objectives of this research as originally stated in the topic proposal were threefold as listed below:

1) To develop a systematic method to produce useful and accurate simulated model scattered field data from known structures to be used to characterize the performance of various inverse scattering imaging algorithms.

2) To use the data from the method in part 1) to understand existing imaging algorithms and to document their limitations and performance under various conditions.

3) To use the data from the method in part 1) to propose a new algorithm that extends existing methods and provides improved quantitative accuracy of the reconstructions.

The first objective was accomplished by creating a virtual test setup up in COMSOL® using finite element analysis to model scattered field data from a representative family of known targets. This data was then imported into MATLAB® for processing and investigating various algorithms and methods. It was demonstrated in Chapter 4 that this task was successfully completed and that the results from using this structured approach produced results that were comparable to the results from experimental data when adopting the exact same experimental conditions. This new method, once validated, proved to be indispensible in producing data for many different scattering conditions that were essential in addressing the other two objectives. With the use of this method and the data that it produced, existing methods and algorithms were studied and evaluated in a way that would be virtually impossible in a laboratory setting. Our algorithm for this will be useful in future research dealing with other scattering and imaging studies.

The data generated allowed objective number 2 to be addressed and the Born approximation and images generated assuming its validity were examined with great detail. In particular, the concept of the number of degrees of freedom associated with a scattering or imaging experiment that had been suggested in previous research [23] was put to the test. The concept of degrees of freedom for 2 dimensional imaging from scattered field data in terms of number of sources and number of receivers was studied for numerous conditions and targets. As demonstrated in Chapters 6.1 and 6.2, this measure of information transfer in a scattering experiment, in relation to the use of the Born approximation, does seem to be valid. The degrees of freedom for each simulated experiment were calculated and for each case this number could be regarded as a threshold identifying the point at which the reconstructed images achieved a “steady state” in their appearance. This is not to say that the images at this point were good or that they modeled the original target exactly, since this depends on the extent of the multiple scattering, but that these reconstructions did not appear to show any significant improvement when using additional sources or receivers beyond this point. The question of the quality or the appearance of each reconstructed image was altogether another issue.

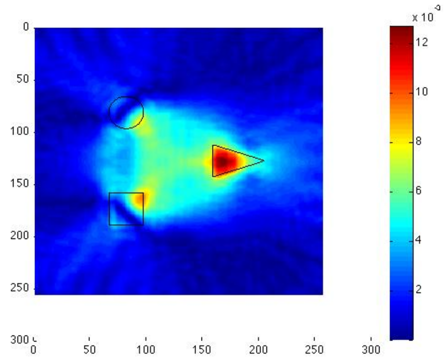
In examining the appearance of the image for circular targets, it was observed that there seem to be what appeared to be a cyclical pattern in the quality of the reconstruction as a function of increasing permittivity. We investigated this by recognizing that this type of target could be classified as a Lorenz-Mie scatterer and sought an explanation for this behavior from this point of view. Realizing that the efficiency factor of scattering near a resonance is characterized by the "Q" factor for Lorenz-Mie scatterers, this was calculated to find a possible explanation. This was done by performing a series of tests on Lorenz-Mie scattering cylinders with increasing permittivity and comparing the corresponding image quality with the plot of the "Q" factor for a Lorenz-Mie scatterer with the same parameters. It was demonstrated as a result of these experiments in Chapter 6.3 that the predicted resonances, and the "good" and "bad" reconstructions, aligned almost perfectly with the graph of the "Q" factor. This then is highly suggestive as a reasonable explanation for why strongly scattering cylinders, near a resonant scattering condition at which the scattering cross section is larger, appear to have a larger than expected area and a more uniform appearance.

The two factors of the number of degrees of freedom and the effects of resonance in determining the quality of a Born reconstructed image are extremely important. This now gives a new, and possibly more meaningful set of criteria for determining or predicting when it is satisfactory to implement and expect to be able to successfully interpret an image based on the Born approximation and Born related methods. The previous criterion of $k\mathrm{V}(\mathrm{r})\mathrm{a}<<1$ only claimed to predict whether a target was a weak scatterer or not, which can be terribly subjective and inconsistent in predicting performance. The new criterion gives a much more definitive predictor since the minimum number of degrees of freedom can be calculated precisely if $n_{\max}$ is known and the resonances might be able to be determined or even identified for a given target. The resonances can be easily predicted for a Mie scatterer, but this may not always be the case for more complex targets. However, resonances for specific target sets could be determined experimentally or via simulation and then possibly characterized by a closed form equation if one uses the structured method developed here. One could determine enough resonance points and then use some type of curve fitting algorithm to determine a resonance relationship much like the “Q” factor for the Mie scatterers. This might be possible theoretically, but for complex structures, our approach is more straightforward. This is a definite opportunity for more research and experimentation in the future. Assuming this relationship for a given target could be determined, this in conjunction with the degrees of freedom criterion should provide valuable tools in predicting the performance of Born approximation-based image reconstruction in the future, as well as investigating the properties of resonant scattering structures.

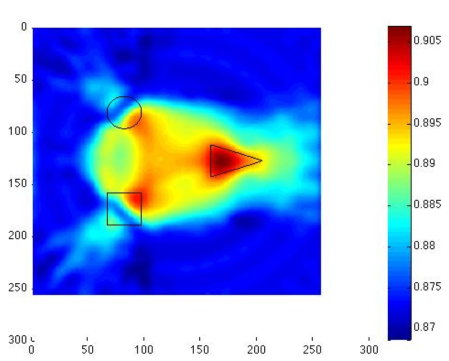
The third objective which was “to propose a new method that extends existing inverse scattering methods and provides improved quantitative accuracy” was pursued and accomplished in a number of ways. The first way this was accomplished was in the implementation in the code applying the cepstrum filtering method on an individual source basis followed by combining these processed data in either the image domain or the cepstrum domain. This is in lieu of the previous approach that simply applies the ceptrum filtering method to the Born approximation image obtained from the combined sources. As demonstrated in Section 6.4, this modified method shows definite improvement in the appearance of the image in relation to the boundaries of the target compared to previous methods. More specifically, the procedure that combined the sources, after processing, in the cepstrum domain showed the most promise in terms of more accurate image boundary definitions. The only difficulty with these methods initially was the quantitative accuracy in that the scales of the magnitudes for the improved reconstructed images did not seem to be reasonable. This issue was addressed with later research.

Also related to the third objective, the question of optimal filter design in the cepstrum domain was examined in Section 6.5. Through a series of tests included in this section it was determined that the optimum filter to be used is a Gaussian type filter centered at the origin using a sigma value equal to 10 (for the specific conditions of the set of targets considered here). Another striking discovery was that the absolute peak value of the filter plays a critical role in determining the magnitude of the reconstructed image produced by the new cepstrum method described above. In particular, it was shown that the magnitude of the output of the method that processes each source individually and recombined them in the cepstrum domain was inversely proportional to the magnitude of the peak of the Gaussian filter. After further examination, it was shown that the peak value of the Gaussian filter needed to be scaled by a factor of 2/(3Ns), where Ns is the number of sources, to make the magnitude of the reconstructed images be of the same range of the magnitude of the Born reconstructed image for the same data. It is not known or understood at this time why this scaling factor is needed. It could be a result of applying 3-dimensional techniques to a 2-dimensional problem or some variance of this. This is an opportunity for future research. In addition to these experiments on the filter characteristics in cepstrum space, a new approach was derived and evaluated for potential benefits as well. That new approach involved subtracting a weighted cepstrum of the incident field during the processing of the sources in cepstrum space. As explained in Section 5.3, this is done in an attempt to reduce or eliminate the $<\Psi>/R'$ term in the cepstrum domain representation of the signal as shown in Eq. 5.13. The results of applying this additional "filtering" step proved to be very promising in that it appears to improve the scale of the magnitude of the reconstructed image resulting from the "combined in cepstrum space" method. In fact, as shown in the results of Section 6.5, the peak of the scale is very near the "correct" value for the permittivity (or better yet, to the index of refraction) of the original target.

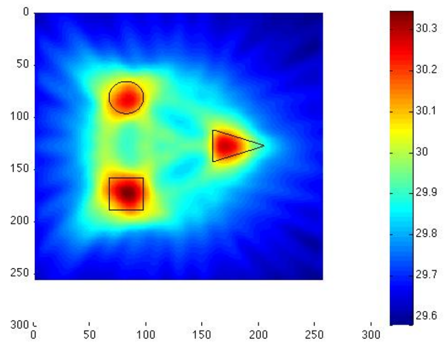
If we now take all of the improvements mentioned above, i.e., processing sources independently, optimized filtering, and subtracting the incident field in the cepstrum domain, we can apply this new improved method to a more complex target set and compare its results to those obtained by using previous methods to observe the improvements. This was done for a target set consisting of a circle with radius of λ, a square with sides of 2λ, and a triangle with a base of 2λ and a height of 3λ. This concluding test was run three times with the targets having a permittivity of 1.1, 1.5, and 1.9 respectively to observer how the methods performed in relation to each other. The results of this final set of experiments are shown in Figures 7.1, 7.2, and 7.3 below. The results shown in Figure 7.1 are extremely encouraging in that the improvement seen going from (a) to (d) is really quite surprising in relation to the target boundaries. The output for method (d) clearly shows that there are 3 distinct targets and makes valid attempts to show the extent of these boundaries. The scale of the magnitude for (d) is also very close in range for the index of refraction for the targets of 1.04.



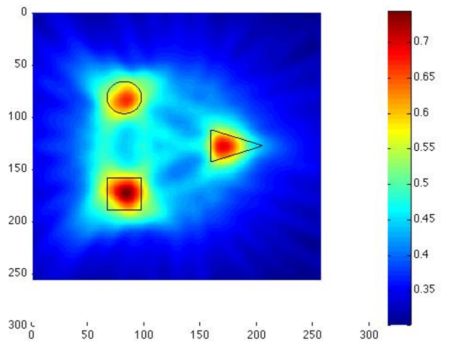
(a)



(b)



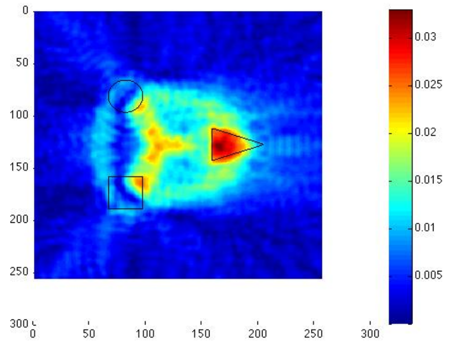
(c)



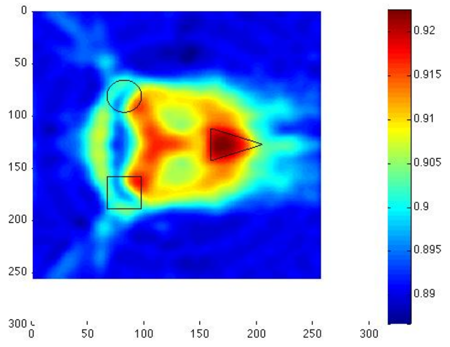
(d)

Figure 7.1. Comparison of reconstructed images from various method outputs for a target set consisting of a circle with radius of λ, a square with sides of 2λ, and a triangle with base of 2λ and height of 3λ. All targets have a permittivity of 1.1. The outputs shown above are from (a) Born approximation, (b) Cepstrum of image in (a) using algorithm developed in [6], (c) Cepstrum filtering of individual sources that are recombined in image space, and (d) Cepstrum filtering of individual sources that are recombined in image space and have a cepstrum version Ψ_{in} subtraced in cepstrum space.

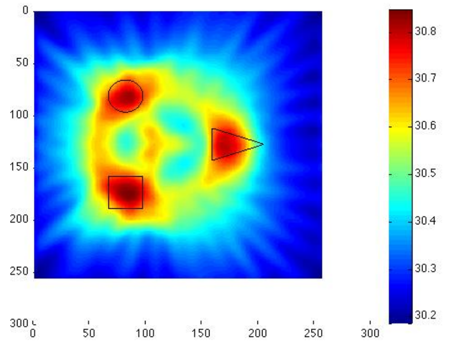
The results shown in Figure 7.2 are encouraging as well, though maybe not quite a striking as the results shown in Figure 7.1. Again, improvement is seen going from (a) to (d) in relation to the target boundaries. The output for method (d) clearly shows again that there are 3 distinct targets and makes a fair attempt to show the extent of these boundaries. The scale of the magnitude for (d) is again very close, actually almost exact in the range for the index of refraction for the targets which has an index value of 1.225. This is not as good of an overall performance as in Figure 7.1, but still very encouraging.



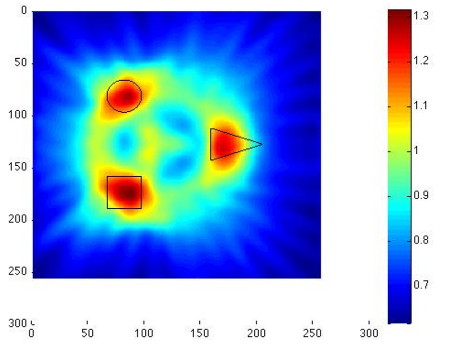
(a)



(b)



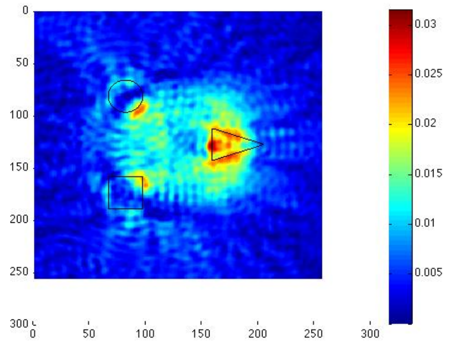
(c)



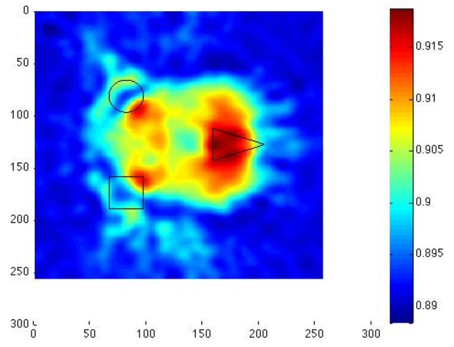
(d)

Figure 7.2. Comparison of reconstructed images from various method outputs for a target set consisting of a circle with radius of λ, a square with sides of 2λ, and a triangle with base of 2λ and height of 3λ. All targets have a permittivity of 1.5. The outputs shown above are from (a) Born approximation, (b) Cepstrum of image in (a) using algorithm developed in [6], (c) Cepstrum filtering of individual sources that are recombined in image space, and (d) Cepstrum filtering of individual sources that are recombined in image space and have a cepstrum version Ψ_{in} subtraced in cepstrum space.

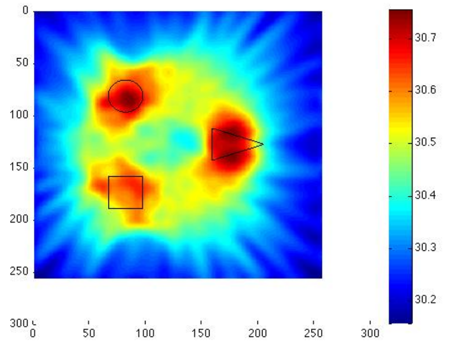
The results shown in Figure 7.3 show progressively poor overall performance as expected, but the image in (d) does show continued improvement compared to the other methods. As previously stated, improvement is seen going from method (a) to (d) especially in relation to the target boundaries. The output for method (d) does show again that there are 3 distinct targets but its attempt at defining the boundaries are fairly poor. The scale of the magnitude for (d) is also very close in range for the index of refraction for the targets which have an index of 1.378.



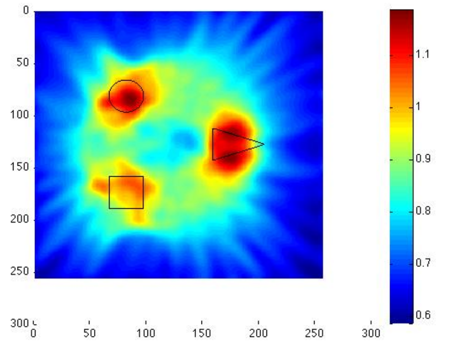
(a)



(b)



(c)



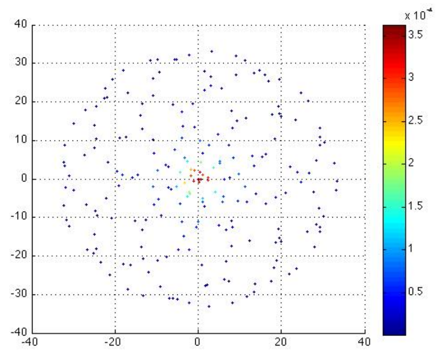
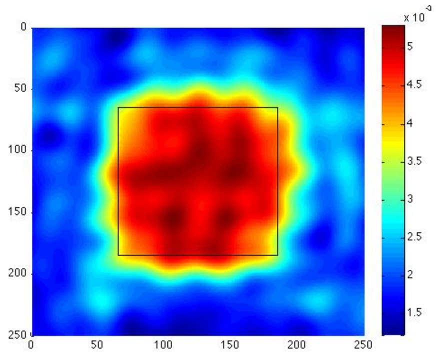
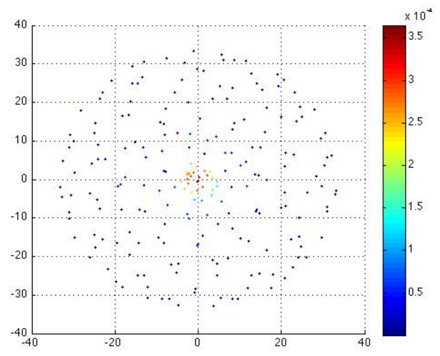
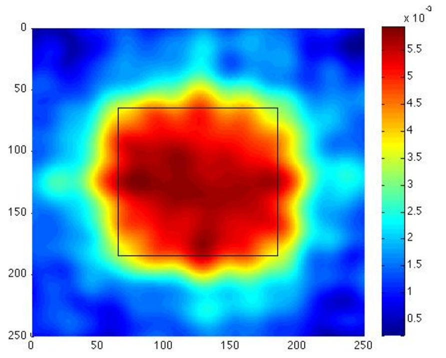
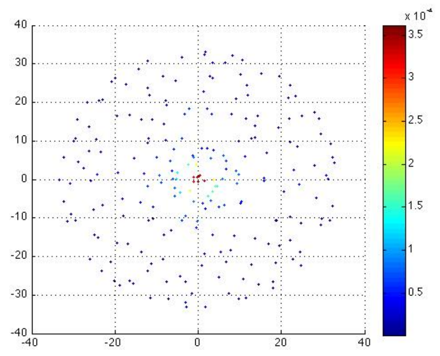
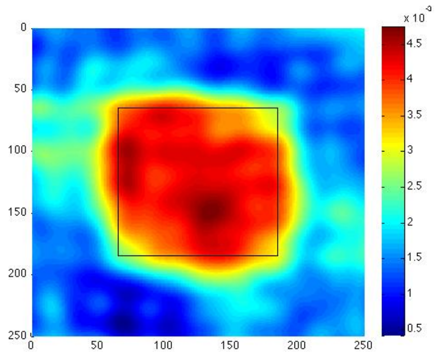
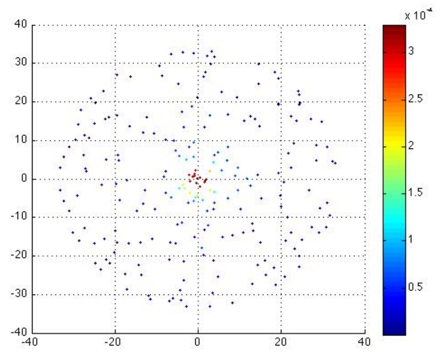
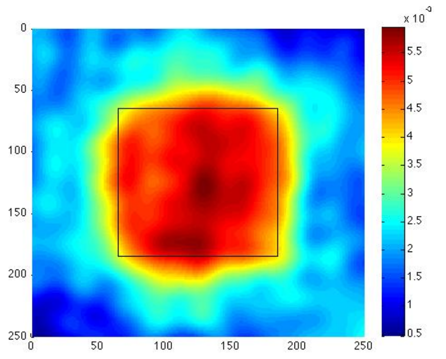
(d)

Figure 7.3. Comparison of reconstructed images from various method outputs for a target set consisting of a circle with radius of λ, a square with sides of 2λ, and a triangle with base of 2λ and height of 3λ. All targets have a permittivity of 1.9. The outputs shown above are from (a) Born approximation, (b) Cepstrum of image in (a) using algorithm developed in [6], (c) Cepstrum filtering of individual sources that are recombined in image space, and (d) Cepstrum filtering of individual sources that are recombined in image space and have a cepstrum version Ψ_{in} subtraced in cepstrum space.

There are two immediate opportunities for future research that I recommend. The first is fairly straightforward in that the issue of the scale of the new cepstrum method is still not fully resolved. Great improvements were incorporated and a better understanding was gained from this research, but it still is not fully understood. There appears to be some other nonlinear dependence on either the number of sources, the permittivity of the target or both. There may be other opportunities to remove or reduce the effects of the <Ψ> term in the cepstrum domain much like the one utilized in this research. There is also a possibility that some improvement may be gained from removing the reference and undoing the step to make the data causal after the filtering process has been completed in the cepstrum domain as well. These are all valid tasks to attempt to see if they have any affect or improvement on the process.

The final opportunity is a bit more complex than the others mentioned above. It was noted in this research that as experiments were being performed on the minimum required number of receivers in relation to the degrees of freedom that a randomness was injected into the receiver locations to address aliasing. In this process, there came a point where it was difficult to distinguish how many sources versus how many receivers there actually were. It began to look more like it was a matter of a total number of receiver points (and possibly their distribution) more than how many sources there were, i.e. that perhaps the more important metric is the number of receivers per source. This is illustrated in Table 7.1. For each of the set of Ewald circles and image reconstructions, the total number of receiver or data points is the same for each case, which is 216 total. Because they are randomly spaced (at least around their original locations), it is difficult to distinguish between the Ewald circle sets while simultaneously the resulting images appear to be very similar in appearance. This would seem to suggest that possibly the minimum requirement is not so much a function of the number of sources and/or the number of receivers required, but more the total number of receiver or data points randomly spaced to fill the *k* space area. If this is the case then it is possible that the minimum degrees of freedom may not be a function of either sources or receivers by themselves, but an expression involving a combination of the two.

Table 7.1. Ewald circle sets with accompanying Born reconstructed images for a series of Source/Receiver combinations all totaling 216 receivers (or data points) with random spacing applied to the receiver locations about their original equispaced locations. Target used is a square with sides equal to 2λ and a permittivity of 1.5.

Src = 9 / Rec = 24		
Src = 12 / Rec = 18		
Src = 18 / Rec = 12		
Src = 36 / Rec = 6		

The first intuitive response could be that it may be simply a product of the two or Ns*Nr, which would simply be N^2. This could be tested quickly using the model previously used. For this scenario Ns/Nr is calculated to be 4.9 so for these conditions we could declare that N=5. If in fact N^2 is the relationship we should be able to satisfy this by setting Ns=6 and Nr=5. The results are shown in Figure 7.4 compared to N = 12,960.

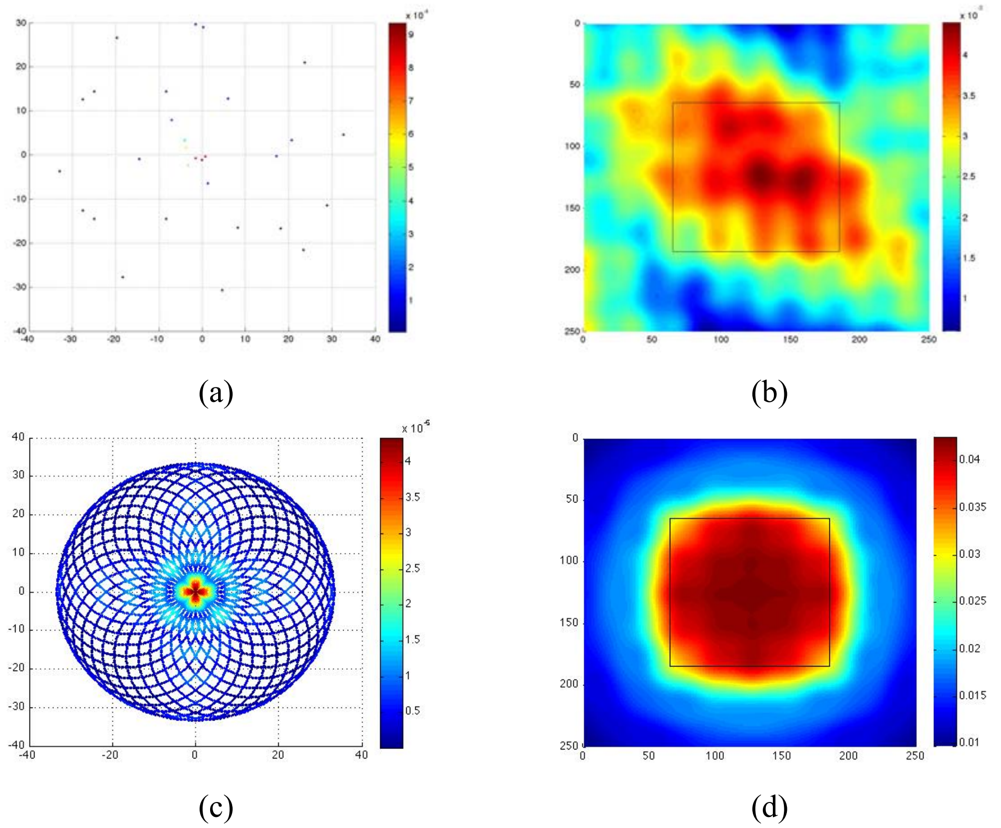
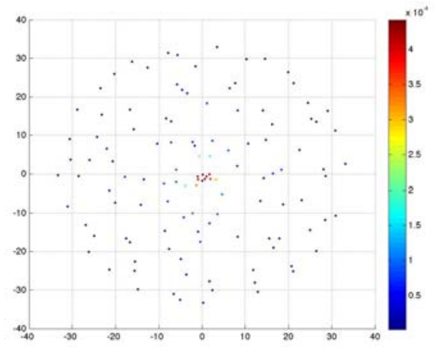
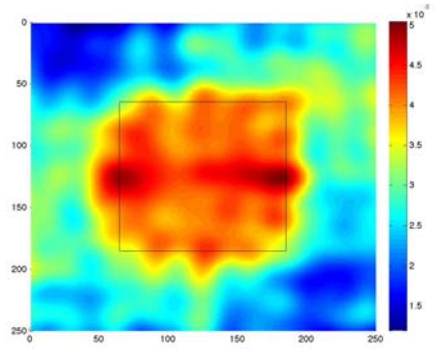


Figure 7.4. (a) Ewald circle using 6 sources and 5 receivers randomly spaced. (b) Resulting reconstructed image from (a). (c) Ewald circle using 36 source and 360 receivers equally spaced for comparison with (a) and (b). (d) Resulting image from (c).

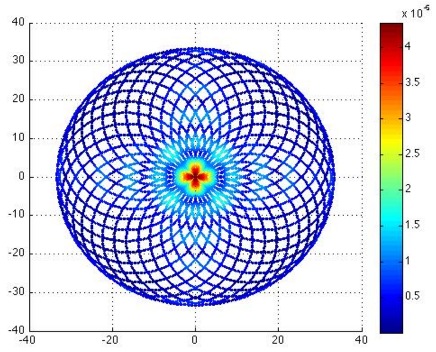
Reviewing the results in Figure 7.4 it does not appear that the "overall" number of degrees of freedom have been satisfied. The combined requirement for the sources and receivers, if this is in fact the case, may be as simple as applying a "Nyquist-Shannon theorem" type approach which might look something like (2Ns)*(2Nr) = $4N^2$. Applying this criterion to the square model used before, would mean that the minimum number of degrees of freedom or data points required would be N = $4(5)^2$ = 100. This can be achieved (or exceeded) in the square model by using 12 sources and 10 receivers randomly spaced. This too was implemented with the results shown in Figure 7.5, again along with the case for N = 12,960.



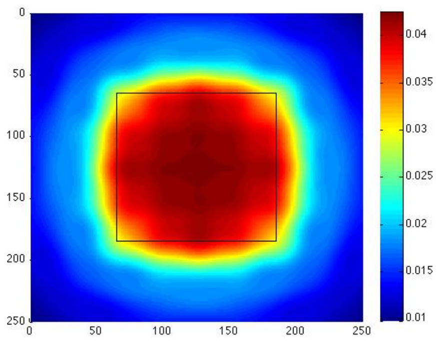
(a)



(b)



(c)



(d)

Figure 7.5. (a) Ewald circle using 12 sources and 10 receivers randomly spaced. (b) Resulting reconstructed image from (a). (c) Ewald circle using 36 source and 360 receivers equally spaced for comparison with (a) and (b). (d) Resulting images from (c).

The image obtained from using 12 sources and 10 receivers shown in Figure 7.4(b) does seem to show potential that the minimum degrees of freedom have been met when compared to the image in Figure 7.4(d) for the case with 36 sources and 360 receivers. It is entirely possible that the minimum has been met in this case. There simply is not enough evidence to say conclusively if this describes the overall degrees of freedom relationship. It does raise an interesting thought in the fact that there does seem to be more work here to be done to properly relate the concept of the number of degrees of freedom requirements to the number of sources and the number of receivers required. One could possibly determine a relationship between them to form a unified degrees of freedom relationship for image reconstruction (possibly utilizing randomly spaced source and receiver locations to provide data points to optimally cover *k* space).

It is truly my hope that the research presented in this dissertation will be of use in future endeavors and research and that continued progress will be made in understanding imaging from inverting scattered field data. To assist those wishing to advance this research, and to offer as validation for the research done in this paper, the master MATLAB code has been included in Appendix A for the readers examination and implementation if so desired.

REFERENCES

1. McGahan R V and Kleinman R E, “Special session on image reconstruction using real data”, *IEEE Antenna Propagat. Mag*.38(3) 39-59, 1996.

2. McGahan R V and Kleinman R E, “Second annual special session on image reconstruction using real data”, *IEEE Antenna Propagat. Mag*. 39(2) 7-32, 1997.

3. McGahan R V and Kleinman R E, “The third annual special session on image reconstruction using real data, part1”, *IEEE Antenna Propagat. Mag*. 41(1) 34-51, 1999.

4. McGahan R V and Kleinman R E, “The third annual special session on image reconstruction using real data, part2”, *IEEE Antenna Propagat. Mag*. 41(2) 20-40, 1999.

5. Christelle Ayraud, Jean-Michel Geffrin, Pierre Sabouroux, Kamal Belkebir and Marc Saillard, “Laboratory controlled data for validating inversion algorithms”, Institut Fresnel, UMR-CNR 6133, France.

6. Shahid U., ‘Signal Processing Based Method for Solving Inverse Scattering Problems’, Ph.D. Dissertation, UNC Charlotte, 2009

7. Avinash C.Kak and Malcom Slaney, *Principles of computerized tomographic imaging*, IEEE press, New York, 1988.

8. E. Wolf, “Three-dimensional structure determination of semi-transparent objects from holographic data,” *Optics Comm*., 1, 1969.

9. W.C.Chew, *Waves and fields in inhomogeneous media*, IEEE Press, Piscataway, 1995.

10. F.C.Lin and M.A.Fiddy, “Image estimation from scattered field data,” *Int. J. Imag. Sys. Technol*., vol.2, pp.76-95, 1990.

11. P.M. Morse and H. Feshbach, *Methods of theoretical physics*, New York: McGraw-Hill, 1953.

12. Angela Maria Darling, Digital object reconstruction from limited data incorporating prior information, Thesis, University of London, 1984.

13. J. Li, X. Wang and T. Wang, “On the validity of the Born approximation”, Prog. In Electromagnetics Research, 107, pp219-237, 2010.

14. A. G. Ramm, “Is the Born approximation good for solving the inverse scattering problem when the potential is small?,” *J. Math. Anal. Appl.*, vol. 147, 480-485, 1990.

15. A. Ishimaru, *Wave propagation and scattering in random media*, Academic press, New York, vol. 1, 1978.

16. M. A. Fiddy, M. Testorf and U. Shahid, “Minimum-phase-based inverse scattering method applied to IPS008,” *SPIE*, vol. 5562, pp 188–195, 2004.

17. Donald G. Childers. David P. Skinner and Robert C. Kemerait, “The cepstrum: A Guide to processing,” *IEEE proc.*, vol. 65, No. 10, 1977.

18. D. Raghuramireddy and R. Unbehauen, “The Two-Dimensional Differential Cepstrum,” *IEEE transactions on acoustics, speech and signal processing*, vol. ASSP-33, No. 4, 1985.

19. Fiddy M A and Shahid U, “Minimum phase and zero distributions in 2D,” *Proc. SPIE*, vol. 5202, pp 201–208, 2003.

20. Jin, J, The *Finite Element Method in Electromagnetics*, IEEE Press, New York, 2002.

21. Silvester P and Ferrari R, Finite Elements for Electrical Engineers, Cabridge University Press, New York,1996

22. U Shahid, M A Fiddy and M E Testorf, “Inversion of strongly scattered data: shape and permittivity recovery”, SPIE 7076, 707606, 2008, Charlotte, 2009.

23. David A. B. Miller, “Fundamental limit for optical components,” *J. Opt. Soc. Am. B*, vol. 24, No.10, 2007.

24. Kasap, S. O., *Optoelectronics devices and photonics: principles and practices*, Prentice-Hall Inc., New Jersey, 2001.

25. Jones, W. B., *Introduction to optical fiber communication system*, Holt, Rinehart, & Winston, Inc., New York, 1988.

26. Mishchenko, M. I., and Travis, “Gustav Mie and the Evolving Discipline of Electromagnetic Scattering by Particles,” American Meteorological Society, pp 1853-1861, December, 2008.

27. Hahn, D. W. “Light Scattering Theory”, University of Florida, July, 2009.

28. Dan E. Dudgeon and Russel M. Mersereau, *Multidimensional digital signal processing*, Prentice-Hall, New Jersey, 1984.

29. A. V. Oppenheim, R. W. Schafer and J. R. Buck, *Discrete-time signal processing*, Prentice-Hall, 1999.

30. Bogert, B. P., Healy, M. J. R., and Tukey, J. W., “The Quefrency Analysis of Time Series for Echoes: Cepstrum, Psuedo-Autocovariance, Cross-cepstrum and Saphe Cracking”, Proceedings of the Symposium on Time Series Analysis, M Rosenblat, Ed., Wiley, NY, 1963, pp 209-243.

31. R. E. Burge, M. A. Fiddy, A. H. Greenaway and G. Ross, “The phase problem,” *Proc. Roy. Soc. Lond.*, A350, pp 191-212, 1976.

32. Fiddy M A and Shahid U, “Minimum phase and zero distributions in 2D,” *Proc. SPIE*, vol. 5202, pp 201–208, 2003.

33. U Shahid, M Testorf, and M A Fiddy, “Minimum-phase-based inverse scattering algorithm,” *Inverse Problems*, vol. 21, pp 1 – 13, 2005.

34. E.C. Titchmarsh, *Introduction to the theory of Fourier integrals*, 2nd Edn. Oxford University Press, 1948.

35. Walstra, P. "Approximation formulae for the light scattering coefficient of dielectric spheres". *British Journal of Applied Physics* **15**: 1545. 1964

36. H. C. van de Hulst, *Light scattering by small particles*, New York, John Wiley and Sons, 1957

37. Pijanka, Jacek Klaudiusz; Kohler, Achim; Yang, Ying; Dumas, Paul; Chio-Srichan, Sirinart; Manfait, Michel; Sockalingum, Ganesh Dhruvananda; Sulé-Suso, Josep. "Spectroscopic signatures of single, isolated cancer cell nuclei using synchrotron infrared microscopy". *The Analyst* 134 (6): 1176, 2009.

38. Mohlenhoff, B; Romeo, M; Diem, M; Wood, BR. "Mie-Type Scattering and Non-Beer-Lambert Absorption Behavior of Human Cells in Infrared Microspectroscopy". *Biophysical Journal* 88 (5): 3635, 2005.

39. Churchill, R. V., Brown, J. W., *Complex Variables and Applications*, 5th Edn. McGraw-Hill, Inc., 1990

40. Gonzalez, R. C., Woods, R. E., *Digital Image Processing*, Addison-Wesley Publishing Company, Inc., 1992.

41. Jackson, L. B., *Digital Filters and Signal Processing*, 2nd Edn., Kluwer Academic Pblishers, 1991.

42. Lustig, M., Donoho, D., Pauly, J. M. “Sparse MRI: The application of Compressed Sensing for Rapid MR Imaging”, Magnetic Resonance in Medicine, 2010

43. E. J. Candes and M. B. Wakin, “An Introduction to Compressive Sampling”, IEEE Signal Processing Magazine, pp21-30, March 2008

44. E J. Candes and J Romberg, “Practical Signal Recovery from Random Projections”, DRAFT, January 25, 2005

APPENDIX A: MATLAB PROGRAM CODE

```
clear all;close all;i = sqrt(-1.0);c0 = 3e8;
frequ = 5;frequency = 1e9.*(frequ);lambda=c0/frequency;
er = input('\nEnter target permitivity value (1.1,1.2,1.3,1.4,1.5,1.6,1.7,1.8,1.9): ');
switch er
   case 1.1
      eri = 1;
   case 1.2
      eri = 2;
   case 1.3
      eri = 3;
   case 1.4
      eri = 4;
   case 1.5
      eri = 5;
   case 1.6
      eri = 6;
   case 1.7
      eri = 7;
   case 1.8
      eri = 8;
   case 1.9
      eri = 9;
   otherwise
      disp('ERROR ... Permitivity value out of range!');
      break;
end
Ns = input('\nEnter number of Sources (1,2,3,4,6,9,12,18,36): ');
Nr = input('\nEnter number of Receivers
(2,4,6,8,10,12,15,18,20,24,30,36,45,60,90,120,180,360): ');
Nf = 1;NrMAX = 360/Nr;
BWn = input(sprintf('\nEnter Jitter Bandwidth (Max = %d, Default = 0): ', NrMAX));
if isempty(BWn)
   BWn=0;
end
Sigma = input('\nEnter value for Sigma (Defult = 10): ');
if isempty(Sigma)
   Sigma=10;
end
DefaultPeak = 2/(3*Ns);
Peak = input(sprintf('\nEnter Filter Multiplier (Defult = %g): ', DefaultPeak));
if isempty(Peak)
   Peak=DefaultPeak;
end
```

```
FiltMult=1;
BackMult = input('\nEnter Background Multiplier (Defult = .9): ');
if isempty(FiltMult)
   BackMult=.9;
end
RefMult = input('\nEnter Reference Multiplier (Default = 1): ');
if isempty(RefMult)
   RefMult=1;
end
RefBias = input('\nEnter Reference Bias (Default = 0): ');
if isempty(RefBias)
   RefBias=0;
end
ZMult = input('\nEnter Data Multiplier (Default = 1): ');
if isempty(ZMult)
   ZMult=1;
end
ZBias = input('\nEnter Data Bias (Default = 0): ');
if isempty(ZBias)
   ZBias=0;
end
ImD = input('\nEnter Image Maximum Dimension in Meters (Default = 1): ');
if isempty(ImD)
   ImD=1;
end
ImR = input('\nEnter Image Resolution (Default = 256): ');
if isempty(ImR)
   ImR=256;
end
ImM = input('\nEnter Independant Source Image Multiplier (Default = 1): ');
if isempty(ImM)
   ImM=1;
end
showEwald = input('\nShow Individual Ewald Images? (Enter 1 for Yes, Default = No):
');
if isempty(showEwald)
   showEwald = 0;
end
showBorn = input('\nShow Individual Born Images? (Enter 1 for Yes, Default = No): ');
if isempty(showBorn)
   showBorn = 0;
end
showCep = input('\nShow Individual Cepstrum Images? (Enter 1 for Yes, Default = No):
');
if isempty(showCep)
   showCep = 0;
```

```
end
 [fb, msg] = fopen ('Ei0536360.dat', 'rt');
if fb == -1
   disp (msg)
   return;
end
[ff, msg] = fopen ('Ez.36.360.dat', 'rt');[fx, msg] = fopen ('X.dat', 'rt');
if fx == -1
   disp (msg)
   return;
end
[fy, msg] = fopen ('Y.dat', 'rt');
if fy == -1
   disp (msg)
   return;
end
fprintf('\nLoading Data Files .'); [J, cj] = fscanf(fx, '%d', [1]);[J, cj] = fscanf(fy, '%d', [1]);
[X, cx] = fscanf(fx, '%g', [360, 1]);[Y, cy] = fscanf(fy, '%g', [360, 1]);
fprintf('.');phirad = zeros(360,1);phideg = zeros(360,1);rdeg = zeros(360,1);clear J;
for pp = 1:360
   if X(pp,1) > 0 && Y(pp,1) >= 0
      phirad(pp,1) = atan(abs(Y(pp,1))/abs(X(pp,1)));
      phideg(pp,1) = phirad(pp,1)*(180/pi);rdeg(pp,1) = phideg(pp,1);
   end
   if X(pp,1) <= 0 && Y(pp,1) > 0
      phirad(pp,1) = atan(abs(Y(pp,1))/abs(X(pp,1)));
      phideg(pp,1) = phirad(pp,1)*(180/pi);rdeg(pp,1) = 180-phideg(pp,1);
   end
   if X(pp,1) >= 0 && Y(pp,1) < 0
      phirad(pp,1) = atan(abs(Y(pp,1))/abs(X(pp,1)));
      phideg(pp,1) = phirad(pp,1)*(180/pi);rdeg(pp,1) = 360-phideg(pp,1);
   end
   if X(pp,1) < 0 && Y(pp,1) <= 0
      phirad(pp,1) = atan(abs(Y(pp,1))/abs(X(pp,1)));
      phideg(pp,1) = phirad(pp,1)*(180/pi);rdeg(pp,1) = 180+phideg(pp,1);
   end
end
fprintf('.');
 [S1, cj] = fscanf(fb, '%d', [1]);[B1, cb] = fscanf(fb, '%g %g', [2, 360]);S(1,1) = 180;
[S2, cj] = fscanf(fb, '%d', [1]);[B2, cb] = fscanf(fb, '%g %g', [2, 360]);S(2,1) = 170;
[S3, cj] = fscanf(fb, '%d', [1]);[B3, cb] = fscanf(fb, '%g %g', [2, 360]);S(3,1) = 160;
[S4, cj] = fscanf(fb, '%d', [1]);[B4, cb] = fscanf(fb, '%g %g', [2, 360]);S(4,1) = 150;
[S5, cj] = fscanf(fb, '%d', [1]);[B5, cb] = fscanf(fb, '%g %g', [2, 360]);S(5,1) = 140;
[S6, cj] = fscanf(fb, '%d', [1]);[B6, cb] = fscanf(fb, '%g %g', [2, 360]);S(6,1) = 130;
[S7, cj] = fscanf(fb, '%d', [1]);[B7, cb] = fscanf(fb, '%g %g', [2, 360]);S(7,1) = 120;
[S8, cj] = fscanf(fb, '%d', [1]);[B8, cb] = fscanf(fb, '%g %g', [2, 360]);S(8,1) = 110;
```

```
[S9, cj] = fscanf(fb, '%d', [1]);[B9, cb] = fscanf(fb, '%g %g', [2, 360]);S(9,1) = 100;
[S10, cj] = fscanf(fb, '%d', [1]);[B10, cb] = fscanf(fb, '%g %g', [2, 360]);S(10,1) = 90;
[S11, cj] = fscanf(fb, '%d', [1]);[B11, cb] = fscanf(fb, '%g %g', [2, 360]);S(11,1) = 80;
[S12, cj] = fscanf(fb, '%d', [1]);[B12, cb] = fscanf(fb, '%g %g', [2, 360]);S(12,1) = 70;
[S13, cj] = fscanf(fb, '%d', [1]);[B13, cb] = fscanf(fb, '%g %g', [2, 360]);S(13,1) = 60;
[S14, cj] = fscanf(fb, '%d', [1]);[B14, cb] = fscanf(fb, '%g %g', [2, 360]);S(14,1) = 50;
[S15, cj] = fscanf(fb, '%d', [1]);[B15, cb] = fscanf(fb, '%g %g', [2, 360]);S(15,1) = 40;
[S16, cj] = fscanf(fb, '%d', [1]);[B16, cb] = fscanf(fb, '%g %g', [2, 360]);S(16,1) = 30;
[S17, cj] = fscanf(fb, '%d', [1]);[B17, cb] = fscanf(fb, '%g %g', [2, 360]);S(17,1) = 20;
[S18, cj] = fscanf(fb, '%d', [1]);[B18, cb] = fscanf(fb, '%g %g', [2, 360]);S(18,1) = 10;
[S19, cj] = fscanf(fb, '%d', [1]);[B19, cb] = fscanf(fb, '%g %g', [2, 360]);S(19,1) = 0;
[S20, cj] = fscanf(fb, '%d', [1]);[B20, cb] = fscanf(fb, '%g %g', [2, 360]);S(20,1) = 350;
[S21, cj] = fscanf(fb, '%d', [1]);[B21, cb] = fscanf(fb, '%g %g', [2, 360]);S(21,1) = 340;
[S22, cj] = fscanf(fb, '%d', [1]);[B22, cb] = fscanf(fb, '%g %g', [2, 360]);S(22,1) = 330;
[S23, cj] = fscanf(fb, '%d', [1]);[B23, cb] = fscanf(fb, '%g %g', [2, 360]);S(23,1) = 320;
[S24, cj] = fscanf(fb, '%d', [1]);[B24, cb] = fscanf(fb, '%g %g', [2, 360]);S(24,1) = 310;
[S25, cj] = fscanf(fb, '%d', [1]);[B25, cb] = fscanf(fb, '%g %g', [2, 360]);S(25,1) = 300;
[S26, cj] = fscanf(fb, '%d', [1]);[B26, cb] = fscanf(fb, '%g %g', [2, 360]);S(26,1) = 290;
[S27, cj] = fscanf(fb, '%d', [1]);[B27, cb] = fscanf(fb, '%g %g', [2, 360]);S(27,1) = 280;
[S28, cj] = fscanf(fb, '%d', [1]);[B28, cb] = fscanf(fb, '%g %g', [2, 360]);S(28,1) = 270;
[S29, cj] = fscanf(fb, '%d', [1]);[B29, cb] = fscanf(fb, '%g %g', [2, 360]);S(29,1) = 260;
[S30, cj] = fscanf(fb, '%d', [1]);[B30, cb] = fscanf(fb, '%g %g', [2, 360]);S(30,1) = 250;
[S31, cj] = fscanf(fb, '%d', [1]);[B31, cb] = fscanf(fb, '%g %g', [2, 360]);S(31,1) = 240;
[S32, cj] = fscanf(fb, '%d', [1]);[B32, cb] = fscanf(fb, '%g %g', [2, 360]);S(32,1) = 230;
[S33, cj] = fscanf(fb, '%d', [1]);[B33, cb] = fscanf(fb, '%g %g', [2, 360]);S(33,1) = 220;
[S34, cj] = fscanf(fb, '%d', [1]);[B34, cb] = fscanf(fb, '%g %g', [2, 360]);S(34,1) = 210;
[S35, cj] = fscanf(fb, '%d', [1]);[B35, cb] = fscanf(fb, '%g %g', [2, 360]);S(35,1) = 200;
[S36, cj] = fscanf(fb, '%d', [1]);[B36, cb] = fscanf(fb, '%g %g', [2, 360]);S(36,1) = 190;
B1 = B1';B2 = B2';B3 = B3';B4 = B4';B5 = B5';B6 = B6';B7 = B7';B8 = B8';
B9 = B9';B10 = B10';B11 = B11';B12 = B12';B13 = B13';B14 = B14';B15 = B15';
B16 = B16';B17 = B17';B18 = B18';B19 = B19';B20 = B20';B21 = B21';B22 = B22';
B23 = B23';B24 = B24';B25 = B25';B26 = B26';B27 = B27';B28 = B28';B29 = B29';
B30 = B30';B31 = B31';B32 = B32';B33 = B33';B34 = B34';B35 = B35';B36 = B36';
Bz = [];Bz = [Bz, B1];Bz = [Bz, B2];Bz = [Bz, B3];Bz = [Bz, B4];Bz = [Bz, B5];
Bz = [Bz, B6];Bz = [Bz, B7];Bz = [Bz, B8];Bz = [Bz, B9];Bz = [Bz, B10];
Bz = [Bz, B11];Bz = [Bz, B12];Bz = [Bz, B13];Bz = [Bz, B14];Bz = [Bz, B15];
Bz = [Bz, B16];Bz = [Bz, B17];Bz = [Bz, B18];Bz = [Bz, B19];Bz = [Bz, B20];
Bz = [Bz, B21];Bz = [Bz, B22];Bz = [Bz, B23];Bz = [Bz, B24];Bz = [Bz, B25];
Bz = [Bz, B26];Bz = [Bz, B27];Bz = [Bz, B28];Bz = [Bz, B29];Bz = [Bz, B30];
Bz = [Bz, B31];Bz = [Bz, B32];Bz = [Bz, B33];Bz = [Bz, B34];Bz = [Bz, B35];
Bz = [Bz, B36];Bz = [Bz, rdeg];
clear B1;clear B2;clear B3;clear B4;clear B5;clear B6;clear B7;clear B8;clear B9;
clear B10;clear B11;clear B12;clear B13;clear B14;clear B15;clear B16;clear B17;
clear B18;clear B19;clear B20;clear B21;clear B22;clear B23;clear B24;clear B25;
clear B26;clear B27;clear B28;clear B29;clear B30;clear B31;clear B32;clear B33;
clear B34;clear B35;clear B36;
```

```
Bz = sortrows(Bz, [73]);fprintf('.');
[S1, cj] = fscanf(ff, '%d', [1]);[F105, cf] = fscanf(ff, '%g %g', [2, 360]);
[F115, cf] = fscanf(ff, '%g %g', [2, 360]);[F120, cf] = fscanf(ff, '%g %g', [2, 360]);
[F125, cf] = fscanf(ff, '%g %g', [2, 360]);[F130, cf] = fscanf(ff, '%g %g', [2, 360]);
[F135, cf] = fscanf(ff, '%g %g', [2, 360]);[F140, cf] = fscanf(ff, '%g %g', [2, 360]);
[F145, cf] = fscanf(ff, '%g %g', [2, 360]);[F150, cf] = fscanf(ff, '%g %g', [2, 360]);
[S2, cj] = fscanf(ff, '%d', [1]);[F205, cf] = fscanf(ff, '%g %g', [2, 360]);
[F215, cf] = fscanf(ff, '%g %g', [2, 360]);[F220, cf] = fscanf(ff, '%g %g', [2, 360]);
[F225, cf] = fscanf(ff, '%g %g', [2, 360]);[F230, cf] = fscanf(ff, '%g %g', [2, 360]);
[F235, cf] = fscanf(ff, '%g %g', [2, 360]);[F240, cf] = fscanf(ff, '%g %g', [2, 360]);
[F245, cf] = fscanf(ff, '%g %g', [2, 360]);[F250, cf] = fscanf(ff, '%g %g', [2, 360]);
[S3, cj] = fscanf(ff, '%d', [1]);[F305, cf] = fscanf(ff, '%g %g', [2, 360]);
[F315, cf] = fscanf(ff, '%g %g', [2, 360]);[F320, cf] = fscanf(ff, '%g %g', [2, 360]);
[F325, cf] = fscanf(ff, '%g %g', [2, 360]);[F330, cf] = fscanf(ff, '%g %g', [2, 360]);
[F335, cf] = fscanf(ff, '%g %g', [2, 360]);[F340, cf] = fscanf(ff, '%g %g', [2, 360]);
[F345, cf] = fscanf(ff, '%g %g', [2, 360]);[F350, cf] = fscanf(ff, '%g %g', [2, 360]);
[S4, cj] = fscanf(ff, '%d', [1]);[F405, cf] = fscanf(ff, '%g %g', [2, 360]);
[F415, cf] = fscanf(ff, '%g %g', [2, 360]);[F420, cf] = fscanf(ff, '%g %g', [2, 360]);
[F425, cf] = fscanf(ff, '%g %g', [2, 360]);[F430, cf] = fscanf(ff, '%g %g', [2, 360]);
[F435, cf] = fscanf(ff, '%g %g', [2, 360]);[F440, cf] = fscanf(ff, '%g %g', [2, 360]);
[F445, cf] = fscanf(ff, '%g %g', [2, 360]);[F450, cf] = fscanf(ff, '%g %g', [2, 360]);
[S5, cj] = fscanf(ff, '%d', [1]);[F505, cf] = fscanf(ff, '%g %g', [2, 360]);
[F515, cf] = fscanf(ff, '%g %g', [2, 360]);[F520, cf] = fscanf(ff, '%g %g', [2, 360]);
[F525, cf] = fscanf(ff, '%g %g', [2, 360]);[F530, cf] = fscanf(ff, '%g %g', [2, 360]);
[F535, cf] = fscanf(ff, '%g %g', [2, 360]);[F540, cf] = fscanf(ff, '%g %g', [2, 360]);
[F545, cf] = fscanf(ff, '%g %g', [2, 360]);[F550, cf] = fscanf(ff, '%g %g', [2, 360]);
[S6, cj] = fscanf(ff, '%d', [1]);[F605, cf] = fscanf(ff, '%g %g', [2, 360]);
[F615, cf] = fscanf(ff, '%g %g', [2, 360]);[F620, cf] = fscanf(ff, '%g %g', [2, 360]);
[F625, cf] = fscanf(ff, '%g %g', [2, 360]);[F630, cf] = fscanf(ff, '%g %g', [2, 360]);
[F635, cf] = fscanf(ff, '%g %g', [2, 360]);[F640, cf] = fscanf(ff, '%g %g', [2, 360]);
[F645, cf] = fscanf(ff, '%g %g', [2, 360]);[F650, cf] = fscanf(ff, '%g %g', [2, 360]);
[S7, cj] = fscanf(ff, '%d', [1]);[F705, cf] = fscanf(ff, '%g %g', [2, 360]);
[F715, cf] = fscanf(ff, '%g %g', [2, 360]);[F720, cf] = fscanf(ff, '%g %g', [2, 360]);
[F725, cf] = fscanf(ff, '%g %g', [2, 360]);[F730, cf] = fscanf(ff, '%g %g', [2, 360]);
[F735, cf] = fscanf(ff, '%g %g', [2, 360]);[F740, cf] = fscanf(ff, '%g %g', [2, 360]);
[F745, cf] = fscanf(ff, '%g %g', [2, 360]);[F750, cf] = fscanf(ff, '%g %g', [2, 360]);
 [S8, cj] = fscanf(ff, '%d', [1]);[F805, cf] = fscanf(ff, '%g %g', [2, 360]);
[F815, cf] = fscanf(ff, '%g %g', [2, 360]);[F820, cf] = fscanf(ff, '%g %g', [2, 360]);
[F825, cf] = fscanf(ff, '%g %g', [2, 360]);[F830, cf] = fscanf(ff, '%g %g', [2, 360]);
[F835, cf] = fscanf(ff, '%g %g', [2, 360]);[F840, cf] = fscanf(ff, '%g %g', [2, 360]);
[F845, cf] = fscanf(ff, '%g %g', [2, 360]);[F850, cf] = fscanf(ff, '%g %g', [2, 360]);
[S9, cj] = fscanf(ff, '%d', [1]);[F905, cf] = fscanf(ff, '%g %g', [2, 360]);
[F915, cf] = fscanf(ff, '%g %g', [2, 360]);[F920, cf] = fscanf(ff, '%g %g', [2, 360]);
[F925, cf] = fscanf(ff, '%g %g', [2, 360]);[F930, cf] = fscanf(ff, '%g %g', [2, 360]);
[F935, cf] = fscanf(ff, '%g %g', [2, 360]);[F940, cf] = fscanf(ff, '%g %g', [2, 360]);
[F945, cf] = fscanf(ff, '%g %g', [2, 360]);[F950, cf] = fscanf(ff, '%g %g', [2, 360]);
```

```
[S10, cj] = fscanf(ff, '%d', [1]);[F1005, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1015, cf] = fscanf(ff, '%g %g', [2, 360]);[F1020, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1025, cf] = fscanf(ff, '%g %g', [2, 360]);[F1030, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1035, cf] = fscanf(ff, '%g %g', [2, 360]);[F1040, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1045, cf] = fscanf(ff, '%g %g', [2, 360]);[F1050, cf] = fscanf(ff, '%g %g', [2, 360]);
[S11, cj] = fscanf(ff, '%d', [1]);[F1105, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1115, cf] = fscanf(ff, '%g %g', [2, 360]);[F1120, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1125, cf] = fscanf(ff, '%g %g', [2, 360]);[F1130, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1135, cf] = fscanf(ff, '%g %g', [2, 360]);[F1140, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1145, cf] = fscanf(ff, '%g %g', [2, 360]);[F1150, cf] = fscanf(ff, '%g %g', [2, 360]);
[S12, cj] = fscanf(ff, '%d', [1]);[F1205, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1215, cf] = fscanf(ff, '%g %g', [2, 360]);[F1220, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1225, cf] = fscanf(ff, '%g %g', [2, 360]);[F1230, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1235, cf] = fscanf(ff, '%g %g', [2, 360]);[F1240, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1245, cf] = fscanf(ff, '%g %g', [2, 360]);[F1250, cf] = fscanf(ff, '%g %g', [2, 360]);
[S13, cj] = fscanf(ff, '%d', [1]);[F1305, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1315, cf] = fscanf(ff, '%g %g', [2, 360]);[F1320, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1325, cf] = fscanf(ff, '%g %g', [2, 360]);[F1330, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1335, cf] = fscanf(ff, '%g %g', [2, 360]);[F1340, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1345, cf] = fscanf(ff, '%g %g', [2, 360]);[F1350, cf] = fscanf(ff, '%g %g', [2, 360]);
[S14, cj] = fscanf(ff, '%d', [1]);[F1405, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1415, cf] = fscanf(ff, '%g %g', [2, 360]);[F1420, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1425, cf] = fscanf(ff, '%g %g', [2, 360]);[F1430, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1435, cf] = fscanf(ff, '%g %g', [2, 360]);[F1440, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1445, cf] = fscanf(ff, '%g %g', [2, 360]);[F1450, cf] = fscanf(ff, '%g %g', [2, 360]);
[S15, cj] = fscanf(ff, '%d', [1]);[F1505, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1515, cf] = fscanf(ff, '%g %g', [2, 360]);[F1520, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1525, cf] = fscanf(ff, '%g %g', [2, 360]);[F1530, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1535, cf] = fscanf(ff, '%g %g', [2, 360]);[F1540, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1545, cf] = fscanf(ff, '%g %g', [2, 360]);[F1550, cf] = fscanf(ff, '%g %g', [2, 360]);
[S16, cj] = fscanf(ff, '%d', [1]);[F1605, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1615, cf] = fscanf(ff, '%g %g', [2, 360]);[F1620, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1625, cf] = fscanf(ff, '%g %g', [2, 360]);[F1630, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1635, cf] = fscanf(ff, '%g %g', [2, 360]);[F1640, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1645, cf] = fscanf(ff, '%g %g', [2, 360]);[F1650, cf] = fscanf(ff, '%g %g', [2, 360]);
[S17, cj] = fscanf(ff, '%d', [1]);[F1705, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1715, cf] = fscanf(ff, '%g %g', [2, 360]);[F1720, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1725, cf] = fscanf(ff, '%g %g', [2, 360]);[F1730, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1735, cf] = fscanf(ff, '%g %g', [2, 360]);[F1740, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1745, cf] = fscanf(ff, '%g %g', [2, 360]);[F1750, cf] = fscanf(ff, '%g %g', [2, 360]);
[S18, cj] = fscanf(ff, '%d', [1]);[F1805, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1815, cf] = fscanf(ff, '%g %g', [2, 360]);[F1820, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1825, cf] = fscanf(ff, '%g %g', [2, 360]);[F1830, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1835, cf] = fscanf(ff, '%g %g', [2, 360]);[F1840, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1845, cf] = fscanf(ff, '%g %g', [2, 360]);[F1850, cf] = fscanf(ff, '%g %g', [2, 360]);
[S19, cj] = fscanf(ff, '%d', [1]);[F1905, cf] = fscanf(ff, '%g %g', [2, 360]);
```

```
[F1915, cf] = fscanf(ff, '%g %g', [2, 360]);[F1920, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1925, cf] = fscanf(ff, '%g %g', [2, 360]);[F1930, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1935, cf] = fscanf(ff, '%g %g', [2, 360]);[F1940, cf] = fscanf(ff, '%g %g', [2, 360]);
[F1945, cf] = fscanf(ff, '%g %g', [2, 360]);[F1950, cf] = fscanf(ff, '%g %g', [2, 360]);
[S20, cj] = fscanf(ff, '%d', [1]);[F2005, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2015, cf] = fscanf(ff, '%g %g', [2, 360]);[F2020, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2025, cf] = fscanf(ff, '%g %g', [2, 360]);[F2030, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2035, cf] = fscanf(ff, '%g %g', [2, 360]);[F2040, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2045, cf] = fscanf(ff, '%g %g', [2, 360]);[F2050, cf] = fscanf(ff, '%g %g', [2, 360]);
[S21, cj] = fscanf(ff, '%d', [1]);[F2105, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2115, cf] = fscanf(ff, '%g %g', [2, 360]);[F2120, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2125, cf] = fscanf(ff, '%g %g', [2, 360]);[F2130, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2135, cf] = fscanf(ff, '%g %g', [2, 360]);[F2140, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2145, cf] = fscanf(ff, '%g %g', [2, 360]);[F2150, cf] = fscanf(ff, '%g %g', [2, 360]);
[S22, cj] = fscanf(ff, '%d', [1]);[F2205, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2215, cf] = fscanf(ff, '%g %g', [2, 360]);[F2220, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2225, cf] = fscanf(ff, '%g %g', [2, 360]);[F2230, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2235, cf] = fscanf(ff, '%g %g', [2, 360]);[F2240, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2245, cf] = fscanf(ff, '%g %g', [2, 360]);[F2250, cf] = fscanf(ff, '%g %g', [2, 360]);
[S23, cj] = fscanf(ff, '%d', [1]);[F2305, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2315, cf] = fscanf(ff, '%g %g', [2, 360]);[F2320, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2325, cf] = fscanf(ff, '%g %g', [2, 360]);[F2330, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2335, cf] = fscanf(ff, '%g %g', [2, 360]);[F2340, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2345, cf] = fscanf(ff, '%g %g', [2, 360]);[F2350, cf] = fscanf(ff, '%g %g', [2, 360]);
[S24, cj] = fscanf(ff, '%d', [1]);[F2405, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2415, cf] = fscanf(ff, '%g %g', [2, 360]);[F2420, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2425, cf] = fscanf(ff, '%g %g', [2, 360]);[F2430, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2435, cf] = fscanf(ff, '%g %g', [2, 360]);[F2440, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2445, cf] = fscanf(ff, '%g %g', [2, 360]);[F2450, cf] = fscanf(ff, '%g %g', [2, 360]);
[S25, cj] = fscanf(ff, '%d', [1]);[F2505, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2515, cf] = fscanf(ff, '%g %g', [2, 360]);[F2520, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2525, cf] = fscanf(ff, '%g %g', [2, 360]);[F2530, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2535, cf] = fscanf(ff, '%g %g', [2, 360]);[F2540, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2545, cf] = fscanf(ff, '%g %g', [2, 360]);[F2550, cf] = fscanf(ff, '%g %g', [2, 360]);
[S26, cj] = fscanf(ff, '%d', [1]);[F2605, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2615, cf] = fscanf(ff, '%g %g', [2, 360]);[F2620, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2625, cf] = fscanf(ff, '%g %g', [2, 360]);[F2630, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2635, cf] = fscanf(ff, '%g %g', [2, 360]);[F2640, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2645, cf] = fscanf(ff, '%g %g', [2, 360]);[F2650, cf] = fscanf(ff, '%g %g', [2, 360]);
[S27, cj] = fscanf(ff, '%d', [1]);[F2705, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2715, cf] = fscanf(ff, '%g %g', [2, 360]);[F2720, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2725, cf] = fscanf(ff, '%g %g', [2, 360]);[F2730, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2735, cf] = fscanf(ff, '%g %g', [2, 360]);[F2740, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2745, cf] = fscanf(ff, '%g %g', [2, 360]);[F2750, cf] = fscanf(ff, '%g %g', [2, 360]);
[S28, cj] = fscanf(ff, '%d', [1]);[F2805, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2815, cf] = fscanf(ff, '%g %g', [2, 360]);[F2820, cf] = fscanf(ff, '%g %g', [2, 360]);
```

```
[F2825, cf] = fscanf(ff, '%g %g', [2, 360]);[F2830, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2835, cf] = fscanf(ff, '%g %g', [2, 360]);[F2840, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2845, cf] = fscanf(ff, '%g %g', [2, 360]);[F2850, cf] = fscanf(ff, '%g %g', [2, 360]);
[S29, cj] = fscanf(ff, '%d', [1]);[F2905, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2915, cf] = fscanf(ff, '%g %g', [2, 360]);[F2920, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2925, cf] = fscanf(ff, '%g %g', [2, 360]);[F2930, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2935, cf] = fscanf(ff, '%g %g', [2, 360]);[F2940, cf] = fscanf(ff, '%g %g', [2, 360]);
[F2945, cf] = fscanf(ff, '%g %g', [2, 360]);[F2950, cf] = fscanf(ff, '%g %g', [2, 360]);
[S30, cj] = fscanf(ff, '%d', [1]);[F3005, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3015, cf] = fscanf(ff, '%g %g', [2, 360]);[F3020, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3025, cf] = fscanf(ff, '%g %g', [2, 360]);[F3030, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3035, cf] = fscanf(ff, '%g %g', [2, 360]);[F3040, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3045, cf] = fscanf(ff, '%g %g', [2, 360]);[F3050, cf] = fscanf(ff, '%g %g', [2, 360]);
[S31, cj] = fscanf(ff, '%d', [1]);[F3105, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3115, cf] = fscanf(ff, '%g %g', [2, 360]);[F3120, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3125, cf] = fscanf(ff, '%g %g', [2, 360]);[F3130, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3135, cf] = fscanf(ff, '%g %g', [2, 360]);[F3140, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3145, cf] = fscanf(ff, '%g %g', [2, 360]);[F3150, cf] = fscanf(ff, '%g %g', [2, 360]);
[S32, cj] = fscanf(ff, '%d', [1]);[F3205, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3215, cf] = fscanf(ff, '%g %g', [2, 360]);[F3220, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3225, cf] = fscanf(ff, '%g %g', [2, 360]);[F3230, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3235, cf] = fscanf(ff, '%g %g', [2, 360]);[F3240, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3245, cf] = fscanf(ff, '%g %g', [2, 360]);[F3250, cf] = fscanf(ff, '%g %g', [2, 360]);
[S33, cj] = fscanf(ff, '%d', [1]);[F3305, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3315, cf] = fscanf(ff, '%g %g', [2, 360]);[F3320, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3325, cf] = fscanf(ff, '%g %g', [2, 360]);[F3330, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3335, cf] = fscanf(ff, '%g %g', [2, 360]);[F3340, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3345, cf] = fscanf(ff, '%g %g', [2, 360]);[F3350, cf] = fscanf(ff, '%g %g', [2, 360]);
[S34, cj] = fscanf(ff, '%d', [1]);[F3405, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3415, cf] = fscanf(ff, '%g %g', [2, 360]);[F3420, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3425, cf] = fscanf(ff, '%g %g', [2, 360]);[F3430, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3435, cf] = fscanf(ff, '%g %g', [2, 360]);[F3440, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3445, cf] = fscanf(ff, '%g %g', [2, 360]);[F3450, cf] = fscanf(ff, '%g %g', [2, 360]);
[S35, cj] = fscanf(ff, '%d', [1]);[F3505, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3515, cf] = fscanf(ff, '%g %g', [2, 360]);[F3520, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3525, cf] = fscanf(ff, '%g %g', [2, 360]);[F3530, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3535, cf] = fscanf(ff, '%g %g', [2, 360]);[F3540, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3545, cf] = fscanf(ff, '%g %g', [2, 360]);[F3550, cf] = fscanf(ff, '%g %g', [2, 360]);
[S36, cj] = fscanf(ff, '%d', [1]);[F3605, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3615, cf] = fscanf(ff, '%g %g', [2, 360]);[F3620, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3625, cf] = fscanf(ff, '%g %g', [2, 360]);[F3630, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3635, cf] = fscanf(ff, '%g %g', [2, 360]);[F3640, cf] = fscanf(ff, '%g %g', [2, 360]);
[F3645, cf] = fscanf(ff, '%g %g', [2, 360]);[F3650, cf] = fscanf(ff, '%g %g', [2, 360]);
F105 = F105';F205 = F205';F305 = F305';F405 = F405';F505 = F505';F605 = F605';
F705 = F705';F805 = F805';F905 = F905';F1005 = F1005';F1105 = F1105';
F1205 = F1205';F1305 = F1305';F1405 = F1405';F1505 = F1505';F1605 = F1605';
```

```
F1705 = F1705';F1805 = F1805';F1905 = F1905';F2005 = F2005';F2105 = F2105';
F2205 = F2205';F2305 = F2305';F2405 = F2405';F2505 = F2505';F2605 = F2605';
F2705 = F2705';F2805 = F2805';F2905 = F2905';F3005 = F3005';F3105 = F3105';
F3205 = F3205';F3305 = F3305';F3405 = F3405';F3505 = F3505';F3605 = F3605';
Fz05 = [];Fz05 = [Fz05, F105];Fz05 = [Fz05, F205];Fz05 = [Fz05, F305];
Fz05 = [Fz05, F405];Fz05 = [Fz05, F505];Fz05 = [Fz05, F605];Fz05 = [Fz05, F705];
Fz05 = [Fz05, F805];Fz05 = [Fz05, F905];Fz05 = [Fz05, F1005];Fz05 = [Fz05, F1105];
Fz05 = [Fz05, F1205];Fz05 = [Fz05, F1305];Fz05 = [Fz05, F1405];
Fz05 = [Fz05, F1505];Fz05 = [Fz05, F1605];Fz05 = [Fz05, F1705];
Fz05 = [Fz05, F1805];Fz05 = [Fz05, F1905];Fz05 = [Fz05, F2005];
Fz05 = [Fz05, F2105];Fz05 = [Fz05, F2205];Fz05 = [Fz05, F2305];
Fz05 = [Fz05, F2405];Fz05 = [Fz05, F2505];Fz05 = [Fz05, F2605];
Fz05 = [Fz05, F2705];Fz05 = [Fz05, F2805];Fz05 = [Fz05, F2905];
Fz05 = [Fz05, F3005];Fz05 = [Fz05, F3105];Fz05 = [Fz05, F3205];
Fz05 = [Fz05, F3305];Fz05 = [Fz05, F3405];Fz05 = [Fz05, F3505];
Fz05 = [Fz05, F3605];Fz05 = [Fz05, rdeg];
Fz05 = sortrows(Fz05, [73]);
clear F105;clear F205;clear F305;clear F405;clear F505;clear F605;clear F705;
clear F805;clear F905;clear F1005;clear F1105;clear F1205;clear F1305;clear F1405;
clear F1505;clear F1605;clear F1705;clear F1805;clear F1905;clear F2005;clear F2105;
clear F2205;clear F2305;clear F2405;clear F2505;clear F2605;clear F2705;clear F2805;
clear F2905;clear F3005;clear F3105;clear F3205;clear F3305;clear F3405;clear F3505;
clear F3605;fprintf('.');
F115 = F115';F215 = F215';F315 = F315';F415 = F415';F515 = F515';F615 = F615';
F715 = F715';F815 = F815';F915 = F915';F1015 = F1015';F1115 = F1115';
F1215 = F1215';F1315 = F1315';F1415 = F1415';F1515 = F1515';F1615 = F1615';
F1715 = F1715';F1815 = F1815';F1915 = F1915';F2015 = F2015';F2115 = F2115';
F2215 = F2215';F2315 = F2315';F2415 = F2415';F2515 = F2515';F2615 = F2615';
F2715 = F2715';F2815 = F2815';F2915 = F2915';F3015 = F3015';F3115 = F3115';
F3215 = F3215';F3315 = F3315';F3415 = F3415';F3515 = F3515';F3615 = F3615';
Fz15 = [];Fz15 = [Fz15, F115];Fz15 = [Fz15, F215];Fz15 = [Fz15, F315];
Fz15 = [Fz15, F415];Fz15 = [Fz15, F515];Fz15 = [Fz15, F615];Fz15 = [Fz15, F715];
Fz15 = [Fz15, F815];Fz15 = [Fz15, F915];Fz15 = [Fz15, F1015];Fz15 = [Fz15, F1115];
Fz15 = [Fz15, F1215];Fz15 = [Fz15, F1315];Fz15 = [Fz15, F1415];
Fz15 = [Fz15, F1515];Fz15 = [Fz15, F1615];Fz15 = [Fz15, F1715];
Fz15 = [Fz15, F1815];Fz15 = [Fz15, F1915];Fz15 = [Fz15, F2015];
Fz15 = [Fz15, F2115];Fz15 = [Fz15, F2215];Fz15 = [Fz15, F2315];
Fz15 = [Fz15, F2415];Fz15 = [Fz15, F2515];Fz15 = [Fz15, F2615];
Fz15 = [Fz15, F2715];Fz15 = [Fz15, F2815];Fz15 = [Fz15, F2915];
Fz15 = [Fz15, F3015];Fz15 = [Fz15, F3115];Fz15 = [Fz15, F3215];
Fz15 = [Fz15, F3315];Fz15 = [Fz15, F3415];Fz15 = [Fz15, F3515];
Fz15 = [Fz15, F3615];Fz15 = [Fz15, rdeg];
Fz15 = sortrows(Fz15, [73]);
clear F115;clear F215;clear F315;clear F415;clear F515;clear F615;clear F715;
clear F815;clear F915;clear F1015;clear F1115;clear F1215;clear F1315;clear F1415;
clear F1515;clear F1615;clear F1715;clear F1815;clear F1915;clear F2015;clear F2115;
clear F2215;clear F2315;clear F2415;clear F2515;clear F2615;clear F2715;clear F2815;
clear F2915;clear F3015;clear F3115;clear F3215;clear F3315;clear F3415;clear F3515;
clear F3615;

clear S1;clear S2;clear S3;clear S4;clear S5;clear S6;clear S7;clear S8;clear S9;clear S10;
clear S11;clear S12;clear S13;clear S14;clear S15;clear S16;clear S17;clear S18;
clear S19;clear S20;clear S21;clear S22;clear S23;clear S24;clear S25;clear S26;
clear S27;clear S28;clear S29;clear S30;clear S31;clear S32;clear S33;clear S34;
clear S35;clear S36;fprintf('.');
F120 = F120';F220 = F220';F320 = F320';F420 = F420';F520 = F520';F620 = F620';
F720 = F720';F820 = F820';F920 = F920';F1020 = F1020';F1120 = F1120';
F1220 = F1220';F1320 = F1320';F1420 = F1420';F1520 = F1520';F1620 = F1620';
F1720 = F1720';F1820 = F1820';F1920 = F1920';F2020 = F2020';F2120 = F2120';
F2220 = F2220';F2320 = F2320';F2420 = F2420';F2520 = F2520';F2620 = F2620';
F2720 = F2720';F2820 = F2820';F2920 = F2920';F3020 = F3020';F3120 = F3120';
F3220 = F3220';F3320 = F3320';F3420 = F3420';F3520 = F3520';F3620 = F3620';
Fz20 = [];Fz20 = [Fz20, F120];Fz20 = [Fz20, F220];Fz20 = [Fz20, F320];
Fz20 = [Fz20, F420];Fz20 = [Fz20, F520];Fz20 = [Fz20, F620];Fz20 = [Fz20, F720];
Fz20 = [Fz20, F820];Fz20 = [Fz20, F920];Fz20 = [Fz20, F1020];Fz20 = [Fz20, F1120];
Fz20 = [Fz20, F1220];Fz20 = [Fz20, F1320];Fz20 = [Fz20, F1420];
Fz20 = [Fz20, F1520];Fz20 = [Fz20, F1620];Fz20 = [Fz20, F1720];
Fz20 = [Fz20, F1820];Fz20 = [Fz20, F1920];Fz20 = [Fz20, F2020];
Fz20 = [Fz20, F2120];Fz20 = [Fz20, F2220];Fz20 = [Fz20, F2320];
Fz20 = [Fz20, F2420];Fz20 = [Fz20, F2520];Fz20 = [Fz20, F2620];
Fz20 = [Fz20, F2720];Fz20 = [Fz20, F2820];Fz20 = [Fz20, F2920];
Fz20 = [Fz20, F3020];Fz20 = [Fz20, F3120];Fz20 = [Fz20, F3220];
Fz20 = [Fz20, F3320];Fz20 = [Fz20, F3420];Fz20 = [Fz20, F3520];
Fz20 = [Fz20, F3620];Fz20 = [Fz20, rdeg];
Fz20 = sortrows(Fz20, [73]);
clear F120;clear F220;clear F320;clear F420;clear F520;clear F620;clear F720;
clear F820;clear F920;clear F1020;clear F1120;clear F1220;clear F1320;clear F1420;
clear F1520;clear F1620;clear F1720;clear F1820;clear F1920;clear F2020;clear F2120;
clear F2220;clear F2320;clear F2420;clear F2520;clear F2620;clear F2720;clear F2820;
clear F2920;clear F3020;clear F3120;clear F3220;clear F3320;clear F3420;clear F3520;
clear F3620;fprintf('.');
F125 = F125';F225 = F225';F325 = F325';F425 = F425';F525 = F525';F625 = F625';
F725 = F725';F825 = F825';F925 = F925';F1025 = F1025';F1125 = F1125';
F1225 = F1225';F1325 = F1325';F1425 = F1425';F1525 = F1525';F1625 = F1625';
F1725 = F1725';F1825 = F1825';F1925 = F1925';F2025 = F2025';F2125 = F2125';
F2225 = F2225';F2325 = F2325';F2425 = F2425';F2525 = F2525';F2625 = F2625';
F2725 = F2725';F2825 = F2825';F2925 = F2925';F3025 = F3025';F3125 = F3125';
F3225 = F3225';F3325 = F3325';F3425 = F3425';F3525 = F3525';F3625 = F3625';
Fz25 = [];Fz25 = [Fz25, F125];Fz25 = [Fz25, F225];Fz25 = [Fz25, F325];
Fz25 = [Fz25, F425];Fz25 = [Fz25, F525];Fz25 = [Fz25, F625];Fz25 = [Fz25, F725];
Fz25 = [Fz25, F825];Fz25 = [Fz25, F925];Fz25 = [Fz25, F1025];Fz25 = [Fz25, F1125];
Fz25 = [Fz25, F1225];Fz25 = [Fz25, F1325];Fz25 = [Fz25, F1425];
```

```
Fz25 = [Fz25, F1525];Fz25 = [Fz25, F1625];Fz25 = [Fz25, F1725];
Fz25 = [Fz25, F1825];Fz25 = [Fz25, F1925];Fz25 = [Fz25, F2025];
Fz25 = [Fz25, F2125];Fz25 = [Fz25, F2225];Fz25 = [Fz25, F2325];
Fz25 = [Fz25, F2425];Fz25 = [Fz25, F2525];Fz25 = [Fz25, F2625];
Fz25 = [Fz25, F2725];Fz25 = [Fz25, F2825];Fz25 = [Fz25, F2925];
Fz25 = [Fz25, F3025];Fz25 = [Fz25, F3125];Fz25 = [Fz25, F3225];
Fz25 = [Fz25, F3325];Fz25 = [Fz25, F3425];Fz25 = [Fz25, F3525];
Fz25 = [Fz25, F3625];Fz25 = [Fz25, rdeg];
Fz25 = sortrows(Fz25, [73]);
clear F125;clear F225;clear F325;clear F425;clear F525;clear F625;clear F725;
clear F825;clear F925;clear F1025;clear F1125;clear F1225;clear F1325;clear F1425;
clear F1525;clear F1625;clear F1725;clear F1825;clear F1925;clear F2025;clear F2125;
clear F2225;clear F2325;clear F2425;clear F2525;clear F2625;clear F2725;clear F2825;
clear F2925;clear F3025;clear F3125;clear F3225;clear F3325;clear F3425;clear F3525;
clear F3625;
fprintf('.');
F130 = F130';F230 = F230';F330 = F330';F430 = F430';F530 = F530';F630 = F630';
F730 = F730';F830 = F830';F930 = F930';F1030 = F1030';F1130 = F1130';
F1230 = F1230';F1330 = F1330';F1430 = F1430';F1530 = F1530';F1630 = F1630';
F1730 = F1730';F1830 = F1830';F1930 = F1930';F2030 = F2030';F2130 = F2130';
F2230 = F2230';F2330 = F2330';F2430 = F2430';F2530 = F2530';F2630 = F2630';
F2730 = F2730';F2830 = F2830';F2930 = F2930';F3030 = F3030';F3130 = F3130';
F3230 = F3230';F3330 = F3330';F3430 = F3430';F3530 = F3530';F3630 = F3630';
Fz30 = [];Fz30 = [Fz30, F130];Fz30 = [Fz30, F230];Fz30 = [Fz30, F330];
Fz30 = [Fz30, F430];Fz30 = [Fz30, F530];Fz30 = [Fz30, F630];Fz30 = [Fz30, F730];
Fz30 = [Fz30, F830];Fz30 = [Fz30, F930];Fz30 = [Fz30, F1030];Fz30 = [Fz30, F1130];
Fz30 = [Fz30, F1230];Fz30 = [Fz30, F1330];Fz30 = [Fz30, F1430];
Fz30 = [Fz30, F1530];Fz30 = [Fz30, F1630];Fz30 = [Fz30, F1730];
Fz30 = [Fz30, F1830];Fz30 = [Fz30, F1930];Fz30 = [Fz30, F2030];
Fz30 = [Fz30, F2130];Fz30 = [Fz30, F2230];Fz30 = [Fz30, F2330];
Fz30 = [Fz30, F2430];Fz30 = [Fz30, F2530];Fz30 = [Fz30, F2630];
Fz30 = [Fz30, F2730];Fz30 = [Fz30, F2830];Fz30 = [Fz30, F2930];
Fz30 = [Fz30, F3030];Fz30 = [Fz30, F3130];Fz30 = [Fz30, F3230];
Fz30 = [Fz30, F3330];Fz30 = [Fz30, F3430];Fz30 = [Fz30, F3530];
Fz30 = [Fz30, F3630];Fz30 = [Fz30, rdeg];
Fz30 = sortrows(Fz30, [73]);
clear F130;clear F230;clear F330;clear F430;clear F530;clear F630;clear F730;
clear F830;clear F930;clear F1030;clear F1130;clear F1230;clear F1330;clear F1430;
clear F1530;clear F1630;clear F1730;clear F1830;clear F1930;clear F2030;clear F2130;
clear F2230;clear F2330;clear F2430;clear F2530;clear F2630;clear F2730;clear F2830;
clear F2930;clear F3030;clear F3130;clear F3230;clear F3330;clear F3430;clear F3530;
clear F3630;fprintf('.');
F135 = F135';F235 = F235';F335 = F335';F435 = F435';F535 = F535';F635 = F635';
F735 = F735';F835 = F835';F935 = F935';F1035 = F1035';F1135 = F1135';
F1235 = F1235';F1335 = F1335';F1435 = F1435';F1535 = F1535';F1635 = F1635';
F1735 = F1735';F1835 = F1835';F1935 = F1935';F2035 = F2035';F2135 = F2135';
```

```
F2235 = F2235';F2335 = F2335';F2435 = F2435';F2535 = F2535';F2635 = F2635';
F2735 = F2735';F2835 = F2835';F2935 = F2935';F3035 = F3035';F3135 = F3135';
F3235 = F3235';F3335 = F3335';F3435 = F3435';F3535 = F3535';F3635 = F3635';

Fz35 = [];Fz35 = [Fz35, F135];Fz35 = [Fz35, F235];Fz35 = [Fz35, F335];
Fz35 = [Fz35, F435];Fz35 = [Fz35, F535];Fz35 = [Fz35, F635];Fz35 = [Fz35, F735];
Fz35 = [Fz35, F835];Fz35 = [Fz35, F935];Fz35 = [Fz35, F1035];Fz35 = [Fz35, F1135];
Fz35 = [Fz35, F1235];Fz35 = [Fz35, F1335];Fz35 = [Fz35, F1435];
Fz35 = [Fz35, F1535];Fz35 = [Fz35, F1635];Fz35 = [Fz35, F1735];
Fz35 = [Fz35, F1835];Fz35 = [Fz35, F1935];Fz35 = [Fz35, F2035];
Fz35 = [Fz35, F2135];Fz35 = [Fz35, F2235];Fz35 = [Fz35, F2335];
Fz35 = [Fz35, F2435];Fz35 = [Fz35, F2535];Fz35 = [Fz35, F2635];
Fz35 = [Fz35, F2735];Fz35 = [Fz35, F2835];Fz35 = [Fz35, F2935];
Fz35 = [Fz35, F3035];Fz35 = [Fz35, F3135];Fz35 = [Fz35, F3235];
Fz35 = [Fz35, F3335];Fz35 = [Fz35, F3435];Fz35 = [Fz35, F3535];
Fz35 = [Fz35, F3635];Fz35 = [Fz35, rdeg];
Fz35 = sortrows(Fz35, [73]);
clear F135;clear F235;clear F335;clear F435;clear F535;clear F635;clear F735;
clear F835;clear F935;clear F1035;clear F1135;clear F1235;clear F1335;clear F1435;
clear F1535;clear F1635;clear F1735;clear F1835;clear F1935;clear F2035;clear F2135;
clear F2235;clear F2335;clear F2435;clear F2535;clear F2635;clear F2735;clear F2835;
clear F2935;clear F3035;clear F3135;clear F3235;clear F3335;clear F3435;clear F3535;
clear F3635;fprintf('.');
F140 = F140';F240 = F240';F340 = F340';F440 = F440';F540 = F540';F640 = F640';
F740 = F740';F840 = F840';F940 = F940';F1040 = F1040';F1140 = F1140';
F1240 = F1240';F1340 = F1340';F1440 = F1440';F1540 = F1540';F1640 = F1640';
F1740 = F1740';F1840 = F1840';F1940 = F1940';F2040 = F2040';F2140 = F2140';
F2240 = F2240';F2340 = F2340';F2440 = F2440';F2540 = F2540';F2640 = F2640';
F2740 = F2740';F2840 = F2840';F2940 = F2940';F3040 = F3040';F3140 = F3140';
F3240 = F3240';F3340 = F3340';F3440 = F3440';F3540 = F3540';F3640 = F3640';
Fz40 = [];Fz40 = [Fz40, F140];Fz40 = [Fz40, F240];Fz40 = [Fz40, F340];
Fz40 = [Fz40, F440];Fz40 = [Fz40, F540];Fz40 = [Fz40, F640];Fz40 = [Fz40, F740];
Fz40 = [Fz40, F840];Fz40 = [Fz40, F940];Fz40 = [Fz40, F1040];Fz40 = [Fz40, F1140];
Fz40 = [Fz40, F1240];Fz40 = [Fz40, F1340];Fz40 = [Fz40, F1440];
Fz40 = [Fz40, F1540];Fz40 = [Fz40, F1640];Fz40 = [Fz40, F1740];
Fz40 = [Fz40, F1840];Fz40 = [Fz40, F1940];Fz40 = [Fz40, F2040];
Fz40 = [Fz40, F2140];Fz40 = [Fz40, F2240];Fz40 = [Fz40, F2340];
Fz40 = [Fz40, F2440];Fz40 = [Fz40, F2540];Fz40 = [Fz40, F2640];
Fz40 = [Fz40, F2740];Fz40 = [Fz40, F2840];Fz40 = [Fz40, F2940];
Fz40 = [Fz40, F3040];Fz40 = [Fz40, F3140];Fz40 = [Fz40, F3240];
Fz40 = [Fz40, F3340];Fz40 = [Fz40, F3440];Fz40 = [Fz40, F3540];
Fz40 = [Fz40, F3640];Fz40 = [Fz40, rdeg];
Fz40 = sortrows(Fz40, [73]);
clear F140;clear F240;clear F340;clear F440;clear F540;clear F640;clear F740;
clear F840;clear F940;clear F1040;clear F1140;clear F1240;clear F1340;clear F1440;
clear F1540;clear F1640;clear F1740;clear F1840;clear F1940;clear F2040;clear F2140;
clear F2240;clear F2340;clear F2440;clear F2540;clear F2640;clear F2740;clear F2840;
clear F2940;clear F3040;clear F3140;clear F3240;clear F3340;clear F3440;clear F3540;
clear F3640;fprintf('.');
F145 = F145';F245 = F245';F345 = F345';F445 = F445';F545 = F545';F645 = F645';
F745 = F745';F845 = F845';F945 = F945';F1045 = F1045';F1145 = F1145';
F1245 = F1245';F1345 = F1345';F1445 = F1445';F1545 = F1545';F1645 = F1645';
F1745 = F1745';F1845 = F1845';F1945 = F1945';F2045 = F2045';F2145 = F2145';
F2245 = F2245';F2345 = F2345';F2445 = F2445';F2545 = F2545';F2645 = F2645';
F2745 = F2745';F2845 = F2845';F2945 = F2945';F3045 = F3045';F3145 = F3145';
F3245 = F3245';F3345 = F3345';F3445 = F3445';F3545 = F3545';F3645 = F3645';
Fz45 = [];Fz45 = [Fz45, F145];Fz45 = [Fz45, F245];Fz45 = [Fz45, F345];
Fz45 = [Fz45, F445];Fz45 = [Fz45, F545];Fz45 = [Fz45, F645];Fz45 = [Fz45, F745];
Fz45 = [Fz45, F845];Fz45 = [Fz45, F945];Fz45 = [Fz45, F1045];Fz45 = [Fz45, F1145];
Fz45 = [Fz45, F1245];Fz45 = [Fz45, F1345];Fz45 = [Fz45, F1445];
Fz45 = [Fz45, F1545];Fz45 = [Fz45, F1645];Fz45 = [Fz45, F1745];
Fz45 = [Fz45, F1845];Fz45 = [Fz45, F1945];Fz45 = [Fz45, F2045];
Fz45 = [Fz45, F2145];Fz45 = [Fz45, F2245];Fz45 = [Fz45, F2345];
Fz45 = [Fz45, F2445];Fz45 = [Fz45, F2545];Fz45 = [Fz45, F2645];
Fz45 = [Fz45, F2745];Fz45 = [Fz45, F2845];Fz45 = [Fz45, F2945];
Fz45 = [Fz45, F3045];Fz45 = [Fz45, F3145];Fz45 = [Fz45, F3245];
Fz45 = [Fz45, F3345];Fz45 = [Fz45, F3445];Fz45 = [Fz45, F3545];
Fz45 = [Fz45, F3645];Fz45 = [Fz45, rdeg];
Fz45 = sortrows(Fz45, [73]);
clear F145;clear F245;clear F345;clear F445;clear F545;clear F645;clear F745;
clear F845;clear F945;clear F1045;clear F1145;clear F1245;clear F1345;clear F1445;
clear F1545;clear F1645;clear F1745;clear F1845;clear F1945;clear F2045;clear F2145;
clear F2245;clear F2345;clear F2445;clear F2545;clear F2645;clear F2745;clear F2845;
clear F2945;clear F3045;clear F3145;clear F3245;clear F3345;clear F3445;clear F3545;
clear F3645;fprintf('.');
F150 = F150';F250 = F250';F350 = F350';F450 = F450';F550 = F550';F650 = F650';
F750 = F750';F850 = F850';F950 = F950';F1050 = F1050';F1150 = F1150';
F1250 = F1250';F1350 = F1350';F1450 = F1450';F1550 = F1550';F1650 = F1650';
F1750 = F1750';F1850 = F1850';F1950 = F1950';F2050 = F2050';F2150 = F2150';
F2250 = F2250';F2350 = F2350';F2450 = F2450';F2550 = F2550';F2650 = F2650';
F2750 = F2750';F2850 = F2850';F2950 = F2950';F3050 = F3050';F3150 = F3150';
F3250 = F3250';F3350 = F3350';F3450 = F3450';F3550 = F3550';F3650 = F3650';
Fz50 = [];Fz50 = [Fz50, F150];Fz50 = [Fz50, F250];Fz50 = [Fz50, F350];
Fz50 = [Fz50, F450];Fz50 = [Fz50, F550];Fz50 = [Fz50, F650];Fz50 = [Fz50, F750];
Fz50 = [Fz50, F850];Fz50 = [Fz50, F950];Fz50 = [Fz50, F1050];Fz50 = [Fz50, F1150];
Fz50 = [Fz50, F1250];Fz50 = [Fz50, F1350];Fz50 = [Fz50, F1450];
Fz50 = [Fz50, F1550];Fz50 = [Fz50, F1650];Fz50 = [Fz50, F1750];
Fz50 = [Fz50, F1850];Fz50 = [Fz50, F1950];Fz50 = [Fz50, F2050];
Fz50 = [Fz50, F2150];Fz50 = [Fz50, F2250];Fz50 = [Fz50, F2350];
Fz50 = [Fz50, F2450];Fz50 = [Fz50, F2550];Fz50 = [Fz50, F2650];
Fz50 = [Fz50, F2750];Fz50 = [Fz50, F2850];Fz50 = [Fz50, F2950];
Fz50 = [Fz50, F3050];Fz50 = [Fz50, F3150];Fz50 = [Fz50, F3250];
```

```
Fz50 = [Fz50, F3350];Fz50 = [Fz50, F3450];Fz50 = [Fz50, F3550];
Fz50 = [Fz50, F3650];Fz50 = [Fz50, rdeg];
Fz50 = sortrows(Fz50, [73]);

clear F150;clear F250;clear F350;clear F450;clear F550;clear F650;clear F750;
clear F850;clear F950;clear F1050;clear F1150;clear F1250;clear F1350;clear F1450;
clear F1550;clear F1650;clear F1750;clear F1850;clear F1950;clear F2050;clear F2150;
clear F2250;clear F2350;clear F2450;clear F2550;clear F2650;clear F2750;clear F2850;
clear F2950;clear F3050;clear F3150;clear F3250;clear F3350;clear F3450;clear F3550;
clear F3650;fprintf('.');fprintf('. Done');
rdeg = sortrows(rdeg);M05 = [];r = randi([-BWn/2 BWn/2],360,1);
sstep = 36/Ns;rstep = 360/Nr;fprintf('\n\nBuilding Matrices .');
for ss = 1:sstep:36
    fprintf('.');
    for aa = 1:rstep:360
        if(aa - (BWn/2) < 1)
            switch eri
                case 1
                    M05 = [M05; [S(ss,1) rdeg(aa,1) frequ Fz05(aa,2*ss-1) Fz05(aa,2*ss)
Bz(aa,2*ss-1) Bz(aa,2*ss)]];
                case 2
                    M05 = [M05; [S(ss,1) rdeg(aa,1) frequ Fz15(aa,2*ss-1) Fz15(aa,2*ss)
Bz(aa,2*ss-1) Bz(aa,2*ss)]];
                case 3
                    M05 = [M05; [S(ss,1) rdeg(aa,1) frequ Fz20(aa,2*ss-1) Fz20(aa,2*ss)
Bz(aa,2*ss-1) Bz(aa,2*ss)]];
                case 4
                    M05 = [M05; [S(ss,1) rdeg(aa,1) frequ Fz25(aa,2*ss-1) Fz25(aa,2*ss)
Bz(aa,2*ss-1) Bz(aa,2*ss)]];
                case 5
                    M05 = [M05; [S(ss,1) rdeg(aa,1) frequ Fz30(aa,2*ss-1) Fz30(aa,2*ss)
Bz(aa,2*ss-1) Bz(aa,2*ss)]];
                case 6
                    M05 = [M05; [S(ss,1) rdeg(aa,1) frequ Fz35(aa,2*ss-1) Fz35(aa,2*ss)
Bz(aa,2*ss-1) Bz(aa,2*ss)]];
                case 7
                    M05 = [M05; [S(ss,1) rdeg(aa,1) frequ Fz40(aa,2*ss-1) Fz40(aa,2*ss)
Bz(aa,2*ss-1) Bz(aa,2*ss)]];
                case 8
                    M05 = [M05; [S(ss,1) rdeg(aa,1) frequ Fz45(aa,2*ss-1) Fz45(aa,2*ss)
Bz(aa,2*ss-1) Bz(aa,2*ss)]];
                case 9
                    M05 = [M05; [S(ss,1) rdeg(aa,1) frequ Fz50(aa,2*ss-1) Fz50(aa,2*ss)
Bz(aa,2*ss-1) Bz(aa,2*ss)]];
                otherwise
                    disp('ERROR ... Permitivity value out of range!');
                break;
            end
        else
            switch eri
                case 1
                    M05 = [M05; [S(ss,1) rdeg(aa+r(aa,1),1) frequ Fz05(aa+r(aa,1),2*ss-1)
Fz05(aa+r(aa,1),2*ss) Bz(aa+r(aa,1),2*ss-1) Bz(aa+r(aa,1),2*ss)]];
                case 2
                    M05 = [M05; [S(ss,1) rdeg(aa+r(aa,1),1) frequ Fz15(aa+r(aa,1),2*ss-1)
Fz15(aa+r(aa,1),2*ss) Bz(aa+r(aa,1),2*ss-1) Bz(aa+r(aa,1),2*ss)]];
                case 3
                    M05 = [M05; [S(ss,1) rdeg(aa+r(aa,1),1) frequ Fz20(aa+r(aa,1),2*ss-1)
Fz20(aa+r(aa,1),2*ss) Bz(aa+r(aa,1),2*ss-1) Bz(aa+r(aa,1),2*ss)]];
                case 4
                    M05 = [M05; [S(ss,1) rdeg(aa+r(aa,1),1) frequ Fz25(aa+r(aa,1),2*ss-1)
Fz25(aa+r(aa,1),2*ss) Bz(aa+r(aa,1),2*ss-1) Bz(aa+r(aa,1),2*ss)]];
                case 5
                    M05 = [M05; [S(ss,1) rdeg(aa+r(aa,1),1) frequ Fz30(aa+r(aa,1),2*ss-1)
Fz30(aa+r(aa,1),2*ss) Bz(aa+r(aa,1),2*ss-1) Bz(aa+r(aa,1),2*ss)]];
                case 6
                    M05 = [M05; [S(ss,1) rdeg(aa+r(aa,1),1) frequ Fz35(aa+r(aa,1),2*ss-1)
Fz35(aa+r(aa,1),2*ss) Bz(aa+r(aa,1),2*ss-1) Bz(aa+r(aa,1),2*ss)]];
                case 7
                    M05 = [M05; [S(ss,1) rdeg(aa+r(aa,1),1) frequ Fz40(aa+r(aa,1),2*ss-1)
Fz40(aa+r(aa,1),2*ss) Bz(aa+r(aa,1),2*ss-1) Bz(aa+r(aa,1),2*ss)]];
                case 8
                    M05 = [M05; [S(ss,1) rdeg(aa+r(aa,1),1) frequ Fz45(aa+r(aa,1),2*ss-1)
Fz45(aa+r(aa,1),2*ss) Bz(aa+r(aa,1),2*ss-1) Bz(aa+r(aa,1),2*ss)]];
                case 9
                    M05 = [M05; [S(ss,1) rdeg(aa+r(aa,1),1) frequ Fz50(aa+r(aa,1),2*ss-1)
Fz50(aa+r(aa,1),2*ss) Bz(aa+r(aa,1),2*ss-1) Bz(aa+r(aa,1),2*ss)]];
                otherwise
                    disp('ERROR ... Permitivity value out of range!');
                    break;
            end
        end
    end
end
fprintf('. Done');
clear S;clear rdeg;clear Fz05;clear Fz15;clear Fz20;clear Fz25;clear Fz30;clear Fz35;
clear Fz40;clear Fz45;clear Fz50;clear Bz;clear X;clear Y;
row = Ns*Nr; col = 7;M05(:,2) = (M05(:,2) - 180);ReEinc05 = 0;ImEinc05 = 0;
fprintf('\n\nCalculating Normalization Factors ');count = 0;statuspt = row/10;
    for ii = 1:row
        count = count+1;
        if(count >= statuspt)
```

```
            fprintf('.');
            count = 0;
        end
        ReEinc05 = ReEinc05 + M05(ii,6)^2;ImEinc05 = ImEinc05 + M05(ii,7)^2;
    end
    DataNorm05 = 1/(sqrt(ReEinc05 + ImEinc05));  fprintf('.. Done');
xt = [];yt = [];zt = [];InT = [];InB = [];x1 = [];y1 = [];z1 = [];Inb1 = [];In1 = [];
x2 = [];y2 = [];z2 = [];Inb2 = [];In2 = [];x3 = [];y3 = [];z3 = [];Inb3 = [];In3 = [];
x4 = [];y4 = [];z4 = [];Inb4 = [];In4 = [];x5 = [];y5 = [];z5 = [];Inb5 = [];In5 = [];
x6 = [];y6 = [];z6 = [];Inb6 = [];In6 = [];x7 = [];y7 = [];z7 = [];Inb7 = [];In7 = [];
x8 = [];y8 = [];z8 = [];Inb8 = [];In8 = [];x9 = [];y9 = [];z9 = [];Inb9 = [];In9 = [];
x10 = [];y10 = [];z10 = [];Inb10 = [];In10 = [];
x11 = [];y11 = [];z11 = [];Inb11 = [];In11 = [];
x12 = [];y12 = [];z12 = [];Inb12 = [];In12 = [];
x13 = [];y13 = [];z13 = [];Inb13 = [];In13 = [];
x14 = [];y14 = [];z14 = [];Inb14 = [];In14 = [];
x15 = [];y15 = [];z15 = [];Inb15 = [];In15 = [];
x16 = [];y16 = [];z16 = [];Inb16 = [];In16 = [];
x17 = [];y17 = [];z17 = [];Inb17 = [];In17 = [];
x18 = [];y18 = [];z18 = [];Inb18 = [];In18 = [];
x19 = [];y19 = [];z19 = [];Inb19 = [];In19 = [];
x20 = [];y20 = [];z20 = [];Inb20 = [];In20 = [];
x21 = [];y21 = [];z21 = [];Inb21 = [];In21 = [];
x22 = [];y22 = [];z22 = [];Inb22 = [];In22 = [];
x23 = [];y23 = [];z23 = [];Inb23 = [];In23 = [];
x24 = [];y24 = [];z24 = [];Inb24 = [];In24 = [];
x25 = [];y25 = [];z25 = [];Inb25 = [];In25 = [];
x26 = [];y26 = [];z26 = [];Inb26 = [];In26 = [];
x27 = [];y27 = [];z27 = [];Inb27 = [];In27 = [];
x28 = [];y28 = [];z28 = [];Inb28 = [];In28 = [];
x29 = [];y29 = [];z29 = [];Inb29 = [];In29 = [];
x30 = [];y30 = [];z30 = [];Inb30 = [];In30 = [];
x31 = [];y31 = [];z31 = [];Inb31 = [];In31 = [];
x32 = [];y32 = [];z32 = [];Inb32 = [];In32 = [];
x33 = [];y33 = [];z33 = [];Inb33 = [];In33 = [];
x34 = [];y34 = [];z34 = [];Inb34 = [];In34 = [];
x35 = [];y35 = [];z35 = [];Inb35 = [];In35 = [];
x36 = [];y36 = [];z36 = [];Inb36 = [];In36 = [];
G1Tc = [];nx = ImR;ny = ImR;dx = ImD/ImR;dy = ImD/ImR;sx = 0;sy = 0;
vsiB = zeros (nx,ny);vsi1 = zeros (nx,ny);vsi2 = zeros (nx,ny);vsi3 = zeros (nx,ny);
vsi4 = zeros (nx,ny);vsi5 = zeros (nx,ny);vsi6 = zeros (nx,ny);vsi7 = zeros (nx,ny);
vsi8 = zeros (nx,ny);vsi9 = zeros (nx,ny);vsi10 = zeros (nx,ny); vsi11 = zeros (nx,ny);
vsi12 = zeros (nx,ny);vsi13 = zeros (nx,ny);vsi14 = zeros (nx,ny);vsi15 = zeros (nx,ny);
vsi16 = zeros (nx,ny);vsi17 = zeros (nx,ny);vsi18 = zeros (nx,ny);vsi19 = zeros (nx,ny);
vsi20 = zeros (nx,ny);vsi21 = zeros (nx,ny);vsi22 = zeros (nx,ny);vsi23 = zeros (nx,ny);
vsi24 = zeros (nx,ny);vsi25 = zeros (nx,ny);vsi26 = zeros (nx,ny);vsi27 = zeros (nx,ny);
```

```
vsi28 = zeros (nx,ny);vsi29 = zeros (nx,ny);vsi30 = zeros (nx,ny);vsi31 = zeros (nx,ny);
vsi32 = zeros (nx,ny);vsi33 = zeros (nx,ny);vsi34 = zeros (nx,ny);vsi35 = zeros (nx,ny);
vsi36 = zeros (nx,ny);vsiT = zeros (nx,ny);
vsib1 = zeros (nx,ny);vsib2 = zeros (nx,ny);vsib3 = zeros (nx,ny);vsib4 = zeros (nx,ny);
vsib5 = zeros (nx,ny);vsib6 = zeros (nx,ny);vsib7 = zeros (nx,ny);vsib8 = zeros (nx,ny);
vsib9 = zeros (nx,ny);vsib10 = zeros (nx,ny);vsib11 = zeros (nx,ny);
vsib12 = zeros (nx,ny);vsib13 = zeros (nx,ny);vsib14 = zeros (nx,ny);
vsib15 = zeros (nx,ny);vsib16 = zeros (nx,ny);vsib17 = zeros (nx,ny);
vsib18 = zeros (nx,ny);vsib19 = zeros (nx,ny);vsib20 = zeros (nx,ny);
vsib21 = zeros (nx,ny);vsib22 = zeros (nx,ny);vsib23 = zeros (nx,ny);
vsib24 = zeros (nx,ny);vsib25 = zeros (nx,ny);vsib26 = zeros (nx,ny);
vsib27 = zeros (nx,ny);vsib28 = zeros (nx,ny);vsib29 = zeros (nx,ny);
vsib30 = zeros (nx,ny);vsib31 = zeros (nx,ny);vsib32 = zeros (nx,ny);
vsib33 = zeros (nx,ny);vsib34 = zeros (nx,ny);vsib35 = zeros (nx,ny);
vsib36 = zeros (nx,ny);count = 0;cc = 0;
for d = 1:Ns
   for ddd = 1:Nr
         dd = d*ddd;cc = cc + 1;
         Zc = (M05(cc,4)-M05(cc,6)) + i*(M05(cc,5)-M05(cc,7));
         Zi = (M05(cc,6)) + i*(M05(cc,7));  Zc = Zc*DataNorm05*ZMult+ZBias;
         Zi = Zi*DataNorm05;k0 = 2.*pi*frequency./c0;
         norm1 = sqrt(8.*pi.*k0.*.76).*exp(-i.*pi./4-i.*k0./c0.*.76)./(k0).^2;
         Zc = Zc*norm1;Zi = Zi*norm1;
         is = deg2rad(M05(cc,1));  ir = deg2rad(M05(cc,2));
         sfx = (frequency./c0).*(cos(ir) - cos(is));  % Spatial freq in x
         sfy = (frequency./c0).*(sin(ir) - sin(is));  % Spatial freq in y
   switch d
      case 1
         x1 = [x1; real(sfx)];y1 = [y1; real(sfy)];z1 = [z1; abs(Zc)];
         In1 = [In1; [sfx sfy Zc]];Inb1 = [Inb1; [sfx sfy Zi]];
      case 2
         x2 = [x2; real(sfx)];y2 = [y2; real(sfy)];z2 = [z2; abs(Zc)];
         In2 = [In2; [sfx sfy Zc]];Inb2 = [Inb2; [sfx sfy Zi]];
      case 3
         x3 = [x3; real(sfx)];y3 = [y3; real(sfy)];z3 = [z3; abs(Zc)];
         In3 = [In3; [sfx sfy Zc]];Inb3 = [Inb3; [sfx sfy Zi]];
      case 4
         x4 = [x4; real(sfx)];y4 = [y4; real(sfy)];z4 = [z4; abs(Zc)];
         In4 = [In4; [sfx sfy Zc]];Inb4 = [Inb4; [sfx sfy Zi]];
      case 5
         x5 = [x5; real(sfx)];y5 = [y5; real(sfy)];z5 = [z5; abs(Zc)];
         In5 = [In5; [sfx sfy Zc]];Inb5 = [Inb5; [sfx sfy Zi]];
      case 6
         x6 = [x6; real(sfx)];y6 = [y6; real(sfy)];z6 = [z6; abs(Zc)];
         In6 = [In6; [sfx sfy Zc]];Inb6 = [Inb6; [sfx sfy Zi]];
      case 7
```

```
    x7 = [x7; real(sfx)];y7 = [y7; real(sfy)];z7 = [z7; abs(Zc)];
    In7 = [In7; [sfx sfy Zc]];Inb7 = [Inb7; [sfx sfy Zi]];
case 8
    x8 = [x8; real(sfx)];y8 = [y8; real(sfy)];z8 = [z8; abs(Zc)];
    In8 = [In8; [sfx sfy Zc]];Inb8 = [Inb8; [sfx sfy Zi]];
case 9
    x9 = [x9; real(sfx)];y9 = [y9; real(sfy)];z9 = [z9; abs(Zc)];
    In9 = [In9; [sfx sfy Zc]];Inb9 = [Inb9; [sfx sfy Zi]];
case 10
    x10 = [x10; real(sfx)];y10 = [y10; real(sfy)];z10 = [z10; abs(Zc)];
    In10 = [In10; [sfx sfy Zc]];Inb10 = [Inb10; [sfx sfy Zi]];
case 11
    x11 = [x11; real(sfx)];y11 = [y11; real(sfy)];z11 = [z11; abs(Zc)];
    In11 = [In11; [sfx sfy Zc]];Inb11 = [Inb11; [sfx sfy Zi]];
case 12
    x12 = [x12; real(sfx)];y12 = [y12; real(sfy)];z12 = [z12; abs(Zc)];
    In12 = [In12; [sfx sfy Zc]];Inb12 = [Inb12; [sfx sfy Zi]];
case 13
    x13 = [x13; real(sfx)];y13 = [y13; real(sfy)];z13 = [z13; abs(Zc)];
    In13 = [In13; [sfx sfy Zc]];Inb13 = [Inb13; [sfx sfy Zi]];
case 14
    x14 = [x14; real(sfx)];y14 = [y14; real(sfy)];z14 = [z14; abs(Zc)];
    In14 = [In14; [sfx sfy Zc]];Inb14 = [Inb14; [sfx sfy Zi]];
case 15
    x15 = [x15; real(sfx)];y15 = [y15; real(sfy)];z15 = [z15; abs(Zc)];
    In15 = [In15; [sfx sfy Zc]];Inb15 = [Inb15; [sfx sfy Zi]];
case 16
    x16 = [x16; real(sfx)];y16 = [y16; real(sfy)];          z16 = [z16; abs(Zc)];
    In16 = [In16; [sfx sfy Zc]];Inb16 = [Inb16; [sfx sfy Zi]];
case 17
    x17 = [x17; real(sfx)];y17 = [y17; real(sfy)];z17 = [z17; abs(Zc)];
    In17 = [In17; [sfx sfy Zc]];Inb17 = [Inb17; [sfx sfy Zi]];
case 18
    x18 = [x18; real(sfx)];y18 = [y18; real(sfy)];z18 = [z18; abs(Zc)];
    In18 = [In18; [sfx sfy Zc]];Inb18 = [Inb18; [sfx sfy Zi]];
case 19
    x19 = [x19; real(sfx)];y19 = [y19; real(sfy)];z19 = [z19; abs(Zc)];
    In19 = [In19; [sfx sfy Zc]];Inb19 = [Inb19; [sfx sfy Zi]];
case 20
    x20 = [x20; real(sfx)];y20 = [y20; real(sfy)];z20 = [z20; abs(Zc)];
    In20 = [In20; [sfx sfy Zc]];Inb20 = [Inb20; [sfx sfy Zi]];
case 21
    x21 = [x21; real(sfx)];y21 = [y21; real(sfy)];z21 = [z21; abs(Zc)];
    In21 = [In21; [sfx sfy Zc]];Inb21 = [Inb21; [sfx sfy Zi]];
case 22
    x22 = [x22; real(sfx)];y22 = [y22; real(sfy)];z22 = [z22; abs(Zc)];
```

```
   In22 = [In22; [sfx sfy Zc]];Inb22 = [Inb22; [sfx sfy Zi]];
case 23
   x23 = [x23; real(sfx)];y23 = [y23; real(sfy)];z23 = [z23; abs(Zc)];
   In23 = [In23; [sfx sfy Zc]];Inb23 = [Inb23; [sfx sfy Zi]];
case 24
   x24 = [x24; real(sfx)];y24 = [y24; real(sfy)];z24 = [z24; abs(Zc)];
   In24 = [In24; [sfx sfy Zc]];Inb24 = [Inb24; [sfx sfy Zi]];
case 25
   x25 = [x25; real(sfx)];y25 = [y25; real(sfy)];z25 = [z25; abs(Zc)];
   In25 = [In25; [sfx sfy Zc]];Inb25 = [Inb25; [sfx sfy Zi]];
case 26
   x26 = [x26; real(sfx)];y26 = [y26; real(sfy)];z26 = [z26; abs(Zc)];
   In26 = [In26; [sfx sfy Zc]];Inb26 = [Inb26; [sfx sfy Zi]];
case 27
   x27 = [x27; real(sfx)];y27 = [y27; real(sfy)];z27 = [z27; abs(Zc)];
   In27 = [In27; [sfx sfy Zc]];Inb27 = [Inb27; [sfx sfy Zi]];
case 28
   x28 = [x28; real(sfx)];y28 = [y28; real(sfy)];z28 = [z28; abs(Zc)];
   In28 = [In28; [sfx sfy Zc]];Inb28 = [Inb28; [sfx sfy Zi]];
case 29
   x29 = [x29; real(sfx)];y29 = [y29; real(sfy)];z29 = [z29; abs(Zc)];
   In29 = [In29; [sfx sfy Zc]];Inb29 = [Inb29; [sfx sfy Zi]];
case 30
   x30 = [x30; real(sfx)];y30 = [y30; real(sfy)];z30 = [z30; abs(Zc)];
   In30 = [In30; [sfx sfy Zc]];Inb30 = [Inb30; [sfx sfy Zi]];
case 31
   x31 = [x31; real(sfx)]; y31 = [y31; real(sfy)];z31 = [z31; abs(Zc)];
   In31 = [In31; [sfx sfy Zc]];Inb31 = [Inb31; [sfx sfy Zi]];
case 32
   x32 = [x32; real(sfx)];y32 = [y32; real(sfy)];z32 = [z32; abs(Zc)];
   In32 = [In32; [sfx sfy Zc]];Inb32 = [Inb32; [sfx sfy Zi]];
case 33
   x33 = [x33; real(sfx)];y33 = [y33; real(sfy)];z33 = [z33; abs(Zc)];
   In33 = [In33; [sfx sfy Zc]];Inb33 = [Inb33; [sfx sfy Zi]];
case 34
   x34 = [x34; real(sfx)];y34 = [y34; real(sfy)];z34 = [z34; abs(Zc)];
   In34 = [In34; [sfx sfy Zc]];Inb34 = [Inb34; [sfx sfy Zi]];
case 35
   x35 = [x35; real(sfx)];y35 = [y35; real(sfy)];z35 = [z35; abs(Zc)];
   In35 = [In35; [sfx sfy Zc]];Inb35 = [Inb35; [sfx sfy Zi]];
case 36
   x36 = [x36; real(sfx)];y36 = [y36; real(sfy)];z36 = [z36; abs(Zc)];
   In36 = [In36; [sfx sfy Zc]];Inb36 = [Inb36; [sfx sfy Zi]];
otherwise
   disp('ERROR ... Source count out of range!');
   break;
```

```
end
        xt = [xt; real(sfx)]; yt = [yt; real(sfy)]; zt = [zt; abs(Zc)];
        InT = [InT; [sfx sfy Zc]];InB = [InB; [sfx sfy Zi]];
end
r = nx;c = nx;
switch d
    case 1
        if(showEwald == 1)
        figure('Name', 'Ewald Circle for Source #1');scatter3(x1,y1,z1,5,z1,'filled');
        view(0,90);colormap(jet);colorbar;
        end
        fprintf('\n\nProcessing Files for Source #1  .');statuspt = nx/10;percentdone = 0;
        for ix=1:nx
            count = count+1;
            if(count >= statuspt)
                fprintf('.');count = 0;percentdone = percentdone+1;
                if(percentdone == 2)
                    fprintf('25%%')
                end
                if(percentdone == 5)
                    fprintf('50%%');
                end
                if(percentdone == 7)
                    fprintf('.75%%')
                end
            end
            for iy=1:ny
                vsi1(ix,iy) = sum(      In1(:,3) .* exp(i.*2.0.*pi.* ...
                    (In1(:,1).*(dx.*(ix-nx/2-1)+sx)+ In1(:,2).*(dy.*(iy-ny/2-1)+sy))));
                vsib1(ix,iy) = sum(      Inb1(:,3) .* exp(i.*2.0.*pi.* ...
                    (Inb1(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb1(:,2).*(dy.*(iy-ny/2-1)+sy))));
            end
        end
        clear In1;fprintf('.. Done\n');
        if (showBorn == 1)
        figure('Name', 'Born VSi for Source #1');
        surf(abs(vsi1));shading flat;colormap(jet); view(0,-90);colorbar
        end
        for ccc = 1:c/2
            for rr = 1:r/2
                vsi1c(rr,ccc) = vsi1(rr*2,ccc*2);vsib1c(rr,ccc) = vsib1(rr*2,ccc*2);
            end;
        end;
        [r1,c1] = size(vsi1c);
        VSI1 = fftshift(fft2(fftshift(vsi1c)));VSIB1 = fftshift(fft2(fftshift(vsib1c)));
        AbsVSI1 = abs(VSI1);AbsVSIB1 = abs(VSIB1);
```

```
MaxAbsVSI1 = max(max(AbsVSI1));MaxAbsVSIB1 = max(max(AbsVSIB1));
NewSize = r1*2;NewMidPoint = (NewSize/2) + 1;
RefRowPos = 1;RefColPos = 1;
RefVal1 = (MaxAbsVSI1+RefBias)*RefMult;
RefValB1 = (MaxAbsVSIB1+RefBias)*RefMult;
VSI1Causal = zeros(NewSize,NewSize);
VSIB1Causal = zeros(NewSize,NewSize);
VSI1Causal(1:r1,NewMidPoint:NewSize) = VSI1;
VSIB1Causal(1:r1,NewMidPoint:NewSize) = VSIB1;
VSI1Causal(RefRowPos,RefColPos) = RefVal1;
VSIB1Causal(RefRowPos,RefColPos) = RefValB1;
vsi1Ref = ifftshift(ifft2(ifftshift(VSI1Causal)));
vsi1Phase = atan2(imag(vsi1Ref),real(vsi1Ref));
vsib1Ref = ifftshift(ifft2(ifftshift(VSIB1Causal)));
logvsi1Ref = log(vsi1Ref);
logvsib1Ref = log(vsib1Ref);
G1 = fftshift(fft2(fftshift(logvsi1Ref)));
GB1 = fftshift(fft2(fftshift(logvsib1Ref)));
GB11 = zeros(nx,ny); GB12 = zeros(nx,ny);GB13 = zeros(nx,ny);
GB14 = zeros(nx,ny);GB15 = zeros(nx,ny);GB16 = zeros(nx,ny);
GB17 = zeros(nx,ny);GB18 = zeros(nx,ny);GB19 = zeros(nx,ny);
for xg=1:1:nx
   for yg=1:1:nx
   GB11(xg,yg) = 1.*GB1(xg,yg);GB12(xg,yg) = .2.*GB1(xg,yg);
   GB13(xg,yg) = .3.*GB1(xg,yg);GB14(xg,yg) = .4.*GB1(xg,yg);
   GB15(xg,yg) = .5.*GB1(xg,yg);GB16(xg,yg) = .6.*GB1(xg,yg);
   GB17(xg,yg) = .7.*GB1(xg,yg);GB18(xg,yg) = .8.*GB1(xg,yg);
   GB19(xg,yg) = .9.*GB1(xg,yg);
   end
end
ux = 0;uy = 0;[x,y] = meshgrid(-(r1-0.5):(r1-0.5));
Gaussian = 1*(1/(Sigma*sqrt(2*pi)) ) * exp( - ( ( (x-ux).^2 + (y-
uy).^2)./(2*(Sigma).^2)));
MaxGF = max(max(Gaussian));
if Peak == 0
   Peak = MaxGF;
end
Gaussian = Peak*(1/MaxGF)*Gaussian;fp = [];
for xf=1:nx
   fPloty = Gaussian(xf,nx/2);fp = [fp; fPloty];
end
figure('Name', 'Filter');plot(fp);
G11 = Gaussian.*G1;G1Tc = G11;
GTc1 = Gaussian.*(G1-GB11);GTc2 = Gaussian.*(G1-GB12);
GTc3 = Gaussian.*(G1-GB13);GTc4 = Gaussian.*(G1-GB14);
GTc5 = Gaussian.*(G1-GB15);GTc6 = Gaussian.*(G1-GB16);
```

```
   GTc7 = Gaussian.*(G1-GB17);GTc8 = Gaussian.*(G1-GB18);
   GTc9 = Gaussian.*(G1-GB19);
   logv1 = ifftshift(ifft2(ifftshift(G11))); v1 = exp(logv1);vTi = v1;
   if(showCep ==1)
   figure('Name', 'Reconstructed V #1');
   surf(abs(v1));shading flat;colormap(jet); view(0,-90);colorbar
   end
   clear logv1;clear logvsi1Ref;clear VSI1Causal;
case 2
   if(showEwald == 1)
   figure('Name', 'Ewald Circle for Source #2');scatter3(x2,y2,z2,5,z2,'filled');
   view(0,90);colormap(jet);colorbar;
   end
   fprintf('\nProcessing Files for Source #2  .');statuspt = nx/10;percentdone = 0;
   for ix=1:nx
      count = count+1;
      if(count >= statuspt)
         fprintf('.');count = 0;percentdone = percentdone+1;
         if(percentdone == 2)
            fprintf('25%%')
         end
         if(percentdone == 5)
            fprintf('50%%');
         end
         if(percentdone == 7)
            fprintf('.75%%')
         end
      end
      for iy=1:ny
         vsi2(ix,iy) = sum(      In2(:,3) .* exp(i.*2.0.*pi.* ...
            (In2(:,1).*(dx.*(ix-nx/2-1)+sx)+ In2(:,2).*(dy.*(iy-ny/2-1)+sy))));
         vsib2(ix,iy) = sum(      Inb2(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb2(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb2(:,2).*(dy.*(iy-ny/2-1)+sy))));
      end
   end
   clear In2;fprintf('. Done\n');
   if (showBorn == 1)
   figure('Name', 'Born VSi for Source #2');
   surf(abs(vsi2));shading flat;colormap(jet); view(0,-90);colorbar
   end
   for ccc = 1:c/2
      for rr = 1:r/2
         vsi2c(rr,ccc) = vsi2(rr*2,ccc*2);vsib2c(rr,ccc) = vsib2(rr*2,ccc*2);
      end;
   end;
   [r1,c1] = size(vsi2c);
```

```
    VSI2 = fftshift(fft2(fftshift(vsi2c)));VSIB2 = fftshift(fft2(fftshift(vsib2c)));
    AbsVSI2 = abs(VSI2);AbsVSIB2 = abs(VSIB2);
    MaxAbsVSI2 = max(max(AbsVSI2));MaxAbsVSIB2 = max(max(AbsVSIB2));
    RefVal2 = (MaxAbsVSI2+RefBias)*RefMult;
    RefValB2 = (MaxAbsVSIB2+RefBias)*RefMult;
    VSI2Causal = zeros(NewSize,NewSize); %
    VSIB2Causal = zeros(NewSize,NewSize);
    VSI2Causal(1:r1,NewMidPoint:NewSize) = VSI2;
    VSIB2Causal(1:r1,NewMidPoint:NewSize) = VSIB2;
    VSI2Causal(RefRowPos,RefColPos) = RefVal2;
    VSIB2Causal(RefRowPos,RefColPos) = RefValB2;
    vsi2Ref = ifftshift(ifft2(ifftshift(VSI2Causal)));
    vsi2Phase = atan2(imag(vsi2Ref),real(vsi2Ref));
    vsib2Ref = ifftshift(ifft2(ifftshift(VSIB2Causal)));
    logvsi2Ref = log(vsi2Ref);
    logvsib2Ref = log(vsib2Ref);
    G2 = fftshift(fft2(fftshift(logvsi2Ref))); % Taking FFT to go to Cepstral domain
    GB2 = fftshift(fft2(fftshift(logvsib2Ref)));
    GB21 = zeros(nx,ny);GB22 = zeros(nx,ny);GB23 = zeros(nx,ny);
    GB24 = zeros(nx,ny);GB25 = zeros(nx,ny);GB26 = zeros(nx,ny);
    GB27 = zeros(nx,ny);GB28 = zeros(nx,ny);GB29 = zeros(nx,ny);
    for xg=1:1:nx
       for yg=1:1:nx
       GB21(xg,yg) = 1.*GB2(xg,yg);GB22(xg,yg) = .2.*GB2(xg,yg);
       GB23(xg,yg) = .3.*GB2(xg,yg);GB24(xg,yg) = .4.*GB2(xg,yg);
       GB25(xg,yg) = .5.*GB2(xg,yg);GB26(xg,yg) = .6.*GB2(xg,yg);
       GB27(xg,yg) = .7.*GB2(xg,yg);GB28(xg,yg) = .8.*GB2(xg,yg);
       GB29(xg,yg) = .9.*GB2(xg,yg);
       end
    end
    G12 = Gaussian.*G2;G1Tc = G1Tc + G12;
    GTc1 = GTc1 + Gaussian.*(G2-GB21);GTc2 = GTc2 + Gaussian.*(G2-GB22);
    GTc3 = GTc3 + Gaussian.*(G2-GB23);GTc4 = GTc4 + Gaussian.*(G2-GB24);
    GTc5 = GTc5 + Gaussian.*(G2-GB25);GTc6 = GTc6 + Gaussian.*(G2-GB26);
    GTc7 = GTc7 + Gaussian.*(G2-GB27);GTc8 = GTc8 + Gaussian.*(G2-GB28);
    GTc9 = GTc9 + Gaussian.*(G2-GB29);
    logv2 = ifftshift(ifft2(ifftshift(G12)));v2 = exp(logv2);vTi = vTi + v2;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #2');
    surf(abs(v2));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv2;clear logvsi2Ref;clear VSI2Causal;
 case 3
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #3');scatter3(x3,y3,z3,5,z3,'filled');
    view(0,90);colormap(jet);colorbar;
```

```
end
fprintf('\nProcessing Files for Source #3  .');statuspt = nx/10;percentdone = 0;
for ix=1:nx
   count = count+1;
   if(count >= statuspt)
      fprintf('.');count = 0;percentdone = percentdone+1;
      if(percentdone == 2)
         fprintf('25%%')
      end
      if(percentdone == 5)
         fprintf('50%%');
      end
      if(percentdone == 7)
         fprintf('.75%%')
      end
   end
   for iy=1:ny
      vsi3(ix,iy) = sum(      In3(:,3) .* exp(i.*2.0.*pi.* ...
         (In3(:,1).*(dx.*(ix-nx/2-1)+sx)+ In3(:,2).*(dy.*(iy-ny/2-1)+sy))));
      vsib3(ix,iy) = sum(      Inb3(:,3) .* exp(i.*2.0.*pi.* ...
         (Inb3(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb3(:,2).*(dy.*(iy-ny/2-1)+sy))));
   end
end
clear In3;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #3');
surf(abs(vsi3));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
   for rr = 1:r/2
      vsi3c(rr,ccc) = vsi3(rr*2,ccc*2);vsib3c(rr,ccc) = vsib3(rr*2,ccc*2);
   end;
end;
[r1,c1] = size(vsi3c);
VSI3 = fftshift(fft2(fftshift(vsi3c)));VSIB3 = fftshift(fft2(fftshift(vsib3c)));
AbsVSI3 = abs(VSI3);AbsVSIB3 = abs(VSIB3);
MaxAbsVSI3 = max(max(AbsVSI3));MaxAbsVSIB3 = max(max(AbsVSIB3));
RefVal3 = (MaxAbsVSI3+RefBias)*RefMult;
RefValB3 = (MaxAbsVSIB3+RefBias)*RefMult;
VSI3Causal = zeros(NewSize,NewSize);
VSIB3Causal = zeros(NewSize,NewSize);
VSI3Causal(1:r1,NewMidPoint:NewSize) = VSI3;
VSIB3Causal(1:r1,NewMidPoint:NewSize) = VSIB3;
VSI3Causal(RefRowPos,RefColPos) = RefVal3;
VSIB3Causal(RefRowPos,RefColPos) = RefValB3;
vsi3Ref = ifftshift(ifft2(ifftshift(VSI3Causal)));
```

```
    vsi3Phase = atan2(imag(vsi3Ref),real(vsi3Ref));
    vsib3Ref = ifftshift(ifft2(ifftshift(VSIB3Causal)));
    logvsi3Ref = log(vsi3Ref); logvsib3Ref = log(vsib3Ref);
    G3 = fftshift(fft2(fftshift(logvsi3Ref)));GB3 = fftshift(fft2(fftshift(logvsib3Ref)));
    GB31 = zeros(nx,ny);GB32 = zeros(nx,ny);GB33 = zeros(nx,ny);
    GB34 = zeros(nx,ny);GB35 = zeros(nx,ny);GB36 = zeros(nx,ny);
    GB37 = zeros(nx,ny);GB38 = zeros(nx,ny);GB39 = zeros(nx,ny);
    for xg=1:1:nx
        for yg=1:1:nx
        GB31(xg,yg) = 1.*GB3(xg,yg);GB32(xg,yg) = .2.*GB3(xg,yg);
        GB33(xg,yg) = .3.*GB3(xg,yg);GB34(xg,yg) = .4.*GB3(xg,yg);
        GB35(xg,yg) = .5.*GB3(xg,yg);GB36(xg,yg) = .6.*GB3(xg,yg);
        GB37(xg,yg) = .7.*GB3(xg,yg);GB38(xg,yg) = .8.*GB3(xg,yg);
        GB39(xg,yg) = .9.*GB3(xg,yg);
        end
    end
    G13 = Gaussian.*G3;G1Tc = G1Tc + G13;
    GTc1 = GTc1 + Gaussian.*(G3-GB31);GTc2 = GTc2 + Gaussian.*(G3-GB32);
    GTc3 = GTc3 + Gaussian.*(G3-GB33);GTc4 = GTc4 + Gaussian.*(G3-GB34);
    GTc5 = GTc5 + Gaussian.*(G3-GB35);GTc6 = GTc6 + Gaussian.*(G3-GB36);
    GTc7 = GTc7 + Gaussian.*(G3-GB37);GTc8 = GTc8 + Gaussian.*(G3-GB38);
    GTc9 = GTc9 + Gaussian.*(G3-GB39);
    logv3 = ifftshift(ifft2(ifftshift(G13)));v3 = exp(logv3);vTi = vTi + v3;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #3');
    surf(abs(v3));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv3;clear logvsi3Ref;clear VSI3Causal;
case 4
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #4');scatter3(x4,y4,z4,5,z4,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #4  .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi4(ix,iy) = sum(      In4(:,3) .* exp(i.*2.0.*pi.* ...
                (In4(:,1).*(dx.*(ix-nx/2-1)+sx)+ In4(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib4(ix,iy) = sum(      Inb4(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb4(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb4(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In4;           fprintf('. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #4');
    surf(abs(vsi4));shading flat;colormap(jet); view(0,-90);colorbar
    end
    for ccc = 1:c/2
        for rr = 1:r/2
            vsi4c(rr,ccc) = vsi4(rr*2,ccc*2);vsib4c(rr,ccc) = vsib4(rr*2,ccc*2);
        end;
    end;
    [r1,c1] = size(vsi4c);
    VSI4 = fftshift(fft2(fftshift(vsi4c)));VSIB4 = fftshift(fft2(fftshift(vsib4c)));
    AbsVSI4 = abs(VSI4);AbsVSIB4 = abs(VSIB4);
    MaxAbsVSI4 = max(max(AbsVSI4));MaxAbsVSIB4 = max(max(AbsVSIB4));
    RefVal4 = (MaxAbsVSI4+RefBias)*RefMult;
    RefValB4 = (MaxAbsVSIB4+RefBias)*RefMult;
    VSI4Causal = zeros(NewSize,NewSize); %
    VSIB4Causal = zeros(NewSize,NewSize);
    VSI4Causal(1:r1,NewMidPoint:NewSize) = VSI4;
    VSIB4Causal(1:r1,NewMidPoint:NewSize) = VSIB4;
    VSI4Causal(RefRowPos,RefColPos) = RefVal4;
    VSIB4Causal(RefRowPos,RefColPos) = RefValB4;
    vsi4Ref = ifftshift(ifft2(ifftshift(VSI4Causal)));
    vsi4Phase = atan2(imag(vsi4Ref),real(vsi4Ref));
    vsib4Ref = ifftshift(ifft2(ifftshift(VSIB4Causal)));
    logvsi4Ref = log(vsi4Ref);logvsib4Ref = log(vsib4Ref);
    G4 = fftshift(fft2(fftshift(logvsi4Ref)));GB4 = fftshift(fft2(fftshift(logvsib4Ref)));
    GB41 = zeros(nx,ny);GB42 = zeros(nx,ny);GB43 = zeros(nx,ny);
    GB44 = zeros(nx,ny);GB45 = zeros(nx,ny);GB46 = zeros(nx,ny);
    GB47 = zeros(nx,ny);GB48 = zeros(nx,ny);GB49 = zeros(nx,ny);
    for xg=1:1:nx
        for yg=1:1:nx
        GB41(xg,yg) = 1.*GB4(xg,yg);GB42(xg,yg) = .2.*GB4(xg,yg);
        GB43(xg,yg) = .3.*GB4(xg,yg);GB44(xg,yg) = .4.*GB4(xg,yg);
        GB45(xg,yg) = .5.*GB4(xg,yg);GB46(xg,yg) = .6.*GB4(xg,yg);
        GB47(xg,yg) = .7.*GB4(xg,yg);GB48(xg,yg) = .8.*GB4(xg,yg);
        GB49(xg,yg) = .9.*GB4(xg,yg);
```

```
        end
    end
    G14 = Gaussian.*G4;G1Tc = G1Tc + G14;
    GTc1 = GTc1 + Gaussian.*(G4-GB41);GTc2 = GTc2 + Gaussian.*(G4-GB42);
    GTc3 = GTc3 + Gaussian.*(G4-GB43);GTc4 = GTc4 + Gaussian.*(G4-GB44);
    GTc5 = GTc5 + Gaussian.*(G4-GB45);GTc6 = GTc6 + Gaussian.*(G4-GB46);
    GTc7 = GTc7 + Gaussian.*(G4-GB47);GTc8 = GTc8 + Gaussian.*(G4-GB48);
    GTc9 = GTc9 + Gaussian.*(G4-GB49);
    logv4 = ifftshift(ifft2(ifftshift(G14)));v4 = exp(logv4);vTi = vTi + v4;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #4');
    surf(abs(v4));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv4;clear logvsi4Ref;clear VSI4Causal;
case 5
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #5');scatter3(x5,y5,z5,5,z5,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #5  .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi5(ix,iy) = sum(      In5(:,3) .* exp(i.*2.0.*pi.* ...
                (In5(:,1).*(dx.*(ix-nx/2-1)+sx)+ In5(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib5(ix,iy) = sum(      Inb5(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb5(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb5(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In5;            fprintf('. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #5');
    surf(abs(vsi5));shading flat;colormap(jet); view(0,-90);colorbar
    end
```

```
for ccc = 1:c/2
   for rr = 1:r/2
      vsi5c(rr,ccc) = vsi5(rr*2,ccc*2);vsib5c(rr,ccc) = vsib5(rr*2,ccc*2);
   end;
end;
[r1,c1] = size(vsi5c);
VSI5 = fftshift(fft2(fftshift(vsi5c)));VSIB5 = fftshift(fft2(fftshift(vsib5c)));
AbsVSI5 = abs(VSI5);AbsVSIB5 = abs(VSIB5);
MaxAbsVSI5 = max(max(AbsVSI5));MaxAbsVSIB5 = max(max(AbsVSIB5));
RefVal5 = (MaxAbsVSI5+RefBias)*RefMult;
RefValB5 = (MaxAbsVSIB5+RefBias)*RefMult;
VSI5Causal = zeros(NewSize,NewSize); %
VSIB5Causal = zeros(NewSize,NewSize);
VSI5Causal(1:r1,NewMidPoint:NewSize) = VSI5;
VSIB5Causal(1:r1,NewMidPoint:NewSize) = VSIB5;
VSI5Causal(RefRowPos,RefColPos) = RefVal5;
VSIB5Causal(RefRowPos,RefColPos) = RefValB5;
vsi5Ref = ifftshift(ifft2(ifftshift(VSI5Causal)));
vsi5Phase = atan2(imag(vsi5Ref),real(vsi5Ref));
vsib5Ref = ifftshift(ifft2(ifftshift(VSIB5Causal)));
logvsi5Ref = log(vsi5Ref); logvsib5Ref = log(vsib5Ref);
G5 = fftshift(fft2(fftshift(logvsi5Ref)));GB5 = fftshift(fft2(fftshift(logvsib5Ref)));
GB51 = zeros(nx,ny);GB52 = zeros(nx,ny);GB53 = zeros(nx,ny);
GB54 = zeros(nx,ny);GB55 = zeros(nx,ny);GB56 = zeros(nx,ny);
GB57 = zeros(nx,ny);GB58 = zeros(nx,ny);GB59 = zeros(nx,ny);
for xg=1:1:nx
   for yg=1:1:nx
   GB51(xg,yg) = 1.*GB5(xg,yg);GB52(xg,yg) = .2.*GB5(xg,yg);
   GB53(xg,yg) = .3.*GB5(xg,yg);GB54(xg,yg) = .4.*GB5(xg,yg);
   GB55(xg,yg) = .5.*GB5(xg,yg);GB56(xg,yg) = .6.*GB5(xg,yg);
   GB57(xg,yg) = .7.*GB5(xg,yg);GB58(xg,yg) = .8.*GB5(xg,yg);
   GB59(xg,yg) = .9.*GB5(xg,yg);
   end
end
G15 = Gaussian.*G5; G1Tc = G1Tc + G15;
GTc1 = GTc1 + Gaussian.*(G5-GB51);GTc2 = GTc2 + Gaussian.*(G5-GB52);
GTc3 = GTc3 + Gaussian.*(G5-GB53);GTc4 = GTc4 + Gaussian.*(G5-GB54);
GTc5 = GTc5 + Gaussian.*(G5-GB55);GTc6 = GTc6 + Gaussian.*(G5-GB56);
GTc7 = GTc7 + Gaussian.*(G5-GB57);GTc8 = GTc8 + Gaussian.*(G5-GB58);
GTc9 = GTc9 + Gaussian.*(G5-GB59);
logv5 = ifftshift(ifft2(ifftshift(G15))); v5 = exp(logv5); vTi = vTi + v5;
if(showCep ==1)
figure('Name', 'Reconstructed V #5');
surf(abs(v5));shading flat;colormap(jet); view(0,-90);colorbar
end
clear logv5;clear logvsi5Ref;clear VSI5Causal;
```

```
case 6
   if(showEwald == 1)
   figure('Name', 'Ewald Circle for Source #6');scatter3(x6,y6,z6,5,z6,'filled');
   view(0,90);colormap(jet);colorbar;
   end
   fprintf('\nProcessing Files for Source #6  .');statuspt = nx/10;percentdone = 0;
   for ix=1:nx
      count = count+1;
      if(count >= statuspt)
         fprintf('.');count = 0;percentdone = percentdone+1;
         if(percentdone == 2)
            fprintf('25%%')
         end
         if(percentdone == 5)
            fprintf('50%%');
         end
         if(percentdone == 7)
            fprintf('.75%%')
         end
      end
      for iy=1:ny
         vsi6(ix,iy) = sum(      In6(:,3) .* exp(i.*2.0.*pi.* ...
            (In6(:,1).*(dx.*(ix-nx/2-1)+sx)+ In6(:,2).*(dy.*(iy-ny/2-1)+sy))));
         vsib6(ix,iy) = sum(      Inb6(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb6(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb6(:,2).*(dy.*(iy-ny/2-1)+sy))));
      end
   end
   clear In6;fprintf('. Done\n');
   if (showBorn == 1)
   figure('Name', 'Born VSi for Source #6');
   surf(abs(vsi6));shading flat;colormap(jet); view(0,-90);colorbar
   end
   for ccc = 1:c/2
      for rr = 1:r/2
         vsi6c(rr,ccc) = vsi6(rr*2,ccc*2);vsib6c(rr,ccc) = vsib6(rr*2,ccc*2);
      end;
   end;
   [r1,c1] = size(vsi6c);
   VSI6 = fftshift(fft2(fftshift(vsi6c)));VSIB6 = fftshift(fft2(fftshift(vsib6c)));
   AbsVSI6 = abs(VSI6);AbsVSIB6 = abs(VSIB6);
   MaxAbsVSI6 = max(max(AbsVSI6));MaxAbsVSIB6 = max(max(AbsVSIB6));
   RefVal6 = (MaxAbsVSI6+RefBias)*RefMult;
   RefValB6 = (MaxAbsVSIB6+RefBias)*RefMult;
   VSI6Causal = zeros(NewSize,NewSize);
   VSIB6Causal = zeros(NewSize,NewSize);
   VSI6Causal(1:r1,NewMidPoint:NewSize) = VSI6;
```

```
    VSIB6Causal(1:r1,NewMidPoint:NewSize) = VSIB6;
    VSI6Causal(RefRowPos,RefColPos) = RefVal6;
    VSIB6Causal(RefRowPos,RefColPos) = RefValB6;
    vsi6Ref = ifftshift(ifft2(ifftshift(VSI6Causal)));
    vsi6Phase = atan2(imag(vsi6Ref),real(vsi6Ref));
    vsib6Ref = ifftshift(ifft2(ifftshift(VSIB6Causal)));
    logvsi6Ref = log(vsi6Ref); ogvsib6Ref = log(vsib6Ref);
    G6 = fftshift(fft2(fftshift(logvsi6Ref)));GB6 = fftshift(fft2(fftshift(logvsib6Ref)));
    GB61 = zeros(nx,ny);GB62 = zeros(nx,ny);GB63 = zeros(nx,ny);
    GB64 = zeros(nx,ny);GB65 = zeros(nx,ny);GB66 = zeros(nx,ny);
    GB67 = zeros(nx,ny);GB68 = zeros(nx,ny);GB69 = zeros(nx,ny);
    for xg=1:1:nx
       for yg=1:1:nx
       GB61(xg,yg) = 1.*GB6(xg,yg);GB62(xg,yg) = .2.*GB6(xg,yg);
       GB63(xg,yg) = .3.*GB6(xg,yg);GB64(xg,yg) = .4.*GB6(xg,yg);
       GB65(xg,yg) = .5.*GB6(xg,yg);GB66(xg,yg) = .6.*GB6(xg,yg);
       GB67(xg,yg) = .7.*GB6(xg,yg);GB68(xg,yg) = .8.*GB6(xg,yg);
       GB69(xg,yg) = .9.*GB6(xg,yg);
      end
    end
    G16 = Gaussian.*G6;G1Tc = G1Tc + G16;
    GTc1 = GTc1 + Gaussian.*(G6-GB61);GTc2 = GTc2 + Gaussian.*(G6-GB62);
    GTc3 = GTc3 + Gaussian.*(G6-GB63);GTc4 = GTc4 + Gaussian.*(G6-GB64);
    GTc5 = GTc5 + Gaussian.*(G6-GB65);GTc6 = GTc6 + Gaussian.*(G6-GB66);
    GTc7 = GTc7 + Gaussian.*(G6-GB67);GTc8 = GTc8 + Gaussian.*(G6-GB68);
    GTc9 = GTc9 + Gaussian.*(G6-GB69);
    logv6 = ifftshift(ifft2(ifftshift(G16))); v6 = exp(logv6); vTi = vTi + v6;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #6');
    surf(abs(v6));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv6; clear logvsi6Ref; clear VSI6Causal;
case 7
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #7');scatter3(x7,y7,z7,5,z7,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #7  .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
       count = count+1;
       if(count >= statuspt)
          fprintf('.');count = 0;percentdone = percentdone+1;
          if(percentdone == 2)
             fprintf('25%%')
          end
          if(percentdone == 5)
```

```
            fprintf('50%%');
        end
        if(percentdone == 7)
            fprintf('.75%%')
        end
    end
    for iy=1:ny
        vsi7(ix,iy) = sum(      In7(:,3) .* exp(i.*2.0.*pi.* ...
            (In7(:,1).*(dx.*(ix-nx/2-1)+sx)+ In7(:,2).*(dy.*(iy-ny/2-1)+sy))));
                    vsib7(ix,iy) = sum(      Inb7(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb7(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb7(:,2).*(dy.*(iy-ny/2-1)+sy))));
    end
end
clear In7;fprintf('.. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #7');
surf(abs(vsi7));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
    for rr = 1:r/2
        vsi7c(rr,ccc) = vsi7(rr*2,ccc*2);vsib7c(rr,ccc) = vsib7(rr*2,ccc*2);
    end;
end;
[r1,c1] = size(vsi7c);

VSI7 = fftshift(fft2(fftshift(vsi7c)));VSIB7 = fftshift(fft2(fftshift(vsib7c)));
AbsVSI7 = abs(VSI7);AbsVSIB7 = abs(VSIB7);
MaxAbsVSI7 = max(max(AbsVSI7));MaxAbsVSIB7 = max(max(AbsVSIB7));
RefVal7 = (MaxAbsVSI7+RefBias)*RefMult;
RefValB7 = (MaxAbsVSIB7+RefBias)*RefMult;
VSI7Causal = zeros(NewSize,NewSize); %
VSIB7Causal = zeros(NewSize,NewSize);
VSI7Causal(1:r1,NewMidPoint:NewSize) = VSI7;
VSIB7Causal(1:r1,NewMidPoint:NewSize) = VSIB7;
VSI7Causal(RefRowPos,RefColPos) = RefVal7;
VSIB7Causal(RefRowPos,RefColPos) = RefValB7;
vsi7Ref = ifftshift(ifft2(ifftshift(VSI7Causal)));
vsi7Phase = atan2(imag(vsi7Ref),real(vsi7Ref));
vsib7Ref = ifftshift(ifft2(ifftshift(VSIB7Causal)));
logvsi7Ref = log(vsi7Ref); logvsib7Ref = log(vsib7Ref);
G7 = fftshift(fft2(fftshift(logvsi7Ref)));GB7 = fftshift(fft2(fftshift(logvsib7Ref)));
GB71 = zeros(nx,ny);GB72 = zeros(nx,ny);GB73 = zeros(nx,ny);
GB74 = zeros(nx,ny);GB75 = zeros(nx,ny);GB76 = zeros(nx,ny);
GB77 = zeros(nx,ny);GB78 = zeros(nx,ny);GB79 = zeros(nx,ny);
for xg=1:1:nx
    for yg=1:1:nx
```

```
      GB71(xg,yg) = 1.*GB7(xg,yg);GB72(xg,yg) = .2.*GB7(xg,yg);
      GB73(xg,yg) = .3.*GB7(xg,yg);GB74(xg,yg) = .4.*GB7(xg,yg);
      GB75(xg,yg) = .5.*GB7(xg,yg);GB76(xg,yg) = .6.*GB7(xg,yg);
      GB77(xg,yg) = .7.*GB7(xg,yg);GB78(xg,yg) = .8.*GB7(xg,yg);
      GB79(xg,yg) = .9.*GB7(xg,yg);
     end
   end
   G17 = Gaussian.*G7;  G1Tc = G1Tc + G17;
   GTc1 = GTc1 + Gaussian.*(G7-GB71);GTc2 = GTc2 + Gaussian.*(G7-GB72);
   GTc3 = GTc3 + Gaussian.*(G7-GB73);GTc4 = GTc4 + Gaussian.*(G7-GB74);
   GTc5 = GTc5 + Gaussian.*(G7-GB75);GTc6 = GTc6 + Gaussian.*(G7-GB76);
   GTc7 = GTc7 + Gaussian.*(G7-GB77);GTc8 = GTc8 + Gaussian.*(G7-GB78);
   GTc9 = GTc9 + Gaussian.*(G7-GB79);
   logv7 = ifftshift(ifft2(ifftshift(G17))); v7 = exp(logv7); vTi = vTi + v7;
   if(showCep ==1)
   figure('Name', 'Reconstructed V #7');
   surf(abs(v7));shading flat;colormap(jet); view(0,-90);colorbar
   end
   clear logv7; clear logvsi7Ref; clear VSI7Causal;
case 8
   if(showEwald == 1)
   figure('Name', 'Ewald Circle for Source #8');scatter3(x8,y8,z8,5,z8,'filled');
   view(0,90);colormap(jet);colorbar;
   end
   fprintf('\nProcessing Files for Source #8  .');statuspt = nx/10;percentdone = 0;
   for ix=1:nx
      count = count+1;
      if(count >= statuspt)
         fprintf('.');count = 0;percentdone = percentdone+1;
         if(percentdone == 2)
            fprintf('25%%')
         end
         if(percentdone == 5)
            fprintf('50%%');
         end
         if(percentdone == 7)
            fprintf('.75%%')
         end
      end
      for iy=1:ny
         vsi8(ix,iy) = sum(      In8(:,3) .* exp(i.*2.0.*pi.* ...
            (In8(:,1).*(dx.*(ix-nx/2-1)+sx)+ In8(:,2).*(dy.*(iy-ny/2-1)+sy))));
                        vsib8(ix,iy) = sum(      Inb8(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb8(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb8(:,2).*(dy.*(iy-ny/2-1)+sy))));
      end
   end
```

```
clear In8;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #8');
surf(abs(vsi8));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
   for rr = 1:r/2
      vsi8c(rr,ccc) = vsi8(rr*2,ccc*2);vsib8c(rr,ccc) = vsib8(rr*2,ccc*2);
   end;
end;
[r1,c1] = size(vsi8c);
VSI8 = fftshift(fft2(fftshift(vsi8c)));VSIB8 = fftshift(fft2(fftshift(vsib8c)));
AbsVSI8 = abs(VSI8);AbsVSIB8 = abs(VSIB8);
MaxAbsVSI8 = max(max(AbsVSI8));MaxAbsVSIB8 = max(max(AbsVSIB8));
RefVal8 = (MaxAbsVSI8+RefBias)*RefMult;
RefValB8 = (MaxAbsVSIB8+RefBias)*RefMult;
VSI8Causal = zeros(NewSize,NewSize); %
VSIB8Causal = zeros(NewSize,NewSize);
VSI8Causal(1:r1,NewMidPoint:NewSize) = VSI8;
VSIB8Causal(1:r1,NewMidPoint:NewSize) = VSIB8;
VSI8Causal(RefRowPos,RefColPos) = RefVal8;
VSIB8Causal(RefRowPos,RefColPos) = RefValB8;
vsi8Ref = ifftshift(ifft2(ifftshift(VSI8Causal)));
vsi8Phase = atan2(imag(vsi8Ref),real(vsi8Ref));
vsib8Ref = ifftshift(ifft2(ifftshift(VSIB8Causal)));

logvsi8Ref = log(vsi8Ref); logvsib8Ref = log(vsib8Ref);
G8 = fftshift(fft2(fftshift(logvsi8Ref)));GB8 = fftshift(fft2(fftshift(logvsib8Ref)));
GB81 = zeros(nx,ny);GB82 = zeros(nx,ny);GB83 = zeros(nx,ny);
GB84 = zeros(nx,ny);GB85 = zeros(nx,ny);GB86 = zeros(nx,ny);
GB87 = zeros(nx,ny);GB88 = zeros(nx,ny);GB89 = zeros(nx,ny);
for xg=1:1:nx
   for yg=1:1:nx
   GB81(xg,yg) = 1.*GB8(xg,yg);GB82(xg,yg) = .2.*GB8(xg,yg);
   GB83(xg,yg) = .3.*GB8(xg,yg);GB84(xg,yg) = .4.*GB8(xg,yg);
   GB85(xg,yg) = .5.*GB8(xg,yg);GB86(xg,yg) = .6.*GB8(xg,yg);
   GB87(xg,yg) = .7.*GB8(xg,yg);GB88(xg,yg) = .8.*GB8(xg,yg);
   GB89(xg,yg) = .9.*GB8(xg,yg);
  end
end
G18 = Gaussian.*G8;  G1Tc = G1Tc + G18;
GTc1 = GTc1 + Gaussian.*(G8-GB81);GTc2 = GTc2 + Gaussian.*(G8-GB82);
GTc3 = GTc3 + Gaussian.*(G8-GB83);GTc4 = GTc4 + Gaussian.*(G8-GB84);
GTc5 = GTc5 + Gaussian.*(G8-GB85);GTc6 = GTc6 + Gaussian.*(G8-GB86);
GTc7 = GTc7 + Gaussian.*(G8-GB87);GTc8 = GTc8 + Gaussian.*(G8-GB88);
GTc9 = GTc9 + Gaussian.*(G8-GB89);
```

```
     logv8 = ifftshift(ifft2(ifftshift(G18))); v8 = exp(logv8);vTi = vTi + v8;
     if(showCep ==1)
    figure('Name', 'Reconstructed V #8');
    surf(abs(v8));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv8; clear logvsi8Ref; VSI8Causal;
case 9
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #9');scatter3(x9,y9,z9,5,z9,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #9  .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi9(ix,iy) = sum(      In9(:,3) .* exp(i.*2.0.*pi.* ...
                (In9(:,1).*(dx.*(ix-nx/2-1)+sx)+ In9(:,2).*(dy.*(iy-ny/2-1)+sy))));
                                vsib9(ix,iy) = sum(      Inb9(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb9(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb9(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In9; fprintf('. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #9');
    surf(abs(vsi9));shading flat;colormap(jet); view(0,-90);colorbar
    end
    for ccc = 1:c/2
        for rr = 1:r/2
            vsi9c(rr,ccc) = vsi9(rr*2,ccc*2);vsib9c(rr,ccc) = vsib9(rr*2,ccc*2);
        end;
    end;
    [r1,c1] = size(vsi9c);
    VSI9 = fftshift(fft2(fftshift(vsi9c)));VSIB9 = fftshift(fft2(fftshift(vsib9c)));
    AbsVSI9 = abs(VSI9);AbsVSIB9 = abs(VSIB9);
    MaxAbsVSI9 = max(max(AbsVSI9));MaxAbsVSIB9 = max(max(AbsVSIB9));
    RefVal9 = (MaxAbsVSI9+RefBias)*RefMult;
    RefValB9 = (MaxAbsVSIB9+RefBias)*RefMult;
    VSI9Causal = zeros(NewSize,NewSize); %
    VSIB9Causal = zeros(NewSize,NewSize);
    VSI9Causal(1:r1,NewMidPoint:NewSize) = VSI9;
    VSIB9Causal(1:r1,NewMidPoint:NewSize) = VSIB9;
    VSI9Causal(RefRowPos,RefColPos) = RefVal9;
    VSIB9Causal(RefRowPos,RefColPos) = RefValB9;
    vsi9Ref = ifftshift(ifft2(ifftshift(VSI9Causal)));
    vsi9Phase = atan2(imag(vsi9Ref),real(vsi9Ref)); % Phase of vsi
    vsib9Ref = ifftshift(ifft2(ifftshift(VSIB9Causal)));
    logvsi9Ref = log(vsi9Ref); logvsib9Ref = log(vsib9Ref);
    G9 = fftshift(fft2(fftshift(logvsi9Ref))); % Taking FFT to go to Cepstral domain
    GB9 = fftshift(fft2(fftshift(logvsib9Ref)));
    GB91 = zeros(nx,ny);GB92 = zeros(nx,ny);GB93 = zeros(nx,ny);
    GB94 = zeros(nx,ny);GB95 = zeros(nx,ny);GB96 = zeros(nx,ny);
    GB97 = zeros(nx,ny);GB98 = zeros(nx,ny);GB99 = zeros(nx,ny);
    for xg=1:1:nx
       for yg=1:1:nx
       GB91(xg,yg) = 1.*GB9(xg,yg);GB92(xg,yg) = .2.*GB9(xg,yg);
       GB93(xg,yg) = .3.*GB9(xg,yg);GB94(xg,yg) = .4.*GB9(xg,yg);
       GB95(xg,yg) = .5.*GB9(xg,yg);GB96(xg,yg) = .6.*GB9(xg,yg);
       GB97(xg,yg) = .7.*GB9(xg,yg);GB98(xg,yg) = .8.*GB9(xg,yg);
       GB99(xg,yg) = .9.*GB9(xg,yg);
      end
    end
    G19 = Gaussian.*G9; G1Tc = G1Tc + G19;
    GTc1 = GTc1 + Gaussian.*(G9-GB91);GTc2 = GTc2 + Gaussian.*(G9-GB92);
    GTc3 = GTc3 + Gaussian.*(G9-GB93);GTc4 = GTc4 + Gaussian.*(G9-GB94);
    GTc5 = GTc5 + Gaussian.*(G9-GB95);GTc6 = GTc6 + Gaussian.*(G9-GB96);
    GTc7 = GTc7 + Gaussian.*(G9-GB97);GTc8 = GTc8 + Gaussian.*(G9-GB98);
    GTc9 = GTc9 + Gaussian.*(G9-GB99);
    logv9 = ifftshift(ifft2(ifftshift(G19))); v9 = exp(logv9); vTi = vTi + v9;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #9');
    surf(abs(v9));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv9; clear logvsi9Ref; clear VSI9Causal;
case 10
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #10');scatter3(x10,y10,z10,5,z10,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #10 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
```

```
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi10(ix,iy) = sum(     In10(:,3) .* exp(i.*2.0.*pi.* ...
                (In10(:,1).*(dx.*(ix-nx/2-1)+sx)+ In10(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib10(ix,iy) = sum(     Inb10(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb10(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb10(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In10;fprintf('. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #10');
    surf(abs(vsi10));shading flat;colormap(jet); view(0,-90);colorbar
    end
    for ccc = 1:c/2
        for rr = 1:r/2
            vsi10c(rr,ccc) = vsi10(rr*2,ccc*2);vsib10c(rr,ccc) = vsib10(rr*2,ccc*2);
        end;
    end;
    [r1,c1] = size(vsi10c);
    VSI10 = fftshift(fft2(fftshift(vsi10c)));VSIB10 = fftshift(fft2(fftshift(vsib10c)));
    AbsVSI10 = abs(VSI10);AbsVSIB10 = abs(VSIB10);
    MaxAbsVSI10 = max(max(AbsVSI10));
    MaxAbsVSIB10 = max(max(AbsVSIB10));
    RefVal10 = (MaxAbsVSI10+RefBias)*RefMult;
    RefValB10 = (MaxAbsVSIB10+RefBias)*RefMult;
    VSI10Causal = zeros(NewSize,NewSize); %
    VSIB10Causal = zeros(NewSize,NewSize);
    VSI10Causal(1:r1,NewMidPoint:NewSize) = VSI10;
    VSIB10Causal(1:r1,NewMidPoint:NewSize) = VSIB10;
    VSI10Causal(RefRowPos,RefColPos) = RefVal10;
    VSIB10Causal(RefRowPos,RefColPos) = RefValB10;
    vsi10Ref = ifftshift(ifft2(ifftshift(VSI10Causal)));
    vsi10Phase = atan2(imag(vsi10Ref),real(vsi10Ref)); % Phase of vsi
    vsib10Ref = ifftshift(ifft2(ifftshift(VSIB10Causal)));
    logvsi10Ref = log(vsi10Ref);
    logvsib10Ref = log(vsib10Ref);
   G10 = fftshift(fft2(fftshift(logvsi10Ref)));
   GB10 = fftshift(fft2(fftshift(logvsib10Ref)));
    GB101 = zeros(nx,ny);GB102 = zeros(nx,ny);GB103 = zeros(nx,ny);
    GB104 = zeros(nx,ny);GB105 = zeros(nx,ny);GB106 = zeros(nx,ny);
    GB107 = zeros(nx,ny);GB108 = zeros(nx,ny);GB109 = zeros(nx,ny);
    for xg=1:1:nx
        for yg=1:1:nx
        GB101(xg,yg) = 1.*GB10(xg,yg);GB102(xg,yg) = .2.*GB10(xg,yg);
        GB103(xg,yg) = .3.*GB10(xg,yg);GB104(xg,yg) = .4.*GB10(xg,yg);
        GB105(xg,yg) = .5.*GB10(xg,yg);GB106(xg,yg) = .6.*GB10(xg,yg);
        GB107(xg,yg) = .7.*GB10(xg,yg);GB108(xg,yg) = .8.*GB10(xg,yg);
        GB109(xg,yg) = .9.*GB10(xg,yg);
       end
    end
    G110 = Gaussian.*G10;  G1Tc = G1Tc + G110;
    GTc1 = GTc1 + Gaussian.*(G10-GB101);
    GTc2 = GTc2 + Gaussian.*(G10-GB102);
    GTc3 = GTc3 + Gaussian.*(G10-GB103);
    GTc4 = GTc4 + Gaussian.*(G10-GB104);
    GTc5 = GTc5 + Gaussian.*(G10-GB105);
    GTc6 = GTc6 + Gaussian.*(G10-GB106);
    GTc7 = GTc7 + Gaussian.*(G10-GB107);
    GTc8 = GTc8 + Gaussian.*(G10-GB108);
    GTc9 = GTc9 + Gaussian.*(G10-GB109);

    logv10 = ifftshift(ifft2(ifftshift(G110))); v10 = exp(logv10); vTi = vTi + v10;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #10');
    surf(abs(v10));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv10; clear logvsi10Ref; clear VSI10Causal;
case 11
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #11');scatter3(x11,y11,z11,5,z11,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #11 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
        if(percentdone == 5)
            fprintf('50%%');
        end
        if(percentdone == 7)
            fprintf('.75%%')
        end
    end
    for iy=1:ny
        vsi11(ix,iy) = sum(      In11(:,3) .* exp(i.*2.0.*pi.* ...
            (In11(:,1).*(dx.*(ix-nx/2-1)+sx)+ In11(:,2).*(dy.*(iy-ny/2-1)+sy))));
        vsib11(ix,iy) = sum(      Inb11(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb11(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb11(:,2).*(dy.*(iy-ny/2-1)+sy))));
    end
end
clear In11;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #11');
surf(abs(vsi11));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
    for rr = 1:r/2
        vsi11c(rr,ccc) = vsi11(rr*2,ccc*2);vsib11c(rr,ccc) = vsib11(rr*2,ccc*2);
    end;
end;
[r1,c1] = size(vsi11c);
VSI11 = fftshift(fft2(fftshift(vsi11c)));VSIB11 = fftshift(fft2(fftshift(vsib11c)));
AbsVSI11 = abs(VSI11);AbsVSIB11 = abs(VSIB11);
MaxAbsVSI11 = max(max(AbsVSI11));
MaxAbsVSIB11 = max(max(AbsVSIB11));
RefVal11 = (MaxAbsVSI11+RefBias)*RefMult;
RefValB11 = (MaxAbsVSIB11+RefBias)*RefMult;
VSI11Causal = zeros(NewSize,NewSize); %
VSIB11Causal = zeros(NewSize,NewSize);
VSI11Causal(1:r1,NewMidPoint:NewSize) = VSI11;
VSIB11Causal(1:r1,NewMidPoint:NewSize) = VSIB11;
VSI11Causal(RefRowPos,RefColPos) = RefVal11;
VSIB11Causal(RefRowPos,RefColPos) = RefValB11;
vsi11Ref = ifftshift(ifft2(ifftshift(VSI11Causal)));
vsi11Phase = atan2(imag(vsi11Ref),real(vsi11Ref));
vsib11Ref = ifftshift(ifft2(ifftshift(VSIB11Causal)));
logvsi11Ref = log(vsi11Ref);
logvsib11Ref = log(vsib11Ref);
G11 = fftshift(fft2(fftshift(logvsi11Ref)));
GB11 = fftshift(fft2(fftshift(logvsib11Ref)));
GB111 = zeros(nx,ny);GB112 = zeros(nx,ny);GB113 = zeros(nx,ny);
GB114 = zeros(nx,ny);GB115 = zeros(nx,ny);GB116 = zeros(nx,ny);
```

```
    GB117 = zeros(nx,ny);GB118 = zeros(nx,ny);GB119 = zeros(nx,ny);

    for xg=1:1:nx
       for yg=1:1:nx
       GB111(xg,yg) = 1.*GB11(xg,yg);GB112(xg,yg) = .2.*GB11(xg,yg);
       GB113(xg,yg) = .3.*GB11(xg,yg);GB114(xg,yg) = .4.*GB11(xg,yg);
       GB115(xg,yg) = .5.*GB11(xg,yg);GB116(xg,yg) = .6.*GB11(xg,yg);
       GB117(xg,yg) = .7.*GB11(xg,yg);GB118(xg,yg) = .8.*GB11(xg,yg);
       GB119(xg,yg) = .9.*GB11(xg,yg);
      end
    end
    G111 = Gaussian.*G11;  G1Tc = G1Tc + G111;
    GTc1 = GTc1 + Gaussian.*(G11-GB111);
    GTc2 = GTc2 + Gaussian.*(G11-GB112);
    GTc3 = GTc3 + Gaussian.*(G11-GB113);
    GTc4 = GTc4 + Gaussian.*(G11-GB114);
    GTc5 = GTc5 + Gaussian.*(G11-GB115);
    GTc6 = GTc6 + Gaussian.*(G11-GB116);
    GTc7 = GTc7 + Gaussian.*(G11-GB117);
    GTc8 = GTc8 + Gaussian.*(G11-GB118);
    GTc9 = GTc9 + Gaussian.*(G11-GB119);
    logv11 = ifftshift(ifft2(ifftshift(G111))); v11 = exp(logv11); vTi = vTi + v11;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #11');
    surf(abs(v11));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv11; clear logvsi11Ref; clear VSI11Causal;
case 12
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #12');scatter3(x12,y12,z12,5,z12,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #12 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
       count = count+1;
       if(count >= statuspt)
          fprintf('.');count = 0;percentdone = percentdone+1;
          if(percentdone == 2)
             fprintf('25%%')
          end
          if(percentdone == 5)
             fprintf('50%%');
          end
          if(percentdone == 7)
             fprintf('.75%%')
          end
```

```
  end
  for iy=1:ny
    vsi12(ix,iy) = sum(    In12(:,3) .* exp(i.*2.0.*pi.* ...
      (In12(:,1).*(dx.*(ix-nx/2-1)+sx)+ In12(:,2).*(dy.*(iy-ny/2-1)+sy))));
    vsib12(ix,iy) = sum(    Inb12(:,3) .* exp(i.*2.0.*pi.* ...
      (Inb12(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb12(:,2).*(dy.*(iy-ny/2-1)+sy))));
  end
end
clear In12;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #12');
surf(abs(vsi12));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
  for rr = 1:r/2
    vsi12c(rr,ccc) = vsi12(rr*2,ccc*2);vsib12c(rr,ccc) = vsib12(rr*2,ccc*2);
  end;
end;
[r1,c1] = size(vsi12c);
VSI12 = fftshift(fft2(fftshift(vsi12c)));AbsVSI12 = abs(VSI12);
MaxAbsVSI12 = max(max(AbsVSI12));
RefVal12 = (MaxAbsVSI12+RefBias)*RefMult;
VSI12Causal = zeros(NewSize,NewSize);
VSI12Causal(1:r1,NewMidPoint:NewSize) = VSI12;
VSI12Causal(RefRowPos,RefColPos) = RefVal12;

VSIB12 = fftshift(fft2(fftshift(vsib12c)));AbsVSIB12 = abs(VSIB12);
MaxAbsVSIB12 = max(max(AbsVSIB12));
RefValB12 = (MaxAbsVSIB12+RefBias)*RefMult;
VSIB12Causal = zeros(NewSize,NewSize);
VSIB12Causal(1:r1,NewMidPoint:NewSize) = VSIB12;
VSIB12Causal(RefRowPos,RefColPos) = RefValB12;
vsi12Ref = ifftshift(ifft2(ifftshift(VSI12Causal)));
vsi12Phase = atan2(imag(vsi12Ref),real(vsi12Ref)); % Phase of vsi
vsib12Ref = ifftshift(ifft2(ifftshift(VSIB12Causal)));
logvsi12Ref = log(vsi12Ref);  logvsib12Ref = log(vsib12Ref);
G12 = fftshift(fft2(fftshift(logvsi12Ref)));
GB12 = fftshift(fft2(fftshift(logvsib12Ref)));
GB121 = zeros(nx,ny);GB122 = zeros(nx,ny);GB123 = zeros(nx,ny);
GB124 = zeros(nx,ny);GB125 = zeros(nx,ny);GB126 = zeros(nx,ny);
GB127 = zeros(nx,ny);GB128 = zeros(nx,ny);GB129 = zeros(nx,ny);
for xg=1:1:nx
  for yg=1:1:nx
  GB121(xg,yg) = 1.*GB12(xg,yg);GB122(xg,yg) = .2.*GB12(xg,yg);
  GB123(xg,yg) = .3.*GB12(xg,yg);GB124(xg,yg) = .4.*GB12(xg,yg);
  GB125(xg,yg) = .5.*GB12(xg,yg);GB126(xg,yg) = .6.*GB12(xg,yg);
```

```
      GB127(xg,yg) = .7.*GB12(xg,yg);GB128(xg,yg) = .8.*GB12(xg,yg);
      GB129(xg,yg) = .9.*GB12(xg,yg);
     end
   end
   G112 = Gaussian.*G12;  G1Tc = G1Tc + G112;
   GTc1 = GTc1 + Gaussian.*(G12-GB121);
   GTc2 = GTc2 + Gaussian.*(G12-GB122);
   GTc3 = GTc3 + Gaussian.*(G12-GB123);
   GTc4 = GTc4 + Gaussian.*(G12-GB124);
   GTc5 = GTc5 + Gaussian.*(G12-GB125);
   GTc6 = GTc6 + Gaussian.*(G12-GB126);
   GTc7 = GTc7 + Gaussian.*(G12-GB127);
   GTc8 = GTc8 + Gaussian.*(G12-GB128);
   GTc9 = GTc9 + Gaussian.*(G12-GB129);
   logv12 = ifftshift(ifft2(ifftshift(G112))); v12 = exp(logv12); vTi = vTi + v12;
   if(showCep ==1)
   figure('Name', 'Reconstructed V #12');
   surf(abs(v12));shading flat;colormap(jet); view(0,-90);colorbar
   end
   clear logv12; clear logvsi12Ref; clear VSI12Causal;
case 13
   if(showEwald == 1)
   figure('Name', 'Ewald Circle for Source #13');scatter3(x13,y13,z13,5,z13,'filled');
   view(0,90);colormap(jet);colorbar;
   end
   fprintf('\nProcessing Files for Source #13 .');statuspt = nx/10;percentdone = 0;
   for ix=1:nx
      count = count+1;
      if(count >= statuspt)
         fprintf('.');count = 0;percentdone = percentdone+1;
         if(percentdone == 2)
            fprintf('25%%')
         end
         if(percentdone == 5)
            fprintf('50%%');
         end
         if(percentdone == 7)
            fprintf('.75%%')
         end
      end
      for iy=1:ny
         vsi13(ix,iy) = sum(      In13(:,3) .* exp(i.*2.0.*pi.* ...
            (In13(:,1).*(dx.*(ix-nx/2-1)+sx)+ In13(:,2).*(dy.*(iy-ny/2-1)+sy))));
         vsib13(ix,iy) = sum(      Inb13(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb13(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb13(:,2).*(dy.*(iy-ny/2-1)+sy))));
      end
```

```
end
clear In13;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #13');
surf(abs(vsi13));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
   for rr = 1:r/2
      vsi13c(rr,ccc) = vsi13(rr*2,ccc*2);vsib13c(rr,ccc) = vsib13(rr*2,ccc*2);
   end;
end;
[r1,c1] = size(vsi13c);
VSI13 = fftshift(fft2(fftshift(vsi13c)));AbsVSI13 = abs(VSI13);
MaxAbsVSI13 = max(max(AbsVSI13));
RefVal13 = (MaxAbsVSI13+RefBias)*RefMult;
VSI13Causal = zeros(NewSize,NewSize); %
VSI13Causal(1:r1,NewMidPoint:NewSize) = VSI13;
VSI13Causal(RefRowPos,RefColPos) = RefVal13;
VSIB13 = fftshift(fft2(fftshift(vsib13c)));AbsVSIB13 = abs(VSIB13);
MaxAbsVSIB13 = max(max(AbsVSIB13));
RefValB13 = (MaxAbsVSIB13+RefBias)*RefMult;
VSIB13Causal = zeros(NewSize,NewSize);
VSIB13Causal(1:r1,NewMidPoint:NewSize) = VSIB13;
VSIB13Causal(RefRowPos,RefColPos) = RefValB13;

vsi13Ref = ifftshift(ifft2(ifftshift(VSI13Causal)));
vsi13Phase = atan2(imag(vsi13Ref),real(vsi13Ref));
vsib13Ref = ifftshift(ifft2(ifftshift(VSIB13Causal)));
logvsi13Ref = log(vsi13Ref);  logvsib13Ref = log(vsib13Ref);
G13 = fftshift(fft2(fftshift(logvsi13Ref)));
GB13 = fftshift(fft2(fftshift(logvsib13Ref)));
GB131 = zeros(nx,ny);GB132 = zeros(nx,ny);GB133 = zeros(nx,ny);
GB134 = zeros(nx,ny);GB135 = zeros(nx,ny);GB136 = zeros(nx,ny);
GB137 = zeros(nx,ny);GB138 = zeros(nx,ny);GB139 = zeros(nx,ny);
for xg=1:1:nx
   for yg=1:1:nx
   GB131(xg,yg) =  1.*GB13(xg,yg);GB132(xg,yg) = .2.*GB13(xg,yg);
   GB133(xg,yg) = .3.*GB13(xg,yg);GB134(xg,yg) = .4.*GB13(xg,yg);
   GB135(xg,yg) = .5.*GB13(xg,yg);GB136(xg,yg) = .6.*GB13(xg,yg);
   GB137(xg,yg) = .7.*GB13(xg,yg);GB138(xg,yg) = .8.*GB13(xg,yg);
   GB139(xg,yg) = .9.*GB13(xg,yg);
  end
end
G113 = Gaussian.*G13;  G1Tc = G1Tc + G113;
GTc1 = GTc1 + Gaussian.*(G13-GB131);
GTc2 = GTc2 + Gaussian.*(G13-GB132);
```

```
    GTc3 = GTc3 + Gaussian.*(G13-GB133);
    GTc4 = GTc4 + Gaussian.*(G13-GB134);
    GTc5 = GTc5 + Gaussian.*(G13-GB135);
    GTc6 = GTc6 + Gaussian.*(G13-GB136);
    GTc7 = GTc7 + Gaussian.*(G13-GB137);
    GTc8 = GTc8 + Gaussian.*(G13-GB138);
    GTc9 = GTc9 + Gaussian.*(G13-GB139);
     logv13 = ifftshift(ifft2(ifftshift(G113))); v13 = exp(logv13); vTi = vTi + v13;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #13');
    surf(abs(v13));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv13; clear logvsi13Ref; clear VSI13Causal;
case 14
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #14');scatter3(x14,y14,z14,5,z14,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #14 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi14(ix,iy) = sum(      In14(:,3) .* exp(i.*2.0.*pi.* ...
                (In14(:,1).*(dx.*(ix-nx/2-1)+sx)+ In14(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib14(ix,iy) = sum(      Inb14(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb14(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb14(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In14;fprintf('.. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #14');
    surf(abs(vsi14));shading flat;colormap(jet); view(0,-90);colorbar
    end
    for ccc = 1:c/2
        for rr = 1:r/2
            vsi14c(rr,ccc) = vsi14(rr*2,ccc*2);vsib14c(rr,ccc) = vsib14(rr*2,ccc*2);
        end;
    end;
    [r1,c1] = size(vsi14c);
    VSI14 = fftshift(fft2(fftshift(vsi14c)));AbsVSI14 = abs(VSI14);
    MaxAbsVSI14 = max(max(AbsVSI14));
    RefVal14 = (MaxAbsVSI14+RefBias)*RefMult;
    VSI14Causal = zeros(NewSize,NewSize); %
    VSI14Causal(1:r1,NewMidPoint:NewSize) = VSI14;
    VSI14Causal(RefRowPos,RefColPos) = RefVal14;
    VSIB14 = fftshift(fft2(fftshift(vsib14c)));AbsVSIB14 = abs(VSIB14);
    MaxAbsVSIB14 = max(max(AbsVSIB14));
    RefValB14 = (MaxAbsVSIB14+RefBias)*RefMult;
    VSIB14Causal = zeros(NewSize,NewSize);
    VSIB14Causal(1:r1,NewMidPoint:NewSize) = VSIB14;
    VSIB14Causal(RefRowPos,RefColPos) = RefValB14;
    vsi14Ref = ifftshift(ifft2(ifftshift(VSI14Causal)));
    vsi14Phase = atan2(imag(vsi14Ref),real(vsi14Ref));
    vsib14Ref = ifftshift(ifft2(ifftshift(VSIB14Causal)));
    logvsi14Ref = log(vsi14Ref);  logvsib14Ref = log(vsib14Ref);
    G14 = fftshift(fft2(fftshift(logvsi14Ref)));
    GB14 = fftshift(fft2(fftshift(logvsib14Ref)));
    GB141 = zeros(nx,ny);GB142 = zeros(nx,ny);GB143 = zeros(nx,ny);
    GB144 = zeros(nx,ny);GB145 = zeros(nx,ny);GB146 = zeros(nx,ny);
    GB147 = zeros(nx,ny);GB148 = zeros(nx,ny);GB149 = zeros(nx,ny);
    for xg=1:1:nx
        for yg=1:1:nx
        GB141(xg,yg) =  1.*GB14(xg,yg);GB142(xg,yg) = .2.*GB14(xg,yg);
        GB143(xg,yg) = .3.*GB14(xg,yg);GB144(xg,yg) = .4.*GB14(xg,yg);
        GB145(xg,yg) = .5.*GB14(xg,yg);GB146(xg,yg) = .6.*GB14(xg,yg);
        GB147(xg,yg) = .7.*GB14(xg,yg);GB148(xg,yg) = .8.*GB14(xg,yg);
        GB149(xg,yg) = .9.*GB14(xg,yg);
       end
    end
    G114 = Gaussian.*G14;  G1Tc = G1Tc + G114;
    GTc1 = GTc1 + Gaussian.*(G14-GB141);
    GTc2 = GTc2 + Gaussian.*(G14-GB142);
    GTc3 = GTc3 + Gaussian.*(G14-GB143);
    GTc4 = GTc4 + Gaussian.*(G14-GB144);
    GTc5 = GTc5 + Gaussian.*(G14-GB145);
    GTc6 = GTc6 + Gaussian.*(G14-GB146);
    GTc7 = GTc7 + Gaussian.*(G14-GB147);
    GTc8 = GTc8 + Gaussian.*(G14-GB148);
    GTc9 = GTc9 + Gaussian.*(G14-GB149);
     logv14 = ifftshift(ifft2(ifftshift(G114))); v14 = exp(logv14); vTi = vTi + v14;
```

```
     if(showCep ==1)
    figure('Name', 'Reconstructed V #14');
    surf(abs(v14));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv14; clear logvsi14Ref; clear VSI14Causal;
case 15
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #15');scatter3(x15,y15,z15,5,z15,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #15 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end

        for iy=1:ny
            vsi15(ix,iy) = sum(      In15(:,3) .* exp(i.*2.0.*pi.* ...
                (In15(:,1).*(dx.*(ix-nx/2-1)+sx)+ In15(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib15(ix,iy) = sum(      Inb15(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb15(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb15(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In15;fprintf('. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #15');
    surf(abs(vsi15));shading flat;colormap(jet); view(0,-90);colorbar
    end
    for ccc = 1:c/2
        for rr = 1:r/2
            vsi15c(rr,ccc) = vsi15(rr*2,ccc*2);vsib15c(rr,ccc) = vsib15(rr*2,ccc*2);
        end;
    end;
    [r1,c1] = size(vsi15c);
    VSI15 = fftshift(fft2(fftshift(vsi15c)));AbsVSI15 = abs(VSI15);
    MaxAbsVSI15 = max(max(AbsVSI15));
    RefVal15 = (MaxAbsVSI15+RefBias)*RefMult;
    VSI15Causal = zeros(NewSize,NewSize); %
    VSI15Causal(1:r1,NewMidPoint:NewSize) = VSI15;
    VSI15Causal(RefRowPos,RefColPos) = RefVal15;
    VSIB15 = fftshift(fft2(fftshift(vsib15c)));
    AbsVSIB15 = abs(VSIB15);
    MaxAbsVSIB15 = max(max(AbsVSIB15));
    RefValB15 = (MaxAbsVSIB15+RefBias)*RefMult;
    VSIB15Causal = zeros(NewSize,NewSize);
    VSIB15Causal(1:r1,NewMidPoint:NewSize) = VSIB15;
    VSIB15Causal(RefRowPos,RefColPos) = RefValB15;
    vsi15Ref = ifftshift(ifft2(ifftshift(VSI15Causal)));
    vsi15Phase = atan2(imag(vsi15Ref),real(vsi15Ref));
    vsib15Ref = ifftshift(ifft2(ifftshift(VSIB15Causal)));
    logvsi15Ref = log(vsi15Ref);  logvsib15Ref = log(vsib15Ref);
    G15 = fftshift(fft2(fftshift(logvsi15Ref)));
    GB15 = fftshift(fft2(fftshift(logvsib15Ref)));
    GB151 = zeros(nx,ny);GB152 = zeros(nx,ny);GB153 = zeros(nx,ny);
    GB154 = zeros(nx,ny);GB155 = zeros(nx,ny);GB156 = zeros(nx,ny);
    GB157 = zeros(nx,ny);GB158 = zeros(nx,ny);GB159 = zeros(nx,ny);
    for xg=1:1:nx
       for yg=1:1:nx
       GB151(xg,yg) =  1.*GB15(xg,yg);GB152(xg,yg) = .2.*GB15(xg,yg);
       GB153(xg,yg) = .3.*GB15(xg,yg);GB154(xg,yg) = .4.*GB15(xg,yg);
       GB155(xg,yg) = .5.*GB15(xg,yg);GB156(xg,yg) = .6.*GB15(xg,yg);
       GB157(xg,yg) = .7.*GB15(xg,yg);GB158(xg,yg) = .8.*GB15(xg,yg);
       GB159(xg,yg) = .9.*GB15(xg,yg);
      end
    end
    G115 = Gaussian.*G15;  G1Tc = G1Tc + G115;
    GTc1 = GTc1 + Gaussian.*(G15-GB151);
    GTc2 = GTc2 + Gaussian.*(G15-GB152);
    GTc3 = GTc3 + Gaussian.*(G15-GB153);
    GTc4 = GTc4 + Gaussian.*(G15-GB154);
    GTc5 = GTc5 + Gaussian.*(G15-GB155);
    GTc6 = GTc6 + Gaussian.*(G15-GB156);
    GTc7 = GTc7 + Gaussian.*(G15-GB157);
    GTc8 = GTc8 + Gaussian.*(G15-GB158);
    GTc9 = GTc9 + Gaussian.*(G15-GB159);
    logv15 = ifftshift(ifft2(ifftshift(G115))); v15 = exp(logv15); vTi = vTi + v15;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #15');
    surf(abs(v15));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv15; clear logvsi15Ref; clear VSI15Causal;
case 16
```

```
if(showEwald == 1)
figure('Name', 'Ewald Circle for Source #16');scatter3(x16,y16,z16,5,z16,'filled');
view(0,90);colormap(jet);colorbar;
end
fprintf('\nProcessing Files for Source #16 .');statuspt = nx/10;percentdone = 0;
for ix=1:nx
   count = count+1;
   if(count >= statuspt)
      fprintf('.');count = 0;percentdone = percentdone+1;
      if(percentdone == 2)
         fprintf('25%%')
      end
      if(percentdone == 5)
         fprintf('50%%');
      end
      if(percentdone == 7)
         fprintf('.75%%')
      end
   end
   for iy=1:ny
      vsi16(ix,iy) = sum(      In16(:,3) .* exp(i.*2.0.*pi.* ...
         (In16(:,1).*(dx.*(ix-nx/2-1)+sx)+ In16(:,2).*(dy.*(iy-ny/2-1)+sy))));
      vsib16(ix,iy) = sum(      Inb16(:,3) .* exp(i.*2.0.*pi.* ...
         (Inb16(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb16(:,2).*(dy.*(iy-ny/2-1)+sy))));
   end
end
clear In16;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #16');
surf(abs(vsi16));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
   for rr = 1:r/2
      vsi16c(rr,ccc) = vsi16(rr*2,ccc*2);vsib16c(rr,ccc) = vsib16(rr*2,ccc*2);
   end;
end;
[r1,c1] = size(vsi16c);
VSI16 = fftshift(fft2(fftshift(vsi16c)));AbsVSI16 = abs(VSI16);
MaxAbsVSI16 = max(max(AbsVSI16));
RefVal16 = (MaxAbsVSI16+RefBias)*RefMult;
VSI16Causal = zeros(NewSize,NewSize); %
VSI16Causal(1:r1,NewMidPoint:NewSize) = VSI16;
VSI16Causal(RefRowPos,RefColPos) = RefVal16;
VSIB16 = fftshift(fft2(fftshift(vsib16c)));AbsVSIB16 = abs(VSIB16);
MaxAbsVSIB16 = max(max(AbsVSIB16));
RefValB16 = (MaxAbsVSIB16+RefBias)*RefMult;
```

```
    VSIB16Causal = zeros(NewSize,NewSize);
    VSIB16Causal(1:r1,NewMidPoint:NewSize) = VSIB16;
    VSIB16Causal(RefRowPos,RefColPos) = RefValB16;
    vsi16Ref = ifftshift(ifft2(ifftshift(VSI16Causal)));
    vsi16Phase = atan2(imag(vsi16Ref),real(vsi16Ref)); % Phase of vsi
    vsib16Ref = ifftshift(ifft2(ifftshift(VSIB16Causal)));
    logvsi16Ref = log(vsi16Ref);  logvsib16Ref = log(vsib16Ref);
    G16 = fftshift(fft2(fftshift(logvsi16Ref)));
    GB16 = fftshift(fft2(fftshift(logvsib16Ref)));
    GB161 = zeros(nx,ny);GB162 = zeros(nx,ny);GB163 = zeros(nx,ny);
    GB164 = zeros(nx,ny);GB165 = zeros(nx,ny);GB166 = zeros(nx,ny);
    GB167 = zeros(nx,ny);GB168 = zeros(nx,ny);GB169 = zeros(nx,ny);
    for xg=1:1:nx
       for yg=1:1:nx
       GB161(xg,yg) =  1.*GB16(xg,yg);GB162(xg,yg) = .2.*GB16(xg,yg);
       GB163(xg,yg) = .3.*GB16(xg,yg);GB164(xg,yg) = .4.*GB16(xg,yg);
       GB165(xg,yg) = .5.*GB16(xg,yg);GB166(xg,yg) = .6.*GB16(xg,yg);
       GB167(xg,yg) = .7.*GB16(xg,yg);GB168(xg,yg) = .8.*GB16(xg,yg);
       GB169(xg,yg) = .9.*GB16(xg,yg);
      end
    end
    G116 = Gaussian.*G16;  G1Tc = G1Tc + G116;
    GTc1 = GTc1 + Gaussian.*(G16-GB161);
    GTc2 = GTc2 + Gaussian.*(G16-GB162);
    GTc3 = GTc3 + Gaussian.*(G16-GB163);
    GTc4 = GTc4 + Gaussian.*(G16-GB164);
    GTc5 = GTc5 + Gaussian.*(G16-GB165);
    GTc6 = GTc6 + Gaussian.*(G16-GB166);
    GTc7 = GTc7 + Gaussian.*(G16-GB167);
    GTc8 = GTc8 + Gaussian.*(G16-GB168);
    GTc9 = GTc9 + Gaussian.*(G16-GB169);
    logv16 = ifftshift(ifft2(ifftshift(G116))); v16 = exp(logv16); vTi = vTi + v16;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #16');
    surf(abs(v16));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv16; clear logvsi16Ref; clear VSI16Causal;
 case 17
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #17');scatter3(x17,y17,z17,5,z17,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #17 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
       count = count+1;
       if(count >= statuspt)
```

```
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi17(ix,iy) = sum(      In17(:,3) .* exp(i.*2.0.*pi.* ...
                (In17(:,1).*(dx.*(ix-nx/2-1)+sx)+ In17(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib17(ix,iy) = sum(      Inb17(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb17(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb17(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In17; fprintf('. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #17');
    surf(abs(vsi17));shading flat;colormap(jet); view(0,-90);colorbar
    end
    for ccc = 1:c/2
        for rr = 1:r/2
            vsi17c(rr,ccc) = vsi17(rr*2,ccc*2);vsib17c(rr,ccc) = vsib17(rr*2,ccc*2);
        end;
    end;
    [r1,c1] = size(vsi17c);
    VSI17 = fftshift(fft2(fftshift(vsi17c)));AbsVSI17 = abs(VSI17);
    MaxAbsVSI17 = max(max(AbsVSI17));
    RefVal17 = (MaxAbsVSI17+RefBias)*RefMult;
    VSI17Causal = zeros(NewSize,NewSize); %
    VSI17Causal(1:r1,NewMidPoint:NewSize) = VSI17;
    VSI17Causal(RefRowPos,RefColPos) = RefVal17;
    VSIB17 = fftshift(fft2(fftshift(vsib17c)));
    AbsVSIB17 = abs(VSIB17);
    MaxAbsVSIB17 = max(max(AbsVSIB17));
    RefValB17 = (MaxAbsVSIB17+RefBias)*RefMult;
    VSIB17Causal = zeros(NewSize,NewSize);
    VSIB17Causal(1:r1,NewMidPoint:NewSize) = VSIB17;
    VSIB17Causal(RefRowPos,RefColPos) = RefValB17;
    vsi17Ref = ifftshift(ifft2(ifftshift(VSI17Causal)));
    vsi17Phase = atan2(imag(vsi17Ref),real(vsi17Ref));
    vsib17Ref = ifftshift(ifft2(ifftshift(VSIB17Causal)));
    logvsi17Ref = log(vsi17Ref);
```

```
        logvsib17Ref = log(vsib17Ref);
        G17 = fftshift(fft2(fftshift(logvsi17Ref)));
        GB17 = fftshift(fft2(fftshift(logvsib17Ref)));
        GB171 = zeros(nx,ny);GB172 = zeros(nx,ny);GB173 = zeros(nx,ny);
        GB174 = zeros(nx,ny);GB175 = zeros(nx,ny);GB176 = zeros(nx,ny);
        GB177 = zeros(nx,ny);GB178 = zeros(nx,ny);GB179 = zeros(nx,ny);
        for xg=1:1:nx
            for yg=1:1:nx
            GB171(xg,yg) =  1.*GB17(xg,yg);GB172(xg,yg) = .2.*GB17(xg,yg);
            GB173(xg,yg) = .3.*GB17(xg,yg);GB174(xg,yg) = .4.*GB17(xg,yg);
            GB175(xg,yg) = .5.*GB17(xg,yg);GB176(xg,yg) = .6.*GB17(xg,yg);
            GB177(xg,yg) = .7.*GB17(xg,yg);GB178(xg,yg) = .8.*GB17(xg,yg);
            GB179(xg,yg) = .9.*GB17(xg,yg);
           end
        end
        G117 = Gaussian.*G17;  G1Tc = G1Tc + G117;
        GTc1 = GTc1 + Gaussian.*(G17-GB171);
        GTc2 = GTc2 + Gaussian.*(G17-GB172);
        GTc3 = GTc3 + Gaussian.*(G17-GB173);
        GTc4 = GTc4 + Gaussian.*(G17-GB174);
        GTc5 = GTc5 + Gaussian.*(G17-GB175);
        GTc6 = GTc6 + Gaussian.*(G17-GB176);
        GTc7 = GTc7 + Gaussian.*(G17-GB177);
        GTc8 = GTc8 + Gaussian.*(G17-GB178);
        GTc9 = GTc9 + Gaussian.*(G17-GB179);
        logv17 = ifftshift(ifft2(ifftshift(G117))); v17 = exp(logv17); vTi = vTi + v17;
        if(showCep ==1)
        figure('Name', 'Reconstructed V #17');
        surf(abs(v17));shading flat;colormap(jet); view(0,-90);colorbar
        end
        clear logv17; clear logvsi17Ref; clear VSI17Causal;
    case 18
        if(showEwald == 1)
        figure('Name', 'Ewald Circle for Source #18');scatter3(x18,y18,z18,5,z18,'filled');
        view(0,90);colormap(jet);colorbar;
        end
        fprintf('\nProcessing Files for Source #18 .');statuspt = nx/10;percentdone = 0;
        for ix=1:nx
            count = count+1;
            if(count >= statuspt)
                fprintf('.');count = 0;percentdone = percentdone+1;
                if(percentdone == 2)
                    fprintf('25%%')
                end
                if(percentdone == 5)
                    fprintf('50%%');
    end
    if(percentdone == 7)
      fprintf('.75%%')
    end
  end
  for iy=1:ny
    vsi18(ix,iy) = sum(      In18(:,3) .* exp(i.*2.0.*pi.* ...
      (In18(:,1).*(dx.*(ix-nx/2-1)+sx)+ In18(:,2).*(dy.*(iy-ny/2-1)+sy))));
    vsib18(ix,iy) = sum(      Inb18(:,3) .* exp(i.*2.0.*pi.* ...
      (Inb18(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb18(:,2).*(dy.*(iy-ny/2-1)+sy))));
  end
end
clear In18;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #18');
surf(abs(vsi18));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
  for rr = 1:r/2
    vsi18c(rr,ccc) = vsi18(rr*2,ccc*2);vsib18c(rr,ccc) = vsib18(rr*2,ccc*2);
  end;
end;
[r1,c1] = size(vsi18c);
VSI18 = fftshift(fft2(fftshift(vsi18c)));AbsVSI18 = abs(VSI18);
MaxAbsVSI18 = max(max(AbsVSI18));
RefVal18 = (MaxAbsVSI18+RefBias)*RefMult;
VSI18Causal = zeros(NewSize,NewSize); %
VSI18Causal(1:r1,NewMidPoint:NewSize) = VSI18;
VSI18Causal(RefRowPos,RefColPos) = RefVal18;
VSIB18 = fftshift(fft2(fftshift(vsib18c)));AbsVSIB18 = abs(VSIB18);
MaxAbsVSIB18 = max(max(AbsVSIB18));
RefValB18 = (MaxAbsVSIB18+RefBias)*RefMult;
VSIB18Causal = zeros(NewSize,NewSize);
VSIB18Causal(1:r1,NewMidPoint:NewSize) = VSIB18;
VSIB18Causal(RefRowPos,RefColPos) = RefValB18;
vsi18Ref = ifftshift(ifft2(ifftshift(VSI18Causal)));
vsi18Phase = atan2(imag(vsi18Ref),real(vsi18Ref)); % Phase of vsi
vsib18Ref = ifftshift(ifft2(ifftshift(VSIB18Causal)));
logvsi18Ref = log(vsi18Ref);  logvsib18Ref = log(vsib18Ref);
G18 = fftshift(fft2(fftshift(logvsi18Ref)));
GB18 = fftshift(fft2(fftshift(logvsib18Ref)));
GB181 = zeros(nx,ny);GB182 = zeros(nx,ny);GB183 = zeros(nx,ny);
GB184 = zeros(nx,ny);GB185 = zeros(nx,ny);GB186 = zeros(nx,ny);
GB187 = zeros(nx,ny);GB188 = zeros(nx,ny);GB189 = zeros(nx,ny);
for xg=1:1:nx
  for yg=1:1:nx
```

```
      GB181(xg,yg) =  1.*GB18(xg,yg);GB182(xg,yg) = .2.*GB18(xg,yg);
      GB183(xg,yg) = .3.*GB18(xg,yg);GB184(xg,yg) = .4.*GB18(xg,yg);
      GB185(xg,yg) = .5.*GB18(xg,yg);GB186(xg,yg) = .6.*GB18(xg,yg);
      GB187(xg,yg) = .7.*GB18(xg,yg);GB188(xg,yg) = .8.*GB18(xg,yg);
      GB189(xg,yg) = .9.*GB18(xg,yg);
     end
   end
    G118 = Gaussian.*G18;  G1Tc = G1Tc + G118;
   GTc1 = GTc1 + Gaussian.*(G18-GB181);
   GTc2 = GTc2 + Gaussian.*(G18-GB182);
   GTc3 = GTc3 + Gaussian.*(G18-GB183);
   GTc4 = GTc4 + Gaussian.*(G18-GB184);
   GTc5 = GTc5 + Gaussian.*(G18-GB185);
   GTc6 = GTc6 + Gaussian.*(G18-GB186);
   GTc7 = GTc7 + Gaussian.*(G18-GB187);
   GTc8 = GTc8 + Gaussian.*(G18-GB188);
   GTc9 = GTc9 + Gaussian.*(G18-GB189);
    logv18 = ifftshift(ifft2(ifftshift(G118))); v18 = exp(logv18); vTi = vTi + v18;
    if(showCep ==1)
   figure('Name', 'Reconstructed V #18');
   surf(abs(v18));shading flat;colormap(jet); view(0,-90);colorbar
   end
   clear logv18; clear logvsi18Ref; clear VSI18Causal;
case 19
   if(showEwald == 1)
   figure('Name', 'Ewald Circle for Source #19');scatter3(x19,y19,z19,5,z19,'filled');
   view(0,90);colormap(jet);colorbar;
   end
   fprintf('\nProcessing Files for Source #19 .');statuspt = nx/10;percentdone = 0;
   for ix=1:nx
      count = count+1;
      if(count >= statuspt)
         fprintf('.');count = 0;percentdone = percentdone+1;
         if(percentdone == 2)
            fprintf('25%%')
         end
         if(percentdone == 5)
            fprintf('50%%');
         end
         if(percentdone == 7)
            fprintf('.75%%')
         end
      end
      for iy=1:ny
         vsi19(ix,iy) = sum(      In19(:,3) .* exp(i.*2.0.*pi.* ...
            (In19(:,1).*(dx.*(ix-nx/2-1)+sx)+ In19(:,2).*(dy.*(iy-ny/2-1)+sy))));
```

```
        vsib19(ix,iy) = sum(      Inb19(:,3) .* exp(i.*2.0.*pi.* ...
           (Inb19(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb19(:,2).*(dy.*(iy-ny/2-1)+sy))));
    end
end
clear In19;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #19');
surf(abs(vsi19));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
    for rr = 1:r/2
        vsi19c(rr,ccc) = vsi19(rr*2,ccc*2);vsib19c(rr,ccc) = vsib19(rr*2,ccc*2);
    end;
end;
[r1,c1] = size(vsi19c);
VSI19 = fftshift(fft2(fftshift(vsi19c)));AbsVSI19 = abs(VSI19);
MaxAbsVSI19 = max(max(AbsVSI19));
RefVal19 = (MaxAbsVSI19+RefBias)*RefMult;
VSI19Causal = zeros(NewSize,NewSize); %
VSI19Causal(1:r1,NewMidPoint:NewSize) = VSI19;
VSI19Causal(RefRowPos,RefColPos) = RefVal19;
VSIB19 = fftshift(fft2(fftshift(vsib19c)));AbsVSIB19 = abs(VSIB19);
MaxAbsVSIB19 = max(max(AbsVSIB19));
RefValB19 = (MaxAbsVSIB19+RefBias)*RefMult;
VSIB19Causal = zeros(NewSize,NewSize);
VSIB19Causal(1:r1,NewMidPoint:NewSize) = VSIB19;
VSIB19Causal(RefRowPos,RefColPos) = RefValB19;
vsi19Ref = ifftshift(ifft2(ifftshift(VSI19Causal)));
vsi19Phase = atan2(imag(vsi19Ref),real(vsi19Ref));
vsib19Ref = ifftshift(ifft2(ifftshift(VSIB19Causal)));
logvsi19Ref = log(vsi19Ref); logvsib19Ref = log(vsib19Ref);
G19 = fftshift(fft2(fftshift(logvsi19Ref)));
GB19 = fftshift(fft2(fftshift(logvsib19Ref)));
GB191 = zeros(nx,ny);GB192 = zeros(nx,ny);GB193 = zeros(nx,ny);
GB194 = zeros(nx,ny);GB195 = zeros(nx,ny);GB196 = zeros(nx,ny);
GB197 = zeros(nx,ny);GB198 = zeros(nx,ny);GB199 = zeros(nx,ny);
for xg=1:1:nx
    for yg=1:1:nx
    GB191(xg,yg) =  1.*GB19(xg,yg);GB192(xg,yg) = .2.*GB19(xg,yg);
    GB193(xg,yg) = .3.*GB19(xg,yg);GB194(xg,yg) = .4.*GB19(xg,yg);
    GB195(xg,yg) = .5.*GB19(xg,yg);GB196(xg,yg) = .6.*GB19(xg,yg);
    GB197(xg,yg) = .7.*GB19(xg,yg);GB198(xg,yg) = .8.*GB19(xg,yg);
    GB199(xg,yg) = .9.*GB19(xg,yg);
   end
end
G119 = Gaussian.*G19;  G1Tc = G1Tc + G119;
```

```
    GTc1 = GTc1 + Gaussian.*(G19-GB191);
    GTc2 = GTc2 + Gaussian.*(G19-GB192);
    GTc3 = GTc3 + Gaussian.*(G19-GB193);
    GTc4 = GTc4 + Gaussian.*(G19-GB194);
    GTc5 = GTc5 + Gaussian.*(G19-GB195);
    GTc6 = GTc6 + Gaussian.*(G19-GB196);
    GTc7 = GTc7 + Gaussian.*(G19-GB197);
    GTc8 = GTc8 + Gaussian.*(G19-GB198);
    GTc9 = GTc9 + Gaussian.*(G19-GB199);
     logv19 = ifftshift(ifft2(ifftshift(G119))); v19 = exp(logv19); vTi = vTi + v19;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #19');
    surf(abs(v19));shading flat;colormap(jet); view(0,-90);colorbar
    end
    clear logv19; clear logvsi19Ref; clear VSI19Causal;
case 20
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #20');scatter3(x20,y20,z20,5,z20,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #20 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi20(ix,iy) = sum(      In20(:,3) .* exp(i.*2.0.*pi.* ...
                (In20(:,1).*(dx.*(ix-nx/2-1)+sx)+ In20(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib20(ix,iy) = sum(      Inb20(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb20(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb20(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In20;fprintf('.. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #20');
    surf(abs(vsi20));shading flat;colormap(jet); view(0,-90);colorbar
```

```
end
for ccc = 1:c/2
   for rr = 1:r/2
      vsi20c(rr,ccc) = vsi20(rr*2,ccc*2);vsib20c(rr,ccc) = vsib20(rr*2,ccc*2);
   end;
end;
[r1,c1] = size(vsi20c);
VSI20 = fftshift(fft2(fftshift(vsi20c)));AbsVSI20 = abs(VSI20);
MaxAbsVSI20 = max(max(AbsVSI20));
RefVal20 = (MaxAbsVSI20+RefBias)*RefMult;
VSI20Causal = zeros(NewSize,NewSize); %
VSI20Causal(1:r1,NewMidPoint:NewSize) = VSI20;
VSI20Causal(RefRowPos,RefColPos) = RefVal20;
VSIB20 = fftshift(fft2(fftshift(vsib20c)));AbsVSIB20 = abs(VSIB20);
MaxAbsVSIB20 = max(max(AbsVSIB20));
RefValB20 = (MaxAbsVSIB20+RefBias)*RefMult;
VSIB20Causal = zeros(NewSize,NewSize);
VSIB20Causal(1:r1,NewMidPoint:NewSize) = VSIB20;
VSIB20Causal(RefRowPos,RefColPos) = RefValB20;
vsi20Ref = ifftshift(ifft2(ifftshift(VSI20Causal)));
vsi20Phase = atan2(imag(vsi20Ref),real(vsi20Ref));
vsib20Ref = ifftshift(ifft2(ifftshift(VSIB20Causal)));
logvsi20Ref = log(vsi20Ref);  logvsib20Ref = log(vsib20Ref);
G20 = fftshift(fft2(fftshift(logvsi20Ref)));
GB20 = fftshift(fft2(fftshift(logvsib20Ref)));
GB201 = zeros(nx,ny);GB202 = zeros(nx,ny);GB203 = zeros(nx,ny);
GB204 = zeros(nx,ny);GB205 = zeros(nx,ny);GB206 = zeros(nx,ny);
GB207 = zeros(nx,ny);GB208 = zeros(nx,ny);GB209 = zeros(nx,ny);
for xg=1:1:nx
   for yg=1:1:nx
   GB201(xg,yg) =  1.*GB20(xg,yg);GB202(xg,yg) = .2.*GB20(xg,yg);
   GB203(xg,yg) = .3.*GB20(xg,yg);GB204(xg,yg) = .4.*GB20(xg,yg);
   GB205(xg,yg) = .5.*GB20(xg,yg);GB206(xg,yg) = .6.*GB20(xg,yg);
   GB207(xg,yg) = .7.*GB20(xg,yg);GB208(xg,yg) = .8.*GB20(xg,yg);
   GB209(xg,yg) = .9.*GB20(xg,yg);
   end
end
G120 = Gaussian.*G20;  G1Tc = G1Tc + G120;
GTc1 = GTc1 + Gaussian.*(G20-GB201);
GTc2 = GTc2 + Gaussian.*(G20-GB202);
GTc3 = GTc3 + Gaussian.*(G20-GB203);
GTc4 = GTc4 + Gaussian.*(G20-GB204);
GTc5 = GTc5 + Gaussian.*(G20-GB205);
GTc6 = GTc6 + Gaussian.*(G20-GB206);
GTc7 = GTc7 + Gaussian.*(G20-GB207);
GTc8 = GTc8 + Gaussian.*(G20-GB208);
```

```
   GTc9 = GTc9 + Gaussian.*(G20-GB209);
   logv20 = ifftshift(ifft2(ifftshift(G120))); v20 = exp(logv20); vTi = vTi + v20;
   if(showCep ==1)
   figure('Name', 'Reconstructed V #20');
   surf(abs(v20));shading flat;colormap(jet); view(0,-90);colorbar
   end
case 21
   if(showEwald == 1)
   figure('Name', 'Ewald Circle for Source #21');scatter3(x21,y21,z21,5,z21,'filled');
   view(0,90);colormap(jet);colorbar;
   end
   fprintf('\nProcessing Files for Source #21 .');statuspt = nx/10;percentdone = 0;
   for ix=1:nx
      count = count+1;
      if(count >= statuspt)
         fprintf('.');count = 0;percentdone = percentdone+1;
         if(percentdone == 2)
            fprintf('25%%')
         end
         if(percentdone == 5)
            fprintf('50%%');
         end
         if(percentdone == 7)
            fprintf('.75%%')
         end
      end
      for iy=1:ny
         vsi21(ix,iy) = sum(      In21(:,3) .* exp(i.*2.0.*pi.* ...
            (In21(:,1).*(dx.*(ix-nx/2-1)+sx)+ In21(:,2).*(dy.*(iy-ny/2-1)+sy))));
         vsib21(ix,iy) = sum(      Inb21(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb21(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb21(:,2).*(dy.*(iy-ny/2-1)+sy))));
      end
   end
   clear In21; fprintf('. Done\n');
   if (showBorn == 1)
   figure('Name', 'Born VSi for Source #21');
   surf(abs(vsi21));shading flat;colormap(jet); view(0,-90);colorbar
   end
   for ccc = 1:c/2
      for rr = 1:r/2
         vsi21c(rr,ccc) = vsi21(rr*2,ccc*2);vsib21c(rr,ccc) = vsib21(rr*2,ccc*2);
      end;
   end;
   [r1,c1] = size(vsi21c);
   VSI21 = fftshift(fft2(fftshift(vsi21c)));AbsVSI21 = abs(VSI21);
   MaxAbsVSI21 = max(max(AbsVSI21));
```

```
    RefVal21 = (MaxAbsVSI21+RefBias)*RefMult;
    VSI21Causal = zeros(NewSize,NewSize); %
    VSI21Causal(1:r1,NewMidPoint:NewSize) = VSI21;
    VSI21Causal(RefRowPos,RefColPos) = RefVal21;
    VSIB21 = fftshift(fft2(fftshift(vsib21c)));AbsVSIB21 = abs(VSIB21);
    MaxAbsVSIB21 = max(max(AbsVSIB21));
    RefValB21 = (MaxAbsVSIB21+RefBias)*RefMult;
    VSIB21Causal = zeros(NewSize,NewSize);
    VSIB21Causal(1:r1,NewMidPoint:NewSize) = VSIB21;
    VSIB21Causal(RefRowPos,RefColPos) = RefValB21;
    vsi21Ref = ifftshift(ifft2(ifftshift(VSI21Causal)));
    vsi21Phase = atan2(imag(vsi21Ref),real(vsi21Ref));
    vsib21Ref = ifftshift(ifft2(ifftshift(VSIB21Causal)));
    logvsi21Ref = log(vsi21Ref); logvsib21Ref = log(vsib21Ref);
    G21 = fftshift(fft2(fftshift(logvsi21Ref)));
    GB21 = fftshift(fft2(fftshift(logvsib21Ref)));
    GB211 = zeros(nx,ny);GB212 = zeros(nx,ny);GB213 = zeros(nx,ny);
    GB214 = zeros(nx,ny);GB215 = zeros(nx,ny);GB216 = zeros(nx,ny);
    GB217 = zeros(nx,ny);GB218 = zeros(nx,ny);GB219 = zeros(nx,ny);
    for xg=1:1:nx
        for yg=1:1:nx
        GB211(xg,yg) =  1.*GB21(xg,yg);GB212(xg,yg) = .2.*GB21(xg,yg);
        GB213(xg,yg) = .3.*GB21(xg,yg);GB214(xg,yg) = .4.*GB21(xg,yg);
        GB215(xg,yg) = .5.*GB21(xg,yg);GB216(xg,yg) = .6.*GB21(xg,yg);
        GB217(xg,yg) = .7.*GB21(xg,yg);GB218(xg,yg) = .8.*GB21(xg,yg);
        GB219(xg,yg) = .9.*GB21(xg,yg);
        end
    end
    G121 = Gaussian.*G21;  G1Tc = G1Tc + G121;
    GTc1 = GTc1 + Gaussian.*(G21-GB211);
    GTc2 = GTc2 + Gaussian.*(G21-GB212);
    GTc3 = GTc3 + Gaussian.*(G21-GB213);
    GTc4 = GTc4 + Gaussian.*(G21-GB214);
    GTc5 = GTc5 + Gaussian.*(G21-GB215);
    GTc6 = GTc6 + Gaussian.*(G21-GB216);
    GTc7 = GTc7 + Gaussian.*(G21-GB217);
    GTc8 = GTc8 + Gaussian.*(G21-GB218);
    GTc9 = GTc9 + Gaussian.*(G21-GB219);
    logv21 = ifftshift(ifft2(ifftshift(G121))); v21 = exp(logv21); vTi = vTi + v21;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #21');
    surf(abs(v21));shading flat;colormap(jet); view(0,-90);colorbar
    end
case 22
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #22');scatter3(x22,y22,z22,5,z22,'filled');
```

```
view(0,90);colormap(jet);colorbar;
end
fprintf('\nProcessing Files for Source #22 .');statuspt = nx/10;percentdone = 0;
for ix=1:nx
   count = count+1;
   if(count >= statuspt)
      fprintf('.');count = 0;percentdone = percentdone+1;
      if(percentdone == 2)
         fprintf('25%%')
      end
      if(percentdone == 5)
         fprintf('50%%');
      end
      if(percentdone == 7)
         fprintf('.75%%')
      end
   end
   for iy=1:ny
      vsi22(ix,iy) = sum(      In22(:,3) .* exp(i.*2.0.*pi.* ...
         (In22(:,1).*(dx.*(ix-nx/2-1)+sx)+ In22(:,2).*(dy.*(iy-ny/2-1)+sy))));
      vsib22(ix,iy) = sum(      Inb22(:,3) .* exp(i.*2.0.*pi.* ...
         (Inb22(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb22(:,2).*(dy.*(iy-ny/2-1)+sy))));
   end
end
clear In22;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #22');
surf(abs(vsi22));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
   for rr = 1:r/2
      vsi22c(rr,ccc) = vsi22(rr*2,ccc*2);vsib22c(rr,ccc) = vsib22(rr*2,ccc*2);
   end;
end;
[r1,c1] = size(vsi22c);
VSI22 = fftshift(fft2(fftshift(vsi22c)));           AbsVSI22 = abs(VSI22);
MaxAbsVSI22 = max(max(AbsVSI22));
RefVal22 = (MaxAbsVSI22+RefBias)*RefMult;
VSI22Causal = zeros(NewSize,NewSize); %
VSI22Causal(1:r1,NewMidPoint:NewSize) = VSI22;
VSI22Causal(RefRowPos,RefColPos) = RefVal22;
VSIB22 = fftshift(fft2(fftshift(vsib22c)));AbsVSIB22 = abs(VSIB22);
MaxAbsVSIB22 = max(max(AbsVSIB22));
RefValB22 = (MaxAbsVSIB22+RefBias)*RefMult;
VSIB22Causal = zeros(NewSize,NewSize);
VSIB22Causal(1:r1,NewMidPoint:NewSize) = VSIB22;
```

```
         VSIB22Causal(RefRowPos,RefColPos) = RefValB22;
         vsi22Ref = ifftshift(ifft2(ifftshift(VSI22Causal)));
         vsi22Phase = atan2(imag(vsi22Ref),real(vsi22Ref));
         vsib22Ref = ifftshift(ifft2(ifftshift(VSIB22Causal)));
         logvsi22Ref = log(vsi22Ref); logvsib22Ref = log(vsib22Ref);
         G22 = fftshift(fft2(fftshift(logvsi22Ref)));
         GB22 = fftshift(fft2(fftshift(logvsib22Ref)));
         GB221 = zeros(nx,ny);GB222 = zeros(nx,ny);GB223 = zeros(nx,ny);
         GB224 = zeros(nx,ny);GB225 = zeros(nx,ny);GB226 = zeros(nx,ny);
         GB227 = zeros(nx,ny);GB228 = zeros(nx,ny);GB229 = zeros(nx,ny);
         for xg=1:1:nx
             for yg=1:1:nx
             GB221(xg,yg) =  1.*GB22(xg,yg);GB222(xg,yg) = .2.*GB22(xg,yg);
             GB223(xg,yg) = .3.*GB22(xg,yg);GB224(xg,yg) = .4.*GB22(xg,yg);
             GB225(xg,yg) = .5.*GB22(xg,yg);GB226(xg,yg) = .6.*GB22(xg,yg);
             GB227(xg,yg) = .7.*GB22(xg,yg);GB228(xg,yg) = .8.*GB22(xg,yg);
             GB229(xg,yg) = .9.*GB22(xg,yg);
             end
         end
         G122 = Gaussian.*G22;  G1Tc = G1Tc + G122;
         GTc1 = GTc1 + Gaussian.*(G22-GB221);
         GTc2 = GTc2 + Gaussian.*(G22-GB222);
         GTc3 = GTc3 + Gaussian.*(G22-GB223);
         GTc4 = GTc4 + Gaussian.*(G22-GB224);
         GTc5 = GTc5 + Gaussian.*(G22-GB225);
         GTc6 = GTc6 + Gaussian.*(G22-GB226);
         GTc7 = GTc7 + Gaussian.*(G22-GB227);
         GTc8 = GTc8 + Gaussian.*(G22-GB228);
         GTc9 = GTc9 + Gaussian.*(G22-GB229);
          logv22 = ifftshift(ifft2(ifftshift(G122))); v22 = exp(logv22); vTi = vTi + v22;
         if(showCep ==1)
         figure('Name', 'Reconstructed V #22');
         surf(abs(v22));shading flat;colormap(jet); view(0,-90);colorbar
         end
     case 23
         if(showEwald == 1)
         figure('Name', 'Ewald Circle for Source #23');scatter3(x23,y23,z23,5,z23,'filled');
         view(0,90);colormap(jet);colorbar;
         end
         fprintf('\nProcessing Files for Source #23 .');statuspt = nx/10;percentdone = 0;
         for ix=1:nx
             count = count+1;
             if(count >= statuspt)
                 fprintf('.');count = 0;percentdone = percentdone+1;
                 if(percentdone == 2)
                     fprintf('25%%')
```

```
        end
        if(percentdone == 5)
            fprintf('50%%');
        end
        if(percentdone == 7)
            fprintf('.75%%')
        end
    end
    for iy=1:ny
        vsi23(ix,iy) = sum(      In23(:,3) .* exp(i.*2.0.*pi.* ...
            (In23(:,1).*(dx.*(ix-nx/2-1)+sx)+ In23(:,2).*(dy.*(iy-ny/2-1)+sy))));
        vsib23(ix,iy) = sum(      Inb23(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb23(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb23(:,2).*(dy.*(iy-ny/2-1)+sy))));
    end
end
clear In23;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #23');
surf(abs(vsi23));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
    for rr = 1:r/2
        vsi23c(rr,ccc) = vsi23(rr*2,ccc*2);vsib23c(rr,ccc) = vsib23(rr*2,ccc*2);
    end;
end;
[r1,c1] = size(vsi23c);
VSI23 = fftshift(fft2(fftshift(vsi23c)));AbsVSI23 = abs(VSI23);
MaxAbsVSI23 = max(max(AbsVSI23));
RefVal23 = (MaxAbsVSI23+RefBias)*RefMult;
VSI23Causal = zeros(NewSize,NewSize); %
VSI23Causal(1:r1,NewMidPoint:NewSize) = VSI23;
VSI23Causal(RefRowPos,RefColPos) = RefVal23;
VSIB23 = fftshift(fft2(fftshift(vsib23c)));AbsVSIB23 = abs(VSIB23);
MaxAbsVSIB23 = max(max(AbsVSIB23));
RefValB23 = (MaxAbsVSIB23+RefBias)*RefMult;
VSIB23Causal = zeros(NewSize,NewSize);
VSIB23Causal(1:r1,NewMidPoint:NewSize) = VSIB23;
VSIB23Causal(RefRowPos,RefColPos) = RefValB23;
vsi23Ref = ifftshift(ifft2(ifftshift(VSI23Causal)));
vsi23Phase = atan2(imag(vsi23Ref),real(vsi23Ref));
vsib23Ref = ifftshift(ifft2(ifftshift(VSIB23Causal)));
logvsi23Ref = log(vsi23Ref);  logvsib23Ref = log(vsib23Ref);
G23 = fftshift(fft2(fftshift(logvsi23Ref)));
GB23 = fftshift(fft2(fftshift(logvsib23Ref)));
GB231 = zeros(nx,ny);GB232 = zeros(nx,ny);GB233 = zeros(nx,ny);
GB234 = zeros(nx,ny);GB235 = zeros(nx,ny);GB236 = zeros(nx,ny);
```

```
   GB237 = zeros(nx,ny);GB238 = zeros(nx,ny);GB239 = zeros(nx,ny);
   for xg=1:1:nx
      for yg=1:1:nx
      GB231(xg,yg) =  1.*GB23(xg,yg);GB232(xg,yg) = .2.*GB23(xg,yg);
      GB233(xg,yg) = .3.*GB23(xg,yg);GB234(xg,yg) = .4.*GB23(xg,yg);
      GB235(xg,yg) = .5.*GB23(xg,yg);GB236(xg,yg) = .6.*GB23(xg,yg);
      GB237(xg,yg) = .7.*GB23(xg,yg);GB238(xg,yg) = .8.*GB23(xg,yg);
      GB239(xg,yg) = .9.*GB23(xg,yg);
      end
   end
   G123 = Gaussian.*G23;  G1Tc = G1Tc + G123;
   GTc1 = GTc1 + Gaussian.*(G23-GB231);
   GTc2 = GTc2 + Gaussian.*(G23-GB232);
   GTc3 = GTc3 + Gaussian.*(G23-GB233);
   GTc4 = GTc4 + Gaussian.*(G23-GB234);
   GTc5 = GTc5 + Gaussian.*(G23-GB235);
   GTc6 = GTc6 + Gaussian.*(G23-GB236);
   GTc7 = GTc7 + Gaussian.*(G23-GB237);
   GTc8 = GTc8 + Gaussian.*(G23-GB238);
   GTc9 = GTc9 + Gaussian.*(G23-GB239);
   logv23 = ifftshift(ifft2(ifftshift(G123))); v23 = exp(logv23); vTi = vTi + v23;
   if(showCep ==1)
   figure('Name', 'Reconstructed V #23');
   surf(abs(v23));shading flat;colormap(jet); view(0,-90);colorbar
   end
case 24
   if(showEwald == 1)
   figure('Name', 'Ewald Circle for Source #24');scatter3(x24,y24,z24,5,z24,'filled');
   view(0,90);colormap(jet);colorbar;
   end
   fprintf('\nProcessing Files for Source #24 .');statuspt = nx/10;percentdone = 0;
   for ix=1:nx
      count = count+1;
      if(count >= statuspt)
         fprintf('.');count = 0;percentdone = percentdone+1;
         if(percentdone == 2)
            fprintf('25%%')
         end
         if(percentdone == 5)
            fprintf('50%%');
         end
         if(percentdone == 7)
            fprintf('.75%%')
         end
      end
      for iy=1:ny
```

```
        vsi24(ix,iy) = sum(      In24(:,3) .* exp(i.*2.0.*pi.* ...
            (In24(:,1).*(dx.*(ix-nx/2-1)+sx)+ In24(:,2).*(dy.*(iy-ny/2-1)+sy))));
        vsib24(ix,iy) = sum(      Inb24(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb24(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb24(:,2).*(dy.*(iy-ny/2-1)+sy))));
    end
end
clear In24;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #24');
surf(abs(vsi24));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
    for rr = 1:r/2
        vsi24c(rr,ccc) = vsi24(rr*2,ccc*2);vsib24c(rr,ccc) = vsib24(rr*2,ccc*2);
    end;
end;
[r1,c1] = size(vsi24c);
VSI24 = fftshift(fft2(fftshift(vsi24c))); AbsVSI24 = abs(VSI24);
MaxAbsVSI24 = max(max(AbsVSI24));
RefVal24 = (MaxAbsVSI24+RefBias)*RefMult;
VSI24Causal = zeros(NewSize,NewSize); %
VSI24Causal(1:r1,NewMidPoint:NewSize) = VSI24;
VSI24Causal(RefRowPos,RefColPos) = RefVal24;
VSIB24 = fftshift(fft2(fftshift(vsib24c)));AbsVSIB24 = abs(VSIB24);
MaxAbsVSIB24 = max(max(AbsVSIB24));
RefValB24 = (MaxAbsVSIB24+RefBias)*RefMult;
VSIB24Causal = zeros(NewSize,NewSize);
VSIB24Causal(1:r1,NewMidPoint:NewSize) = VSIB24;
VSIB24Causal(RefRowPos,RefColPos) = RefValB24;
vsi24Ref = ifftshift(ifft2(ifftshift(VSI24Causal)));
vsi24Phase = atan2(imag(vsi24Ref),real(vsi24Ref));
vsib24Ref = ifftshift(ifft2(ifftshift(VSIB24Causal)));
logvsi24Ref = log(vsi24Ref);
logvsib24Ref = log(vsib24Ref);
G24 = fftshift(fft2(fftshift(logvsi24Ref)));
GB24 = fftshift(fft2(fftshift(logvsib24Ref)));
GB241 = zeros(nx,ny);GB242 = zeros(nx,ny);GB243 = zeros(nx,ny);
GB244 = zeros(nx,ny);GB245 = zeros(nx,ny);GB246 = zeros(nx,ny);
GB247 = zeros(nx,ny);GB248 = zeros(nx,ny);GB249 = zeros(nx,ny);
for xg=1:1:nx
    for yg=1:1:nx
    GB241(xg,yg) =  1.*GB24(xg,yg);GB242(xg,yg) = .2.*GB24(xg,yg);
    GB243(xg,yg) = .3.*GB24(xg,yg);GB244(xg,yg) = .4.*GB24(xg,yg);
    GB245(xg,yg) = .5.*GB24(xg,yg);GB246(xg,yg) = .6.*GB24(xg,yg);
    GB247(xg,yg) = .7.*GB24(xg,yg);GB248(xg,yg) = .8.*GB24(xg,yg);
    GB249(xg,yg) = .9.*GB24(xg,yg);
```

```
        end
    end
    G124 = Gaussian.*G24;  G1Tc = G1Tc + G124;
    GTc1 = GTc1 + Gaussian.*(G24-GB241);
    GTc2 = GTc2 + Gaussian.*(G24-GB242);
    GTc3 = GTc3 + Gaussian.*(G24-GB243);
    GTc4 = GTc4 + Gaussian.*(G24-GB244);
    GTc5 = GTc5 + Gaussian.*(G24-GB245);
    GTc6 = GTc6 + Gaussian.*(G24-GB246);
    GTc7 = GTc7 + Gaussian.*(G24-GB247);
    GTc8 = GTc8 + Gaussian.*(G24-GB248);
    GTc9 = GTc9 + Gaussian.*(G24-GB249);
    logv24 = ifftshift(ifft2(ifftshift(G124))); v24 = exp(logv24); vTi = vTi + v24;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #24');
    surf(abs(v24));shading flat;colormap(jet); view(0,-90);colorbar
    end
case 25
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #25');scatter3(x25,y25,z25,5,z25,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #25 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi25(ix,iy) = sum(      In25(:,3) .* exp(i.*2.0.*pi.* ...
                (In25(:,1).*(dx.*(ix-nx/2-1)+sx)+ In25(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib25(ix,iy) = sum(      Inb25(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb25(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb25(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In25;fprintf('. Done\n');
    if (showBorn == 1)
```

```
figure('Name', 'Born VSi for Source #25');
surf(abs(vsi25));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
   for rr = 1:r/2
      vsi25c(rr,ccc) = vsi25(rr*2,ccc*2);vsib25c(rr,ccc) = vsib25(rr*2,ccc*2);
   end;
end;
[r1,c1] = size(vsi25c);
VSI25 = fftshift(fft2(fftshift(vsi25c)));AbsVSI25 = abs(VSI25);
MaxAbsVSI25 = max(max(AbsVSI25));
RefVal25 = (MaxAbsVSI25+RefBias)*RefMult;
VSI25Causal = zeros(NewSize,NewSize); %
VSI25Causal(1:r1,NewMidPoint:NewSize) = VSI25;
VSI25Causal(RefRowPos,RefColPos) = RefVal25;
VSIB25 = fftshift(fft2(fftshift(vsib25c)));AbsVSIB25 = abs(VSIB25);
MaxAbsVSIB25 = max(max(AbsVSIB25));
RefValB25 = (MaxAbsVSIB25+RefBias)*RefMult;
VSIB25Causal = zeros(NewSize,NewSize);
VSIB25Causal(1:r1,NewMidPoint:NewSize) = VSIB25;
VSIB25Causal(RefRowPos,RefColPos) = RefValB25;
vsi25Ref = ifftshift(ifft2(ifftshift(VSI25Causal)));
vsi25Phase = atan2(imag(vsi25Ref),real(vsi25Ref));
vsib25Ref = ifftshift(ifft2(ifftshift(VSIB25Causal)));
logvsi25Ref = log(vsi25Ref);  logvsib25Ref = log(vsib25Ref);
G25 = fftshift(fft2(fftshift(logvsi25Ref)));
GB25 = fftshift(fft2(fftshift(logvsib25Ref)));
GB251 = zeros(nx,ny);GB252 = zeros(nx,ny);GB253 = zeros(nx,ny);
GB254 = zeros(nx,ny);GB255 = zeros(nx,ny);GB256 = zeros(nx,ny);
GB257 = zeros(nx,ny);GB258 = zeros(nx,ny);GB259 = zeros(nx,ny);
for xg=1:1:nx
   for yg=1:1:nx
   GB251(xg,yg) =  1.*GB25(xg,yg);GB252(xg,yg) = .2.*GB25(xg,yg);
   GB253(xg,yg) = .3.*GB25(xg,yg);GB254(xg,yg) = .4.*GB25(xg,yg);
   GB255(xg,yg) = .5.*GB25(xg,yg);GB256(xg,yg) = .6.*GB25(xg,yg);
   GB257(xg,yg) = .7.*GB25(xg,yg);GB258(xg,yg) = .8.*GB25(xg,yg);
   GB259(xg,yg) = .9.*GB25(xg,yg);
   end
end
G125 = Gaussian.*G25;  G1Tc = G1Tc + G125;
GTc1 = GTc1 + Gaussian.*(G25-GB251);
GTc2 = GTc2 + Gaussian.*(G25-GB252);
GTc3 = GTc3 + Gaussian.*(G25-GB253);
GTc4 = GTc4 + Gaussian.*(G25-GB254);
GTc5 = GTc5 + Gaussian.*(G25-GB255);
GTc6 = GTc6 + Gaussian.*(G25-GB256);
```

```
    GTc7 = GTc7 + Gaussian.*(G25-GB257);
    GTc8 = GTc8 + Gaussian.*(G25-GB258);
    GTc9 = GTc9 + Gaussian.*(G25-GB259);
    logv25 = ifftshift(ifft2(ifftshift(G125))); v25 = exp(logv25); vTi = vTi + v25;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #25');
    surf(abs(v25));shading flat;colormap(jet); view(0,-90);colorbar
    end
case 26
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #26');scatter3(x26,y26,z26,5,z26,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #26 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi26(ix,iy) = sum(     In26(:,3) .* exp(i.*2.0.*pi.* ...
                (In26(:,1).*(dx.*(ix-nx/2-1)+sx)+ In26(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib26(ix,iy) = sum(     Inb26(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb26(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb26(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In26;fprintf('. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #26');
    surf(abs(vsi26));shading flat;colormap(jet); view(0,-90);colorbar
    end
    for ccc = 1:c/2
        for rr = 1:r/2
            vsi26c(rr,ccc) = vsi26(rr*2,ccc*2);vsib26c(rr,ccc) = vsib26(rr*2,ccc*2);
        end;
    end;
    [r1,c1] = size(vsi26c);
```

```
VSI26 = fftshift(fft2(fftshift(vsi26c)));AbsVSI26 = abs(VSI26);
MaxAbsVSI26 = max(max(AbsVSI26));
RefVal26 = (MaxAbsVSI26+RefBias)*RefMult;
VSI26Causal = zeros(NewSize,NewSize); %
VSI26Causal(1:r1,NewMidPoint:NewSize) = VSI26;
VSI26Causal(RefRowPos,RefColPos) = RefVal26;
VSIB26 = fftshift(fft2(fftshift(vsib26c)));AbsVSIB26 = abs(VSIB26);
MaxAbsVSIB26 = max(max(AbsVSIB26));
RefValB26 = (MaxAbsVSIB26+RefBias)*RefMult;
VSIB26Causal = zeros(NewSize,NewSize);
VSIB26Causal(1:r1,NewMidPoint:NewSize) = VSIB26;
VSIB26Causal(RefRowPos,RefColPos) = RefValB26;
vsi26Ref = ifftshift(ifft2(ifftshift(VSI26Causal)));
vsi26Phase = atan2(imag(vsi26Ref),real(vsi26Ref));
vsib26Ref = ifftshift(ifft2(ifftshift(VSIB26Causal)));
logvsi26Ref = log(vsi26Ref);
logvsib26Ref = log(vsib26Ref);
G26 = fftshift(fft2(fftshift(logvsi26Ref)));
GB26 = fftshift(fft2(fftshift(logvsib26Ref)));
GB261 = zeros(nx,ny);GB262 = zeros(nx,ny);GB263 = zeros(nx,ny);
GB264 = zeros(nx,ny);GB265 = zeros(nx,ny);GB266 = zeros(nx,ny);
GB267 = zeros(nx,ny);GB268 = zeros(nx,ny);GB269 = zeros(nx,ny);
for xg=1:1:nx
   for yg=1:1:nx
   GB261(xg,yg) =  1.*GB26(xg,yg);GB262(xg,yg) = .2.*GB26(xg,yg);
   GB263(xg,yg) = .3.*GB26(xg,yg);GB264(xg,yg) = .4.*GB26(xg,yg);
   GB265(xg,yg) = .5.*GB26(xg,yg);GB266(xg,yg) = .6.*GB26(xg,yg);
   GB267(xg,yg) = .7.*GB26(xg,yg);GB268(xg,yg) = .8.*GB26(xg,yg);
   GB269(xg,yg) = .9.*GB26(xg,yg);
   end
end
G126 = Gaussian.*G26;  G1Tc = G1Tc + G126;
GTc1 = GTc1 + Gaussian.*(G26-GB261);
GTc2 = GTc2 + Gaussian.*(G26-GB262);
GTc3 = GTc3 + Gaussian.*(G26-GB263);
GTc4 = GTc4 + Gaussian.*(G26-GB264);
GTc5 = GTc5 + Gaussian.*(G26-GB265);
GTc6 = GTc6 + Gaussian.*(G26-GB266);
GTc7 = GTc7 + Gaussian.*(G26-GB267);
GTc8 = GTc8 + Gaussian.*(G26-GB268);
GTc9 = GTc9 + Gaussian.*(G26-GB269);
 logv26 = ifftshift(ifft2(ifftshift(G126))); v26 = exp(logv26); vTi = vTi + v26;
if(showCep ==1)
figure('Name', 'Reconstructed V #26');
surf(abs(v26));shading flat;colormap(jet); view(0,-90);colorbar
end
```

```
case 27
   if(showEwald == 1)
   figure('Name', 'Ewald Circle for Source #27');scatter3(x27,y27,z27,5,z27,'filled');
   view(0,90);colormap(jet);colorbar;
   end
   fprintf('\nProcessing Files for Source #27 .');statuspt = nx/10;percentdone = 0;
   for ix=1:nx
      count = count+1;
      if(count >= statuspt)
         fprintf('.');count = 0;percentdone = percentdone+1;
         if(percentdone == 2)
            fprintf('25%%')
         end
         if(percentdone == 5)
            fprintf('50%%');
         end
         if(percentdone == 7)
            fprintf('.75%%')
         end
      end
      for iy=1:ny
         vsi27(ix,iy) = sum(      In27(:,3) .* exp(i.*2.0.*pi.* ...
            (In27(:,1).*(dx.*(ix-nx/2-1)+sx)+ In27(:,2).*(dy.*(iy-ny/2-1)+sy))));
         vsib27(ix,iy) = sum(      Inb27(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb27(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb27(:,2).*(dy.*(iy-ny/2-1)+sy))));
      end
   end
   clear In27;fprintf('.. Done\n');
   if (showBorn == 1)
   figure('Name', 'Born VSi for Source #27');
   surf(abs(vsi27));shading flat;colormap(jet); view(0,-90);colorbar
   end
   for ccc = 1:c/2
      for rr = 1:r/2
         vsi27c(rr,ccc) = vsi27(rr*2,ccc*2);vsib27c(rr,ccc) = vsib27(rr*2,ccc*2);
      end;
   end;
   [r1,c1] = size(vsi27c);
   VSI27 = fftshift(fft2(fftshift(vsi27c)));AbsVSI27 = abs(VSI27);
   MaxAbsVSI27 = max(max(AbsVSI27));
   RefVal27 = (MaxAbsVSI27+RefBias)*RefMult;
   VSI27Causal = zeros(NewSize,NewSize); %
   VSI27Causal(1:r1,NewMidPoint:NewSize) = VSI27;
   VSI27Causal(RefRowPos,RefColPos) = RefVal27;
   VSIB27 = fftshift(fft2(fftshift(vsib27c)));AbsVSIB27 = abs(VSIB27);
   MaxAbsVSIB27 = max(max(AbsVSIB27));
```

```
    RefValB27 = (MaxAbsVSIB27+RefBias)*RefMult;
    VSIB27Causal = zeros(NewSize,NewSize);
    VSIB27Causal(1:r1,NewMidPoint:NewSize) = VSIB27;
    VSIB27Causal(RefRowPos,RefColPos) = RefValB27;
    vsi27Ref = ifftshift(ifft2(ifftshift(VSI27Causal)));
    vsi27Phase = atan2(imag(vsi27Ref),real(vsi27Ref));
    vsib27Ref = ifftshift(ifft2(ifftshift(VSIB27Causal)));
    logvsi27Ref = log(vsi27Ref);
    logvsib27Ref = log(vsib27Ref);
    G27 = fftshift(fft2(fftshift(logvsi27Ref)));
    GB27 = fftshift(fft2(fftshift(logvsib27Ref)));
    GB271 = zeros(nx,ny);GB272 = zeros(nx,ny);GB273 = zeros(nx,ny);
    GB274 = zeros(nx,ny);GB275 = zeros(nx,ny);GB276 = zeros(nx,ny);
    GB277 = zeros(nx,ny);GB278 = zeros(nx,ny);GB279 = zeros(nx,ny);
    for xg=1:1:nx
      for yg=1:1:nx
      GB271(xg,yg) =  1.*GB27(xg,yg);GB272(xg,yg) = .2.*GB27(xg,yg);
      GB273(xg,yg) = .3.*GB27(xg,yg);GB274(xg,yg) = .4.*GB27(xg,yg);
      GB275(xg,yg) = .5.*GB27(xg,yg);GB276(xg,yg) = .6.*GB27(xg,yg);
      GB277(xg,yg) = .7.*GB27(xg,yg);GB278(xg,yg) = .8.*GB27(xg,yg);
      GB279(xg,yg) = .9.*GB27(xg,yg);
      end
    end
    G127 = Gaussian.*G27; G1Tc = G1Tc + G127;
    GTc1 = GTc1 + Gaussian.*(G27-GB271);
    GTc2 = GTc2 + Gaussian.*(G27-GB272);
    GTc3 = GTc3 + Gaussian.*(G27-GB273);
    GTc4 = GTc4 + Gaussian.*(G27-GB274);
    GTc5 = GTc5 + Gaussian.*(G27-GB275);
    GTc6 = GTc6 + Gaussian.*(G27-GB276);
    GTc7 = GTc7 + Gaussian.*(G27-GB277);
    GTc8 = GTc8 + Gaussian.*(G27-GB278);
    GTc9 = GTc9 + Gaussian.*(G27-GB279);
    logv27 = ifftshift(ifft2(ifftshift(G127))); v27 = exp(logv27); vTi = vTi + v27;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #27');
    surf(abs(v27));shading flat;colormap(jet); view(0,-90);colorbar
    end
  case 28
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #28');scatter3(x28,y28,z28,5,z28,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #28 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
      count = count+1;
    if(count >= statuspt)
        fprintf('.');count = 0;percentdone = percentdone+1;
        if(percentdone == 2)
            fprintf('25%%')
        end
        if(percentdone == 5)
            fprintf('50%%');
        end
        if(percentdone == 7)
            fprintf('.75%%')
        end
    end
    for iy=1:ny
        vsi28(ix,iy) = sum(      In28(:,3) .* exp(i.*2.0.*pi.* ...
            (In28(:,1).*(dx.*(ix-nx/2-1)+sx)+ In28(:,2).*(dy.*(iy-ny/2-1)+sy))));
        vsib28(ix,iy) = sum(      Inb28(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb28(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb28(:,2).*(dy.*(iy-ny/2-1)+sy))));
    end
end
clear In28;fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #28');
surf(abs(vsi28));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
    for rr = 1:r/2
        vsi28c(rr,ccc) = vsi28(rr*2,ccc*2);vsib28c(rr,ccc) = vsib28(rr*2,ccc*2);
    end;
end;
[r1,c1] = size(vsi28c);
VSI28 = fftshift(fft2(fftshift(vsi28c)));AbsVSI28 = abs(VSI28);
MaxAbsVSI28 = max(max(AbsVSI28));
RefVal28 = (MaxAbsVSI28+RefBias)*RefMult;
VSI28Causal = zeros(NewSize,NewSize); %
VSI28Causal(1:r1,NewMidPoint:NewSize) = VSI28;
VSI28Causal(RefRowPos,RefColPos) = RefVal28;
VSIB28 = fftshift(fft2(fftshift(vsib28c)));AbsVSIB28 = abs(VSIB28);
MaxAbsVSIB28 = max(max(AbsVSIB28));
RefValB28 = (MaxAbsVSIB28+RefBias)*RefMult;
VSIB28Causal = zeros(NewSize,NewSize);
VSIB28Causal(1:r1,NewMidPoint:NewSize) = VSIB28;
VSIB28Causal(RefRowPos,RefColPos) = RefValB28;
vsi28Ref = ifftshift(ifft2(ifftshift(VSI28Causal)));
vsi28Phase = atan2(imag(vsi28Ref),real(vsi28Ref));
vsib28Ref = ifftshift(ifft2(ifftshift(VSIB28Causal)));
logvsi28Ref = log(vsi28Ref);  logvsib28Ref = log(vsib28Ref);
```

```
   G28 = fftshift(fft2(fftshift(logvsi28Ref)));
   GB28 = fftshift(fft2(fftshift(logvsib28Ref)));
   GB281 = zeros(nx,ny);GB282 = zeros(nx,ny);GB283 = zeros(nx,ny);
   GB284 = zeros(nx,ny);GB285 = zeros(nx,ny);GB286 = zeros(nx,ny);
   GB287 = zeros(nx,ny);GB288 = zeros(nx,ny); GB289 = zeros(nx,ny);
   for xg=1:1:nx
      for yg=1:1:nx
      GB281(xg,yg) =  1.*GB28(xg,yg);GB282(xg,yg) = .2.*GB28(xg,yg);
      GB283(xg,yg) = .3.*GB28(xg,yg);GB284(xg,yg) = .4.*GB28(xg,yg);
      GB285(xg,yg) = .5.*GB28(xg,yg);GB286(xg,yg) = .6.*GB28(xg,yg);
      GB287(xg,yg) = .7.*GB28(xg,yg);GB288(xg,yg) = .8.*GB28(xg,yg);
      GB289(xg,yg) = .9.*GB28(xg,yg);
      end
   end
   G128 = Gaussian.*G28;  G1Tc = G1Tc + G128;
   GTc1 = GTc1 + Gaussian.*(G28-GB281);
   GTc2 = GTc2 + Gaussian.*(G28-GB282);
   GTc3 = GTc3 + Gaussian.*(G28-GB283);
   GTc4 = GTc4 + Gaussian.*(G28-GB284);
   GTc5 = GTc5 + Gaussian.*(G28-GB285);
   GTc6 = GTc6 + Gaussian.*(G28-GB286);
   GTc7 = GTc7 + Gaussian.*(G28-GB287);
   GTc8 = GTc8 + Gaussian.*(G28-GB288);
   GTc9 = GTc9 + Gaussian.*(G28-GB289);
   logv28 = ifftshift(ifft2(ifftshift(G128))); v28 = exp(logv28); vTi = vTi + v28;
   if(showCep ==1)
   figure('Name', 'Reconstructed V #28');
   surf(abs(v28));shading flat;colormap(jet); view(0,-90);colorbar
   end
case 29
   if(showEwald == 1)
   figure('Name', 'Ewald Circle for Source #29');scatter3(x29,y29,z29,5,z29,'filled');
   view(0,90);colormap(jet);colorbar;
   end
   fprintf('\nProcessing Files for Source #29 .');statuspt = nx/10;percentdone = 0;
   for ix=1:nx
      count = count+1;
      if(count >= statuspt)
         fprintf('.');count = 0;percentdone = percentdone+1;
         if(percentdone == 2)
            fprintf('25%%')
         end
         if(percentdone == 5)
            fprintf('50%%');
         end
         if(percentdone == 7)
```

```
            fprintf('.75%%')
         end
      end
      for iy=1:ny
         vsi29(ix,iy) = sum(      In29(:,3) .* exp(i.*2.0.*pi.* ...
            (In29(:,1).*(dx.*(ix-nx/2-1)+sx)+ In29(:,2).*(dy.*(iy-ny/2-1)+sy))));
         vsib29(ix,iy) = sum(      Inb29(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb29(:,1).*(dx.*(ix-nx/2-1)+sx)+ In29(:,2).*(dy.*(iy-ny/2-1)+sy))));
      end
   end
   clear In29; fprintf('. Done\n');
   if (showBorn == 1)
   figure('Name', 'Born VSi for Source #29');
   surf(abs(vsi29));shading flat;colormap(jet); view(0,-90);colorbar
   end
   for ccc = 1:c/2
      for rr = 1:r/2
         vsi29c(rr,ccc) = vsi29(rr*2,ccc*2);vsib29c(rr,ccc) = vsib29(rr*2,ccc*2);
      end;
   end;
   [r1,c1] = size(vsi29c);
   VSI29 = fftshift(fft2(fftshift(vsi29c)));AbsVSI29 = abs(VSI29);
   MaxAbsVSI29 = max(max(AbsVSI29));
   RefVal29 = (MaxAbsVSI29+RefBias)*RefMult;
   VSI29Causal = zeros(NewSize,NewSize); %
   VSI29Causal(1:r1,NewMidPoint:NewSize) = VSI29;
   VSI29Causal(RefRowPos,RefColPos) = RefVal29;
   VSIB29 = fftshift(fft2(fftshift(vsib29c)));AbsVSIB29 = abs(VSIB29);
   MaxAbsVSIB29 = max(max(AbsVSIB20));
   RefValB29 = (MaxAbsVSIB29+RefBias)*RefMult;
   VSIB29Causal = zeros(NewSize,NewSize);
   VSIB29Causal(1:r1,NewMidPoint:NewSize) = VSIB29;
   VSIB29Causal(RefRowPos,RefColPos) = RefValB29;
   vsi29Ref = ifftshift(ifft2(ifftshift(VSI29Causal)));
   vsi29Phase = atan2(imag(vsi29Ref),real(vsi29Ref)); % Phase of vsi
   vsib29Ref = ifftshift(ifft2(ifftshift(VSIB29Causal)));
   logvsi29Ref = log(vsi29Ref);  logvsib29Ref = log(vsib29Ref);
   G29 = fftshift(fft2(fftshift(logvsi29Ref)));
   GB29 = fftshift(fft2(fftshift(logvsib29Ref)));
   GB291 = zeros(nx,ny);GB292 = zeros(nx,ny);GB293 = zeros(nx,ny);
   GB294 = zeros(nx,ny);GB295 = zeros(nx,ny);GB296 = zeros(nx,ny);
   GB297 = zeros(nx,ny);GB298 = zeros(nx,ny);GB299 = zeros(nx,ny);
   for xg=1:1:nx
      for yg=1:1:nx
      GB291(xg,yg) =  1.*GB29(xg,yg);GB292(xg,yg) = .2.*GB29(xg,yg);
      GB293(xg,yg) = .3.*GB29(xg,yg);GB294(xg,yg) = .4.*GB29(xg,yg);
```

```
            GB295(xg,yg) = .5.*GB29(xg,yg);GB296(xg,yg) = .6.*GB29(xg,yg);
            GB297(xg,yg) = .7.*GB29(xg,yg);GB298(xg,yg) = .8.*GB29(xg,yg);
            GB299(xg,yg) = .9.*GB29(xg,yg);
            end
        end
        G129 = Gaussian.*G29;  G1Tc = G1Tc + G129;
        GTc1 = GTc1 + Gaussian.*(G29-GB291);
        GTc2 = GTc2 + Gaussian.*(G29-GB292);
        GTc3 = GTc3 + Gaussian.*(G29-GB293);
        GTc4 = GTc4 + Gaussian.*(G29-GB294);
        GTc5 = GTc5 + Gaussian.*(G29-GB295);
        GTc6 = GTc6 + Gaussian.*(G29-GB296);
        GTc7 = GTc7 + Gaussian.*(G29-GB297);
        GTc8 = GTc8 + Gaussian.*(G29-GB298);
        GTc9 = GTc9 + Gaussian.*(G29-GB299);
        logv29 = ifftshift(ifft2(ifftshift(G129))); v29 = exp(logv29); vTi = vTi + v29;
        if(showCep ==1)
        figure('Name', 'Reconstructed V #29');
        surf(abs(v29));shading flat;colormap(jet); view(0,-90);colorbar
        end
    case 30
        if(showEwald == 1)
        figure('Name', 'Ewald Circle for Source #30');scatter3(x30,y30,z30,5,z30,'filled');
        view(0,90);colormap(jet);colorbar;
        end
        fprintf('\nProcessing Files for Source #30 .');statuspt = nx/10;percentdone = 0;
        for ix=1:nx
            count = count+1;
            if(count >= statuspt)
                fprintf('.');count = 0;percentdone = percentdone+1;
                if(percentdone == 2)
                    fprintf('25%%')
                end
                if(percentdone == 5)
                    fprintf('50%%');
                end
                if(percentdone == 7)
                    fprintf('.75%%')
                end
            end
            for iy=1:ny
                vsi30(ix,iy) = sum(      In30(:,3) .* exp(i.*2.0.*pi.* ...
                    (In30(:,1).*(dx.*(ix-nx/2-1)+sx)+ In30(:,2).*(dy.*(iy-ny/2-1)+sy))));
                vsib30(ix,iy) = sum(      Inb30(:,3) .* exp(i.*2.0.*pi.* ...
                    (Inb30(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb30(:,2).*(dy.*(iy-ny/2-1)+sy))));
            end
```

```
end
clear In30; fprintf('. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #30');
surf(abs(vsi30));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
   for rr = 1:r/2
      vsi30c(rr,ccc) = vsi30(rr*2,ccc*2);vsib30c(rr,ccc) = vsib30(rr*2,ccc*2);
   end;
end;
[r1,c1] = size(vsi30c);
VSI30 = fftshift(fft2(fftshift(vsi30c)));AbsVSI30 = abs(VSI30);
MaxAbsVSI30 = max(max(AbsVSI30));
RefVal30 = (MaxAbsVSI30+RefBias)*RefMult;
VSI30Causal = zeros(NewSize,NewSize); %
VSI30Causal(1:r1,NewMidPoint:NewSize) = VSI30;
VSI30Causal(RefRowPos,RefColPos) = RefVal30;
VSIB30 = fftshift(fft2(fftshift(vsib30c)));AbsVSIB30 = abs(VSIB30);
MaxAbsVSIB30 = max(max(AbsVSIB30));
RefValB30 = (MaxAbsVSIB30+RefBias)*RefMult;
VSIB30Causal = zeros(NewSize,NewSize);
VSIB30Causal(1:r1,NewMidPoint:NewSize) = VSIB30;
VSIB30Causal(RefRowPos,RefColPos) = RefValB30;

vsi30Ref = ifftshift(ifft2(ifftshift(VSI30Causal)));
vsi30Phase = atan2(imag(vsi30Ref),real(vsi30Ref)); % Phase of vsi
vsib30Ref = ifftshift(ifft2(ifftshift(VSIB30Causal)));
logvsi30Ref = log(vsi30Ref);  logvsib30Ref = log(vsib30Ref);
G30 = fftshift(fft2(fftshift(logvsi30Ref)));
GB30 = fftshift(fft2(fftshift(logvsib30Ref)));
GB301 = zeros(nx,ny);GB302 = zeros(nx,ny);GB303 = zeros(nx,ny);
GB304 = zeros(nx,ny);GB305 = zeros(nx,ny);GB306 = zeros(nx,ny);
GB307 = zeros(nx,ny);GB308 = zeros(nx,ny);GB309 = zeros(nx,ny);
for xg=1:1:nx
   for yg=1:1:nx
   GB301(xg,yg) =  1.*GB30(xg,yg);GB302(xg,yg) = .2.*GB30(xg,yg);
   GB303(xg,yg) = .3.*GB30(xg,yg);GB304(xg,yg) = .4.*GB30(xg,yg);
   GB305(xg,yg) = .5.*GB30(xg,yg);GB306(xg,yg) = .6.*GB30(xg,yg);
   GB307(xg,yg) = .7.*GB30(xg,yg);GB308(xg,yg) = .8.*GB30(xg,yg);
   GB309(xg,yg) = .9.*GB30(xg,yg);
   end
end
G130 = Gaussian.*G30;  G1Tc = G1Tc + G130;
GTc1 = GTc1 + Gaussian.*(G30-GB301);
GTc2 = GTc2 + Gaussian.*(G30-GB302);
```

```
  GTc3 = GTc3 + Gaussian.*(G30-GB303);
  GTc4 = GTc4 + Gaussian.*(G30-GB304);
  GTc5 = GTc5 + Gaussian.*(G30-GB305);
  GTc6 = GTc6 + Gaussian.*(G30-GB306);
  GTc7 = GTc7 + Gaussian.*(G30-GB307);
  GTc8 = GTc8 + Gaussian.*(G30-GB308);
  GTc9 = GTc9 + Gaussian.*(G30-GB309);
  logv30 = ifftshift(ifft2(ifftshift(G130))); v30 = exp(logv30); vTi = vTi + v30;
  if(showCep ==1)
  figure('Name', 'Reconstructed V #30');
  surf(abs(v30));shading flat;colormap(jet); view(0,-90);colorbar
  end
case 31
  if(showEwald == 1)
  figure('Name', 'Ewald Circle for Source #31');scatter3(x31,y31,z31,5,z31,'filled');
  view(0,90);colormap(jet);colorbar;
  end
  fprintf('\nProcessing Files for Source #31 .');statuspt = nx/10;percentdone = 0;
  for ix=1:nx
    count = count+1;
    if(count >= statuspt)
      fprintf('.');count = 0;percentdone = percentdone+1;
      if(percentdone == 2)
        fprintf('25%%')
      end
      if(percentdone == 5)
        fprintf('50%%');
      end
      if(percentdone == 7)
        fprintf('.75%%')
      end
    end
    for iy=1:ny
      vsi31(ix,iy) = sum(    In31(:,3) .* exp(i.*2.0.*pi.* ...
        (In31(:,1).*(dx.*(ix-nx/2-1)+sx)+ In31(:,2).*(dy.*(iy-ny/2-1)+sy))));
      vsib31(ix,iy) = sum(    Inb31(:,3) .* exp(i.*2.0.*pi.* ...
        (Inb31(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb31(:,2).*(dy.*(iy-ny/2-1)+sy))));
    end
  end
  clear In31;fprintf('. Done\n');
  if (showBorn == 1)
  figure('Name', 'Born VSi for Source #31');
  surf(abs(vsi31));shading flat;colormap(jet); view(0,-90);colorbar
  end
  for ccc = 1:c/2
    for rr = 1:r/2
```

```
        vsi31c(rr,ccc) = vsi31(rr*2,ccc*2);vsib31c(rr,ccc) = vsib31(rr*2,ccc*2);
    end;
end;
[r1,c1] = size(vsi31c);
VSI31 = fftshift(fft2(fftshift(vsi31c))); AbsVSI31 = abs(VSI31);
MaxAbsVSI31 = max(max(AbsVSI31));
RefVal31 = (MaxAbsVSI31+RefBias)*RefMult;
VSI31Causal = zeros(NewSize,NewSize); %
VSI31Causal(1:r1,NewMidPoint:NewSize) = VSI31;
VSIB31 = fftshift(fft2(fftshift(vsib31c)));AbsVSIB31 = abs(VSIB31);
MaxAbsVSIB31 = max(max(AbsVSIB31));
RefValB31 = (MaxAbsVSIB31+RefBias)*RefMult;
VSIB31Causal = zeros(NewSize,NewSize);
VSIB31Causal(1:r1,NewMidPoint:NewSize) = VSIB31;
VSIB31Causal(RefRowPos,RefColPos) = RefValB31;
VSI31Causal(RefRowPos,RefColPos) = RefVal30;
vsi31Ref = ifftshift(ifft2(ifftshift(VSI31Causal)));
vsi31Phase = atan2(imag(vsi31Ref),real(vsi31Ref));
vsib31Ref = ifftshift(ifft2(ifftshift(VSIB31Causal)));
logvsi31Ref = log(vsi31Ref);  logvsib31Ref = log(vsib31Ref);
G31 = fftshift(fft2(fftshift(logvsi31Ref)));
GB31 = fftshift(fft2(fftshift(logvsib31Ref)));
GB311 = zeros(nx,ny);GB312 = zeros(nx,ny);GB313 = zeros(nx,ny);
GB314 = zeros(nx,ny);GB315 = zeros(nx,ny);GB316 = zeros(nx,ny);
GB317 = zeros(nx,ny);GB318 = zeros(nx,ny);GB319 = zeros(nx,ny);
for xg=1:1:nx
    for yg=1:1:nx
    GB311(xg,yg) =  1.*GB31(xg,yg);GB312(xg,yg) = .2.*GB31(xg,yg);
    GB313(xg,yg) = .3.*GB31(xg,yg);GB314(xg,yg) = .4.*GB31(xg,yg);
    GB315(xg,yg) = .5.*GB31(xg,yg);GB316(xg,yg) = .6.*GB31(xg,yg);
    GB317(xg,yg) = .7.*GB31(xg,yg);GB318(xg,yg) = .8.*GB31(xg,yg);
    GB319(xg,yg) = .9.*GB31(xg,yg);
    end
end
G131 = Gaussian.*G31;  G1Tc = G1Tc + G131;
GTc1 = GTc1 + Gaussian.*(G31-GB311);
GTc2 = GTc2 + Gaussian.*(G31-GB312);
GTc3 = GTc3 + Gaussian.*(G31-GB313);
GTc4 = GTc4 + Gaussian.*(G31-GB314);
GTc5 = GTc5 + Gaussian.*(G31-GB315);
GTc6 = GTc6 + Gaussian.*(G31-GB316);
GTc7 = GTc7 + Gaussian.*(G31-GB317);
GTc8 = GTc8 + Gaussian.*(G31-GB318);
GTc9 = GTc9 + Gaussian.*(G31-GB319);
logv31 = ifftshift(ifft2(ifftshift(G131))); v31 = exp(logv31); vTi = vTi + v31;
if(showCep ==1)
```

```
    figure('Name', 'Reconstructed V #31');
    surf(abs(v31));shading flat;colormap(jet); view(0,-90);colorbar
    end
case 32
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #32');scatter3(x32,y32,z32,5,z32,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #32 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi32(ix,iy) = sum(      In32(:,3) .* exp(i.*2.0.*pi.* ...
                (In32(:,1).*(dx.*(ix-nx/2-1)+sx)+ In32(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib32(ix,iy) = sum(      Inb32(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb32(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb32(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In32;fprintf('. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #32');
    surf(abs(vsi32));shading flat;colormap(jet); view(0,-90);colorbar
    end
    for ccc = 1:c/2
        for rr = 1:r/2
            vsi32c(rr,ccc) = vsi32(rr*2,ccc*2);vsib32c(rr,ccc) = vsib32(rr*2,ccc*2);
        end;
    end;
    [r1,c1] = size(vsi32c);
    VSI32 = fftshift(fft2(fftshift(vsi32c)));AbsVSI32 = abs(VSI32);
    MaxAbsVSI32 = max(max(AbsVSI32));
    RefVal32 = (MaxAbsVSI32+RefBias)*RefMult;
    VSI32Causal = zeros(NewSize,NewSize); %
    VSI32Causal(1:r1,NewMidPoint:NewSize) = VSI32;
```

```
   VSI32Causal(RefRowPos,RefColPos) = RefVal32;
   VSIB32 = fftshift(fft2(fftshift(vsib32c)));AbsVSIB32 = abs(VSIB32);
   MaxAbsVSIB32 = max(max(AbsVSIB32));
   RefValB32 = (MaxAbsVSIB32+RefBias)*RefMult;
   VSIB32Causal = zeros(NewSize,NewSize);
   VSIB32Causal(1:r1,NewMidPoint:NewSize) = VSIB32;
   VSIB32Causal(RefRowPos,RefColPos) = RefValB32;
   vsi32Ref = ifftshift(ifft2(ifftshift(VSI32Causal)));
   vsi32Phase = atan2(imag(vsi32Ref),real(vsi32Ref));
   vsib32Ref = ifftshift(ifft2(ifftshift(VSIB32Causal)));
   logvsi32Ref = log(vsi32Ref);  logvsib32Ref = log(vsib32Ref);
   G32 = fftshift(fft2(fftshift(logvsi32Ref)));
   GB32 = fftshift(fft2(fftshift(logvsib32Ref)));
   GB321 = zeros(nx,ny);GB322 = zeros(nx,ny);GB323 = zeros(nx,ny);
   GB324 = zeros(nx,ny);GB325 = zeros(nx,ny);GB326 = zeros(nx,ny);
   GB327 = zeros(nx,ny);GB328 = zeros(nx,ny);GB329 = zeros(nx,ny);
   for xg=1:1:nx
      for yg=1:1:nx
      GB321(xg,yg) =  1.*GB32(xg,yg);GB322(xg,yg) = .2.*GB32(xg,yg);
      GB323(xg,yg) = .3.*GB32(xg,yg);GB324(xg,yg) = .4.*GB32(xg,yg);
      GB325(xg,yg) = .5.*GB32(xg,yg);GB326(xg,yg) = .6.*GB32(xg,yg);
      GB327(xg,yg) = .7.*GB32(xg,yg);GB328(xg,yg) = .8.*GB32(xg,yg);
      GB329(xg,yg) = .9.*GB32(xg,yg);
      end
   end
   G132 = Gaussian.*G32;  G1Tc = G1Tc + G132;
   GTc1 = GTc1 + Gaussian.*(G32-GB321);
   GTc2 = GTc2 + Gaussian.*(G32-GB322);
   GTc3 = GTc3 + Gaussian.*(G32-GB323);
   GTc4 = GTc4 + Gaussian.*(G32-GB324);
   GTc5 = GTc5 + Gaussian.*(G32-GB325);
   GTc6 = GTc6 + Gaussian.*(G32-GB326);
   GTc7 = GTc7 + Gaussian.*(G32-GB327);
   GTc8 = GTc8 + Gaussian.*(G32-GB328);
   GTc9 = GTc9 + Gaussian.*(G32-GB329);
   logv32 = ifftshift(ifft2(ifftshift(G132))); v32 = exp(logv32); vTi = vTi + v32;
   if(showCep ==1)
   figure('Name', 'Reconstructed V #32');
   surf(abs(v32));shading flat;colormap(jet); view(0,-90);colorbar
   end
case 33
   if(showEwald == 1)
   figure('Name', 'Ewald Circle for Source #33');scatter3(x33,y33,z33,5,z33,'filled');
   view(0,90);colormap(jet);colorbar;
   end
   fprintf('\nProcessing Files for Source #33 .');statuspt = nx/10;percentdone = 0;
```

```
for ix=1:nx
    count = count+1;
    if(count >= statuspt)
        fprintf('.');count = 0;percentdone = percentdone+1;
        if(percentdone == 2)
            fprintf('25%%')
        end
        if(percentdone == 5)
            fprintf('50%%');
        end
        if(percentdone == 7)
            fprintf('.75%%')
        end
    end
    for iy=1:ny
        vsi33(ix,iy) = sum(      In33(:,3) .* exp(i.*2.0.*pi.* ...
            (In33(:,1).*(dx.*(ix-nx/2-1)+sx)+ In33(:,2).*(dy.*(iy-ny/2-1)+sy))));
        vsib33(ix,iy) = sum(      Inb33(:,3) .* exp(i.*2.0.*pi.* ...
            (Inb33(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb33(:,2).*(dy.*(iy-ny/2-1)+sy))));
    end
end
clear In33;fprintf('.. Done\n');
if (showBorn == 1)
figure('Name', 'Born VSi for Source #33');

surf(abs(vsi33));shading flat;colormap(jet); view(0,-90);colorbar
end
for ccc = 1:c/2
    for rr = 1:r/2
        vsi33c(rr,ccc) = vsi33(rr*2,ccc*2);vsib33c(rr,ccc) = vsib33(rr*2,ccc*2);
    end;
end;
[r1,c1] = size(vsi33c);
VSI33 = fftshift(fft2(fftshift(vsi33c)));AbsVSI33 = abs(VSI33);
MaxAbsVSI33 = max(max(AbsVSI33));
RefVal33 = (MaxAbsVSI33+RefBias)*RefMult;
VSI33Causal = zeros(NewSize,NewSize); %
VSI33Causal(1:r1,NewMidPoint:NewSize) = VSI33;
VSI33Causal(RefRowPos,RefColPos) = RefVal33;
VSIB33 = fftshift(fft2(fftshift(vsib33c)));AbsVSIB33 = abs(VSIB33);
MaxAbsVSIB33 = max(max(AbsVSIB33));
RefValB33 = (MaxAbsVSIB33+RefBias)*RefMult;
VSIB33Causal = zeros(NewSize,NewSize);
VSIB33Causal(1:r1,NewMidPoint:NewSize) = VSIB33;
VSIB33Causal(RefRowPos,RefColPos) = RefValB33;
vsi33Ref = ifftshift(ifft2(ifftshift(VSI33Causal)));
```

```
    vsi33Phase = atan2(imag(vsi33Ref),real(vsi33Ref));
    vsib33Ref = ifftshift(ifft2(ifftshift(VSIB33Causal)));
    logvsi33Ref = log(vsi33Ref);  logvsib33Ref = log(vsib33Ref);
    G33 = fftshift(fft2(fftshift(logvsi33Ref)));
    GB33 = fftshift(fft2(fftshift(logvsib33Ref)));
    GB331 = zeros(nx,ny);GB332 = zeros(nx,ny);GB333 = zeros(nx,ny);
    GB334 = zeros(nx,ny);GB335 = zeros(nx,ny);GB336 = zeros(nx,ny);
    GB337 = zeros(nx,ny);GB338 = zeros(nx,ny);GB339 = zeros(nx,ny);
    for xg=1:1:nx
        for yg=1:1:nx
        GB331(xg,yg) =  1.*GB33(xg,yg);GB332(xg,yg) = .2.*GB33(xg,yg);
        GB333(xg,yg) = .3.*GB33(xg,yg);GB334(xg,yg) = .4.*GB33(xg,yg);
        GB335(xg,yg) = .5.*GB33(xg,yg);GB336(xg,yg) = .6.*GB33(xg,yg);
        GB337(xg,yg) = .7.*GB33(xg,yg);GB338(xg,yg) = .8.*GB33(xg,yg);
        GB339(xg,yg) = .9.*GB33(xg,yg);
        end
    end
    G133 = Gaussian.*G33;  G1Tc = G1Tc + G133;
    GTc1 = GTc1 + Gaussian.*(G33-GB331);
    GTc2 = GTc2 + Gaussian.*(G33-GB332);
    GTc3 = GTc3 + Gaussian.*(G33-GB333);
    GTc4 = GTc4 + Gaussian.*(G33-GB334);
    GTc5 = GTc5 + Gaussian.*(G33-GB335);
    GTc6 = GTc6 + Gaussian.*(G33-GB336);
    GTc7 = GTc7 + Gaussian.*(G33-GB337);
    GTc8 = GTc8 + Gaussian.*(G33-GB338);
    GTc9 = GTc9 + Gaussian.*(G33-GB339);
    logv33 = ifftshift(ifft2(ifftshift(G133))); v33 = exp(logv33); vTi = vTi + v33;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #33');
    surf(abs(v33));shading flat;colormap(jet); view(0,-90);colorbar
    end
case 34
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #34');scatter3(x34,y34,z34,5,z34,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #34 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi34(ix,iy) = sum(      In34(:,3) .* exp(i.*2.0.*pi.* ...
                (In34(:,1).*(dx.*(ix-nx/2-1)+sx)+ In34(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib34(ix,iy) = sum(      In34(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb34(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb34(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In34;fprintf('. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #34');
    surf(abs(vsi34));shading flat;colormap(jet); view(0,-90);colorbar
    end
    for ccc = 1:c/2
        for rr = 1:r/2
            vsi34c(rr,ccc) = vsi34(rr*2,ccc*2);vsib34c(rr,ccc) = vsib34(rr*2,ccc*2);
        end;
    end;
    [r1,c1] = size(vsi34c);
    VSI34 = fftshift(fft2(fftshift(vsi34c)));AbsVSI34 = abs(VSI34);
    MaxAbsVSI34 = max(max(AbsVSI34));
    RefVal34 = (MaxAbsVSI34+RefBias)*RefMult;
    VSI34Causal = zeros(NewSize,NewSize); %
    VSI34Causal(1:r1,NewMidPoint:NewSize) = VSI34;
    VSI34Causal(RefRowPos,RefColPos) = RefVal34;
    VSIB34 = fftshift(fft2(fftshift(vsib34c)));AbsVSIB34 = abs(VSIB34);
    MaxAbsVSIB34 = max(max(AbsVSIB34));
    RefValB34 = (MaxAbsVSIB34+RefBias)*RefMult;
    VSIB34Causal = zeros(NewSize,NewSize);
    VSIB34Causal(1:r1,NewMidPoint:NewSize) = VSIB34;
    VSIB34Causal(RefRowPos,RefColPos) = RefValB34;
    vsi34Ref = ifftshift(ifft2(ifftshift(VSI34Causal)));
    vsi34Phase = atan2(imag(vsi34Ref),real(vsi34Ref));
    vsib34Ref = ifftshift(ifft2(ifftshift(VSIB34Causal)));
    logvsi34Ref = log(vsi34Ref);  logvsib34Ref = log(vsib34Ref);
     G34 = fftshift(fft2(fftshift(logvsi34Ref)));
    GB34 = fftshift(fft2(fftshift(logvsib34Ref)));
    GB341 = zeros(nx,ny);GB342 = zeros(nx,ny);GB343 = zeros(nx,ny);
    GB344 = zeros(nx,ny);GB345 = zeros(nx,ny);GB346 = zeros(nx,ny);
    GB347 = zeros(nx,ny);GB348 = zeros(nx,ny);GB349 = zeros(nx,ny);
    for xg=1:1:nx
```

```
            for yg=1:1:nx
            GB341(xg,yg) =  1.*GB34(xg,yg);GB342(xg,yg) = .2.*GB34(xg,yg);
            GB343(xg,yg) = .3.*GB34(xg,yg);GB344(xg,yg) = .4.*GB34(xg,yg);
            GB345(xg,yg) = .5.*GB34(xg,yg);GB346(xg,yg) = .6.*GB34(xg,yg);
            GB347(xg,yg) = .7.*GB34(xg,yg);GB348(xg,yg) = .8.*GB34(xg,yg);
            GB349(xg,yg) = .9.*GB34(xg,yg);
            end
        end
        G134 = Gaussian.*G34;  G1Tc = G1Tc + G134;
        GTc1 = GTc1 + Gaussian.*(G34-GB341);
        GTc2 = GTc2 + Gaussian.*(G34-GB342);
        GTc3 = GTc3 + Gaussian.*(G34-GB343);
        GTc4 = GTc4 + Gaussian.*(G34-GB344);
        GTc5 = GTc5 + Gaussian.*(G34-GB345);
        GTc6 = GTc6 + Gaussian.*(G34-GB346);
        GTc7 = GTc7 + Gaussian.*(G34-GB347);
        GTc8 = GTc8 + Gaussian.*(G34-GB348);
        GTc9 = GTc9 + Gaussian.*(G34-GB349);
        logv34 = ifftshift(ifft2(ifftshift(G134))); v34 = exp(logv34); vTi = vTi + v34;
        if(showCep ==1)
        figure('Name', 'Reconstructed V #34');
        surf(abs(v34));shading flat;colormap(jet); view(0,-90);colorbar
        end
    case 35
        if(showEwald == 1)
        figure('Name', 'Ewald Circle for Source #35');scatter3(x35,y35,z35,5,z35,'filled');
        view(0,90);colormap(jet);colorbar;
        end
        fprintf('\nProcessing Files for Source #35 .');statuspt = nx/10;percentdone = 0;
        for ix=1:nx
            count = count+1;
            if(count >= statuspt)
                fprintf('.');count = 0;percentdone = percentdone+1;
                if(percentdone == 2)
                    fprintf('25%%')
                end
                if(percentdone == 5)
                    fprintf('50%%');
                end
                if(percentdone == 7)
                    fprintf('.75%%')
                end
            end
            for iy=1:ny
                vsi35(ix,iy) = sum(      In35(:,3) .* exp(i.*2.0.*pi.* ...
                    (In35(:,1).*(dx.*(ix-nx/2-1)+sx)+ In35(:,2).*(dy.*(iy-ny/2-1)+sy))));
                vsib35(ix,iy) = sum(      Inb35(:,3) .* exp(i.*2.0.*pi.* ...
                    (Inb35(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb35(:,2).*(dy.*(iy-ny/2-1)+sy))));
            end
        end
        clear In35;fprintf('. Done\n');
        if (showBorn == 1)
        figure('Name', 'Born VSi for Source #35');
        surf(abs(vsi35));shading flat;colormap(jet); view(0,-90);colorbar
        end
        for ccc = 1:c/2
            for rr = 1:r/2
                vsi35c(rr,ccc) = vsi35(rr*2,ccc*2);vsib35c(rr,ccc) = vsib35(rr*2,ccc*2);
            end;
        end;
        [r1,c1] = size(vsi35c);
        VSI35 = fftshift(fft2(fftshift(vsi35c)));AbsVSI35 = abs(VSI35);
        MaxAbsVSI35 = max(max(AbsVSI35));
        RefVal35 = (MaxAbsVSI35+RefBias)*RefMult;
        VSI35Causal = zeros(NewSize,NewSize); %
        VSI35Causal(1:r1,NewMidPoint:NewSize) = VSI35;
        VSI35Causal(RefRowPos,RefColPos) = RefVal35;
        VSIB35 = fftshift(fft2(fftshift(vsib35c)));AbsVSIB35 = abs(VSIB35);
        MaxAbsVSIB35 = max(max(AbsVSIB35));
        RefValB35 = (MaxAbsVSIB35+RefBias)*RefMult;
        VSIB35Causal = zeros(NewSize,NewSize);
        VSIB35Causal(1:r1,NewMidPoint:NewSize) = VSIB35;
        VSIB35Causal(RefRowPos,RefColPos) = RefValB35;
        vsi35Ref = ifftshift(ifft2(ifftshift(VSI35Causal)));
        vsi35Phase = atan2(imag(vsi35Ref),real(vsi35Ref)); % Phase of vsi
        vsib35Ref = ifftshift(ifft2(ifftshift(VSIB35Causal)));
        logvsi35Ref = log(vsi35Ref);  logvsib35Ref = log(vsib35Ref);
        G35 = fftshift(fft2(fftshift(logvsi35Ref)));
        GB35 = fftshift(fft2(fftshift(logvsib35Ref)));
        GB351 = zeros(nx,ny);GB352 = zeros(nx,ny);GB353 = zeros(nx,ny);
        GB354 = zeros(nx,ny);GB355 = zeros(nx,ny);GB356 = zeros(nx,ny);
        GB357 = zeros(nx,ny);GB358 = zeros(nx,ny);GB359 = zeros(nx,ny);
        for xg=1:1:nx
            for yg=1:1:nx
            GB351(xg,yg) =  1.*GB35(xg,yg);GB352(xg,yg) = .2.*GB35(xg,yg);
            GB353(xg,yg) = .3.*GB35(xg,yg);GB354(xg,yg) = .4.*GB35(xg,yg);
            GB355(xg,yg) = .5.*GB35(xg,yg);GB356(xg,yg) = .6.*GB35(xg,yg);
            GB357(xg,yg) = .7.*GB35(xg,yg);GB358(xg,yg) = .8.*GB35(xg,yg);
            GB359(xg,yg) = .9.*GB35(xg,yg);
            end
        end
        G135 = Gaussian.*G35;  G1Tc = G1Tc + G135;
```

```
    GTc1 = GTc1 + Gaussian.*(G35-GB351);
    GTc2 = GTc2 + Gaussian.*(G35-GB352);
    GTc3 = GTc3 + Gaussian.*(G35-GB353);
    GTc4 = GTc4 + Gaussian.*(G35-GB354);
    GTc5 = GTc5 + Gaussian.*(G35-GB355);
    GTc6 = GTc6 + Gaussian.*(G35-GB356);
    GTc7 = GTc7 + Gaussian.*(G35-GB357);
    GTc8 = GTc8 + Gaussian.*(G35-GB358);
    GTc9 = GTc9 + Gaussian.*(G35-GB359);
    logv35 = ifftshift(ifft2(ifftshift(G135))); v35 = exp(logv35); vTi = vTi + v35;
    if(showCep ==1)
    figure('Name', 'Reconstructed V #35');
    surf(abs(v35));shading flat;colormap(jet); view(0,-90);colorbar
    end
case 36
    if(showEwald == 1)
    figure('Name', 'Ewald Circle for Source #36');scatter3(x36,y36,z36,5,z36,'filled');
    view(0,90);colormap(jet);colorbar;
    end
    fprintf('\nProcessing Files for Source #36 .');statuspt = nx/10;percentdone = 0;
    for ix=1:nx
        count = count+1;
        if(count >= statuspt)
            fprintf('.');count = 0;percentdone = percentdone+1;
            if(percentdone == 2)
                fprintf('25%%')
            end
            if(percentdone == 5)
                fprintf('50%%');
            end
            if(percentdone == 7)
                fprintf('.75%%')
            end
        end
        for iy=1:ny
            vsi36(ix,iy) = sum(      In36(:,3) .* exp(i.*2.0.*pi.* ...
                (In36(:,1).*(dx.*(ix-nx/2-1)+sx)+ In36(:,2).*(dy.*(iy-ny/2-1)+sy))));
            vsib36(ix,iy) = sum(      Inb36(:,3) .* exp(i.*2.0.*pi.* ...
                (Inb36(:,1).*(dx.*(ix-nx/2-1)+sx)+ Inb36(:,2).*(dy.*(iy-ny/2-1)+sy))));
        end
    end
    clear In36;fprintf('. Done\n');
    if (showBorn == 1)
    figure('Name', 'Born VSi for Source #36');
    surf(abs(vsi36));shading flat;colormap(jet); view(0,-90);colorbar
    end
```

```
for ccc = 1:c/2
   for rr = 1:r/2
       vsi36c(rr,ccc) = vsi36(rr*2,ccc*2);vsib36c(rr,ccc) = vsib36(rr*2,ccc*2);
   end;
end;
[r1,c1] = size(vsi36c);
VSI36 = fftshift(fft2(fftshift(vsi36c)));AbsVSI36 = abs(VSI36);
MaxAbsVSI36 = max(max(AbsVSI36));
RefVal36 = (MaxAbsVSI36+RefBias)*RefMult;
VSI36Causal = zeros(NewSize,NewSize); %
VSI36Causal(1:r1,NewMidPoint:NewSize) = VSI36;
VSI36Causal(RefRowPos,RefColPos) = RefVal36;
VSIB36 = fftshift(fft2(fftshift(vsib36c)));AbsVSIB36 = abs(VSIB36);
MaxAbsVSIB36 = max(max(AbsVSIB36));
RefValB36 = (MaxAbsVSIB36+RefBias)*RefMult;
VSIB36Causal = zeros(NewSize,NewSize);
VSIB36Causal(1:r1,NewMidPoint:NewSize) = VSIB36;
VSIB36Causal(RefRowPos,RefColPos) = RefValB36;
vsi36Ref = ifftshift(ifft2(ifftshift(VSI36Causal)));
vsi36Phase = atan2(imag(vsi30Ref),real(vsi36Ref));
vsib36Ref = ifftshift(ifft2(ifftshift(VSIB36Causal)));
logvsi36Ref = log(vsi36Ref); logvsib36Ref = log(vsib36Ref);
G36 = fftshift(fft2(fftshift(logvsi36Ref)));
GB36 = fftshift(fft2(fftshift(logvsib36Ref)));
 GB361 = zeros(nx,ny);GB362 = zeros(nx,ny);GB363 = zeros(nx,ny);
GB364 = zeros(nx,ny);GB365 = zeros(nx,ny);GB366 = zeros(nx,ny);
GB367 = zeros(nx,ny);GB368 = zeros(nx,ny);GB369 = zeros(nx,ny);
for xg=1:1:nx
   for yg=1:1:nx
   GB361(xg,yg) =  1.*GB36(xg,yg);GB362(xg,yg) = .2.*GB36(xg,yg);
   GB363(xg,yg) = .3.*GB36(xg,yg);GB364(xg,yg) = .4.*GB36(xg,yg);
   GB365(xg,yg) = .5.*GB36(xg,yg);GB366(xg,yg) = .6.*GB36(xg,yg);
   GB367(xg,yg) = .7.*GB36(xg,yg);GB368(xg,yg) = .8.*GB36(xg,yg);
   GB369(xg,yg) = .9.*GB36(xg,yg);
   end
end
G136 = Gaussian.*G36;  G1Tc = G1Tc + G136;
GTc1 = GTc1 + Gaussian.*(G36-GB361);
GTc2 = GTc2 + Gaussian.*(G36-GB362);
GTc3 = GTc3 + Gaussian.*(G36-GB363);
GTc4 = GTc4 + Gaussian.*(G36-GB364);
GTc5 = GTc5 + Gaussian.*(G36-GB365);
GTc6 = GTc6 + Gaussian.*(G36-GB366);
GTc7 = GTc7 + Gaussian.*(G36-GB367);
GTc8 = GTc8 + Gaussian.*(G36-GB368);
GTc9 = GTc9 + Gaussian.*(G36-GB369);
```

```
            logv36 = ifftshift(ifft2(ifftshift(G136))); v36 = exp(logv36); vTi = vTi + v36;
            if(showCep ==1)
            figure('Name', 'Reconstructed V #36');
            surf(abs(v36));shading flat;colormap(jet); view(0,-90);colorbar
            end
        otherwise
    end
end
fprintf('\n\nProcessing Files for Background  .');
            statuspt = nx/10;percentdone = 0;
            for ix=1:nx
                count = count+1;
                if(count >= statuspt)
                    fprintf('.');count = 0;percentdone = percentdone+1;
                    if(percentdone == 2)
                        fprintf('25%%')
                    end
                    if(percentdone == 5)
                        fprintf('50%%');
                    end
                    if(percentdone == 7)
                        fprintf('.75%%')
                    end
                end
                for iy=1:ny
                    vsiB(ix,iy) = sum(      InB(:,3) .* exp(i.*2.0.*pi.* ...
                        (InB(:,1).*(dx.*(ix-nx/2-1)+sx)+ InB(:,2).*(dy.*(iy-ny/2-1)+sy))));
                end
            end
            clear InB; fprintf('.. Done\n');fprintf('\nProcessing Files for All Sources .');
            statuspt = nx/10;percentdone = 0;
            for ix=1:nx
                count = count+1;
                if(count >= statuspt)
                    fprintf('.');count = 0;percentdone = percentdone+1;
                    if(percentdone == 2)
                        fprintf('25%%')
                    end
                    if(percentdone == 5)
                        fprintf('50%%');
                    end
                    if(percentdone == 7)
                        fprintf('.75%%')
                    end
                end
                for iy=1:ny
        vsiT(ix,iy) = sum(      InT(:,3) .* exp(i.*2.0.*pi.* ...
            (InT(:,1).*(dx.*(ix-nx/2-1)+sx)+ InT(:,2).*(dy.*(iy-ny/2-1)+sy))));
    end
end
fprintf('. Done\n\n');
figure('Name', 'Born of Combined Sources');
surf(abs(vsiT));shading flat;colormap(jet); view(0,-90);colorbar
for ccc = 1:c/2
    for rr = 1:r/2
        vsiTc(rr,ccc) = vsiT(rr*2,ccc*2);vsiBc(rr,ccc) = vsiB(rr*2, ccc*2);
    end;
end;
[r1,c1] = size(vsiTc);
VSIT = fftshift(fft2(fftshift(vsiTc)));VSIB = fftshift(fft2(fftshift(vsiBc)));
AbsVSIT = abs(VSIT);  MaxAbsVSIT = max(max(AbsVSIT));
RefValT = (MaxAbsVSIT+RefBias)*RefMult;
VSITCausal = zeros(NewSize,NewSize);
VSIBCausal = zeros(NewSize,NewSize);
VSITCausal(1:r1,NewMidPoint:NewSize) = VSIT;
VSIBCausal(1:r1,NewMidPoint:NewSize) = VSIB;
VSITCausal(RefRowPos,RefColPos) = RefValT;
vsiTRef = ifftshift(ifft2(ifftshift(VSITCausal)));
vsiBRef = ifftshift(ifft2(ifftshift(VSIBCausal)));
vsiTPhase = atan2(imag(vsiTRef),real(vsiTRef));
logvsiTRef = log(vsiTRef);  logvsiBRef = log(vsiBRef);
GT = fftshift(fft2(fftshift(logvsiTRef)));
GTB = fftshift(fft2(fftshift(logvsiBRef)));
GTBM1 = zeros(nx,ny);GTBM2 = zeros(nx,ny);GTBM3 = zeros(nx,ny);
GTBM4 = zeros(nx,ny);GTBM5 = zeros(nx,ny);GTBM6 = zeros(nx,ny);
GTBM7 = zeros(nx,ny);GTBM8 = zeros(nx,ny);GTBM9 = zeros(nx,ny);
for xg=1:1:nx
    for yg=1:1:nx
    GTBM1(xg,yg) = .1.*GTB(xg,yg);GTBM2(xg,yg) = .2.*GTB(xg,yg);
    GTBM3(xg,yg) = .3.*GTB(xg,yg);GTBM4(xg,yg) = .4.*GTB(xg,yg);
    GTBM5(xg,yg) = .5.*GTB(xg,yg);GTBM6(xg,yg) = .6.*GTB(xg,yg);
    GTBM7(xg,yg) = .7.*GTB(xg,yg);GTBM8(xg,yg) = .8.*GTB(xg,yg);
    GTBM9(xg,yg) = .9.*GTB(xg,yg);
    end
end
G1T0 = Gaussian.*GT;G1T1 = Gaussian.*(GT-GTBM1);
G1T2 = Gaussian.*(GT-GTBM2);G1T3 = Gaussian.*(GT-GTBM3);
G1T4 = Gaussian.*(GT-GTBM4);G1T5 = Gaussian.*(GT-GTBM5);
G1T6 = Gaussian.*(GT-GTBM6);G1T7 = Gaussian.*(GT-GTBM7);
G1T8 = Gaussian.*(GT-GTBM8);G1T9 = Gaussian.*(GT-GTBM9);
logvT0 = ifftshift(ifft2(ifftshift(G1T1)));logvT1 = ifftshift(ifft2(ifftshift(G1T1)));
logvT2 = ifftshift(ifft2(ifftshift(G1T2)));logvT3 = ifftshift(ifft2(ifftshift(G1T3)));
```

```
logvT4 = ifftshift(ifft2(ifftshift(G1T4)));logvT5 = ifftshift(ifft2(ifftshift(G1T5)));
logvT6 = ifftshift(ifft2(ifftshift(G1T6)));logvT7 = ifftshift(ifft2(ifftshift(G1T7)));
logvT8 = ifftshift(ifft2(ifftshift(G1T8)));logvT9 = ifftshift(ifft2(ifftshift(G1T9)));
vT0 = exp(logvT0);vT1 = exp(logvT1);vT2 = exp(logvT2);vT3 = exp(logvT3);
vT4 = exp(logvT4);vT5 = exp(logvT5);vT6 = exp(logvT6);vT7 = exp(logvT7);
vT8 = exp(logvT8);vT9 = exp(logvT9);
figure('Name', 'Cepstrum of Total Born');
surf(abs(vT0));shading flat;colormap(jet); view(0,-90);colorbar
figure('Name', 'Cepstrum of Total Born 1');
surf(abs(vT1));shading flat;colormap(jet); view(0,-90);colorbar
figure('Name', 'Cepstrum of Total Born 2');
surf(abs(vT2));shading flat;colormap(jet); view(0,-90);colorbar
figure('Name', 'Cepstrum of Total Born 3');
surf(abs(vT3));shading flat;colormap(jet); view(0,-90);colorbar
figure('Name', 'Cepstrum of Total Born 4');
surf(abs(vT4));shading flat;colormap(jet); view(0,-90);colorbar
figure('Name', 'Cepstrum of Total Born 5');
surf(abs(vT5));shading flat;colormap(jet); view(0,-90);colorbar
figure('Name', 'Cepstrum of Total Born 6');
surf(abs(vT6));shading flat;colormap(jet); view(0,-90);colorbar
figure('Name', 'Cepstrum of Total Born 7');
surf(abs(vT7));shading flat;colormap(jet); view(0,-90);colorbar
figure('Name', 'Cepstrum of Total Born 8');
surf(abs(vT8));shading flat;colormap(jet); view(0,-90);colorbar
figure('Name', 'Cepstrum of Total Born 9');
surf(abs(vT9));shading flat;colormap(jet); view(0,-90);colorbar
logvTc = ifftshift(ifft2(ifftshift(G1Tc))); vTc = exp(logvTc);
figure('Name', 'Combined Cepstrums');
surf(abs(vTc));shading flat;colormap(jet); view(0,-90);colorbar
figure('Name', 'Combined Images');
surf(abs(vTi*ImM));shading flat;colormap(jet); view(0,-90);colorbar
logvTc5 = ifftshift(ifft2(ifftshift(GTc5))); vTc5 = exp(logvTc5);
figure('Name', 'Combined Cepstrums - .5B');
surf(abs(vTc5));shading flat;colormap(jet); view(0,-90);colorbar
 logvTc9 = ifftshift(ifft2(ifftshift(GTc9))); vTc9 = exp(logvTc9);
figure('Name', 'Combined Cepstrums - .9B');
surf(abs(vTc9));shading flat;colormap(jet); view(0,-90);colorbar
logvTc1 = ifftshift(ifft2(ifftshift(GTc1))); vTc1 = exp(logvTc1);
figure('Name', 'Combined Cepstrums - 1B');
surf(abs(vTc1));shading flat;colormap(jet); view(0,-90);colorbar
figure('Name', 'Combined Circles');
scatter3(xt,yt,zt,5,zt,'filled'); view(0,90); colormap(jet); colorbar;
```

VITA

Richard Shane Ritter was born in Memphis, Tennessee and at age 8 moved to Clinton, Mississippi where he lived until completing high school. Shane received his AA in General Engineering from Hinds Community College in the fall of 1988 where he was named "Engineering Student of the Year" for that same year. Shane received his Bachelor of Science in electrical engineering from Mississippi State University in the fall of 1990 with concentrations in power, communications, and computers. Shane received his Master of Science in electrical engineering from Mississippi State University in the spring of 1997 with a concentration in communication theory, where he served as a graduate teaching assistant. Shane became a registered/licensed professional engineer (PE) in 1998 and is currently licensed in over 25 states. Shane became a Registered Communications Distribution Designer (BICSI RCDD) in 2003. Shane became a Leadership in Energy and Environmental Design Accredited Professional (USGBC LEED AP®) in 2004. Shane also became lighting certified (NCQLP LC) in 2005. Shane served as an adjunct faculty member in mathematics, statistics, and research for the University of Phoenix from 2001-2009. Shane also served as an adjunct faculty member in electrical and electronics engineering for the ITT Technical Institute in 2010. Currently, Shane is the Electrical Engineering Department Head for Perigon International Inc. Consulting Engineers in Matthews, NC.