Brigham Young University
BYU ScholarsArchive

Theses and Dissertations

2005-08-10

A Method for Characterizing the Properties of Industrial Foams

Shaun M. Salisbury
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Mechanical Engineering Commons

BYU ScholarsArchive Citation
Salisbury, Shaun M., "A Method for Characterizing the Properties of Industrial Foams" (2005). *Theses and Dissertations*. 699.
https://scholarsarchive.byu.edu/etd/699

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

A METHOD FOR CHARACTERIZING THE PROPERTIES OF INDUSTRIAL FOAMS

by

Shaun M. Salisbury

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

December 2005

Copyright © 2005 Shaun M. Salisbury

All Rights Reserved

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Shaun M. Salisbury

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Matthew R. Jones, Chair
Date	Brent W. Webb
Date	Dale R. Tree

BRIGHAM YOUNG UNIVERSITY

As chair of the candidates graduate committee, I have read the thesis of Shaun M. Salisbury in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Matthew R. Jones Chair, Graduate Committee

Accepted for the Department

Larry L. Howell
Department Chair

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of Engineering and Technology

ABSTRACT

A METHOD FOR CHARACTERIZING THE PROPERTIES OF INDUSTRIAL FOAMS

Shaun M. Salisbury

Department of Mechanical Engineering

Masters of Science

Assessing the effect of foam layers on transport phenomena is of significant interest in many industries, so a method for predicting foam layer properties has been developed. A model of the propagation of radiation from an amplitude-modulated laser beam through a non-absorbing foam layer has been developed using diffusion theory. Measurements predicted by diffusion theory were compared to results generated using Monte Carlo methods for a variety of foam layer properties in both the time-domain and the frequency-domain. The properties that were varied include the layer thickness, the scattering coefficient, and the asymmetry parameter. Layer thicknesses between 8.5 mm and 18 cm were considered. Values of the scattering coefficient ranged from about 600 m^{-1} to 14000 m^{-1}, while the asymmetry parameter varied between 0 and 1.

A conjugate-gradient algorithm was used to minimize the difference between simulated Monte Carlo measurements and diffusion theory predicted measurements. A large set of simulated measurements, calculated at various source-detector separations and three discrete frequencies were used to predict the layer properties. Ten blind cases were considered and property predictions were made for each. The predicted properties were within approximately 10% of the actual values, and on average the errors were approximately 4%. Predictions of μ_s' were all within approximately 5% with the majority being within 3%. Predictions of L were all within approximately 10% with the majority being within 7%. Attempts to separate g from μ_s' were unsuccessful, and it was determined that implementation of different source models might make such attempts possible.

It was shown that with a large number of measurements, properties could be accurately predicted. A method for reducing the number of measurements needed for accurate property estimation was developed. Starting with a single measurement location, property predictions were made. An approach for updating the optimal detector location, based on the current estimate of the properties, was developed and applied to three cases. Property predictions for the three cases were made to within 10% of the actual values. A maximum of three measurement locations were necessary to obtain such predictions, a significant reduction as compared to the previously illustrated method.

ACKNOWLEDGEMENTS

First of all I would like to acknowledge my wife and two children. They have been extremely supportive throughout the years and have endured much during this time. At times they may have forgotten that they had a husband and father, but they stuck with me and gave me encouragement along the way. Other family members have also been very supportive through the years. I especially need to thank my parents for helping me get to the point where I could pursue a Master's Degree. They always encouraged me to pursue an education that I would enjoy and in which I could excel. Because of such encouragement I might say that I am here today.

Secondly, I would like to thank Dr. Jones and Dr. Webb for giving me the opportunity to work with them on this project. They have consulted and encouraged me through the hard times and rejoiced with me during the good times (these were few). I must also acknowledge the assistance of the "Marylou" computer family (Marylou, Marylou2, Marylou4, and Maryloux). Without their many long hours of work, my research would have probably taken many more years. These resources are a big reason why I can even write this today.

Finally, I would like to thank all of my research colleagues both past and present, down in room 131 of the Clyde Building. At times their assistance was helpful, but their conversations were always entertaining. And so I bid farewell to all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Acronyms

Acronym	Definition
AC	Amplitude of oscillation for the reflected flux
DC	Offset around which the *AC* component oscillates
FFT	Fast Fourier Transform
MC	Monte Carlo
MPS	Modified Point Source
RTE	Radiative Transfer Equation

Roman Symbols

Symbol	Unit(s)	Definition
a	——	aspect ratio, $a = r_o / L$
A	——	amplitude of the modulation
c	m/s	speed of light in the foam layer
$\mathbf{d^k}$	——	direction of descent vector for conjugate gradient algorithm
D	——	dimensionless diffusion coefficient, $D = c / 3r_o^2 \mu_s' \omega_o$
f	MHz	variable denoting frequency
f_o	MHz	source frequency
f_1	——	portion of Γ that accounts for the sensitivities
f_1^*	——	allowable value of f_1
$\tilde{f}_1$	——	indifference value of f_1
f_2	——	portion of Γ that accounts for the modeling error
f_2^*	——	allowable value of f_2
$\tilde{f}_2$	——	indifference value of f_2
$f(\tau)$	——	time-dependent source in terms of non-dimensional variables
$\tilde{f}(\tau)$	——	time-dependent source in terms of dimensional variables
$\hat{F}$	——	Fourier Transform of time dependent source representation
$\hat{F}^*$	——	Fourier Transform of Monte Carlo simulated data
g	——	asymmetry parameter

h	——	internal reflection coefficient
H	m^{-1}	boundary coefficient
I	——	number of measurements used in the inverse problem
$I(\vec{r},\hat{s},t)$	W/m^2	diffuse intensity or radiance
$Im()$	——	imaginary component of complex number resulting from FFT
$\boldsymbol{J}$	——	sensitivity matrix
$J(\vec{r},t)$	W/m^2	radiative flux vector
k_n	——	Kernel for the Fourier Transform
L	m	layer thickness
$\boldsymbol{M}$	——	matrix of simulated measurements used in the inverse problem
N	——	number of parameters to be estimated
$\boldsymbol{P}$	——	vector of parameters to be estimated
P	W	laser beam power
$p(\lambda,\beta_n)$	——	term used for simplification of equations
r	m	radial coordinate
r_o	m	radius of the laser beam
r_d	m	source-detector separation distance
R	——	pulsed reflectance based on diffusion theory
R^*	——	pulsed reflectance from Monte Carlo simulations
$\tilde{R}$	——	modulated reflectance based on diffusion theory
$\tilde{R}^*$	——	modulated reflectance obtained through FFT of R^*
$\hat{R}$	——	Fourier Transform of arbitrary reflectance profile
$\hat{R}^*$	——	Fourier Transform of Monte Carlo simulated data
$Re()$	——	real component of complex number resulting from FFT
$\hat{s}$	——	unit direction vector
$S(\vec{r},\hat{s},t)$	W/m^3	radiative source
$S_o(\vec{r},t)$	W/m^3	isotropic source
S, T, U	——	convenience terms used to define the reflectance profile
$\bar{S},\bar{T},\bar{U}$	——	convenience terms used for reflectance in Fourier space
$\bar{\bar{S}},\bar{\bar{T}},\bar{\bar{U}}$	——	convenience terms used for reflectance in Fourier/Hankel space
t	s	temporal coordinate
t'_o	s	time at which the point source is localized
$v(\tau)$	——	arbitrary time-dependent source input to Monte Carlo method
$v^*(\tau)$	——	pulsed time-dependent source used for Monte Carlo simulations

X	——	function to be minimized in inverse algorithm
z	m	spatial coordinate normal to the foam layer
z_o'	m	depth into the layer at which the point source is localized
$Z(f)$	——	transfer function for the Monte Carlo “system”

Greek Symbols

Symbol	**Unit(s)**	**Definition**
β_n	——	eigenvalues
β^k	——	step size used in the inverse algorithm for the kth iteration
γ^k	——	conjugation coefficient for the kth iteration
Γ	——	objective function used in optimizing measurement conditions
$\varepsilon_{\mu_s'}$	——	relative error for the reduced scattering coefficient
ε_L	——	relative error for the layer thickness
ε	——	total relative error for the estimated properties
ζ	——	non-dimensional form of *z*, $\varsigma = z/L$
ς_o'	——	non-dimensional form of z_o', $\varsigma_o' = 1/\mu_s' L$
θ	radians	phase shift
λ	——	variable used in the inverse Hankel transform
μ_a	m^{-1}	absorption coefficient
μ_s	m^{-1}	scattering coefficient
μ_s'	m^{-1}	reduced scattering coefficient, $\mu_s(1-g)$
μ_t	m^{-1}	attenuation coefficient, $\mu_a + \mu_s$
π	——	constant, *3.14159*
ρ	——	non-dimensional form of *r*, $\rho = r/r_o$
τ	——	non-dimensional form of *t*, $\tau = \omega_o t$
τ_o'	——	non-dimensional form of t_o', $\tau_o' = \omega_o/\mu_s' c$
φ	W/m^2	fluence rate
φ_o	W/m^2	characteristic fluence rate
Φ	——	scattering phase function
ψ	——	non-dimensional fluence rate $\psi = \varphi/\varphi_o$
$\bar{\psi}$	——	non-dimensional fluence rate in Fourier space
$\bar{\bar{\psi}}$	——	non-dimensional fluence rate in Fourier/Hankel space
$\mathbf{\Psi}$	——	vector of predicted measurements
ω_o	radians/s	angular source frequency

Ω	sr	solid angle

Mathematical Symbols

Symbol	**Unit(s)**	**Definition**
J_o	——	Bessel's function of the first kind of zeroth order
K_o	——	modified Bessel's function of the second kind of zeroth order
K_1	——	modified Bessel's function of the second kind of zeroth order
$\delta(\ )$	——	Dirac Delta function

1 - INTRODUCTION

1.1 MOTIVATION AND DEFINITION OF PROBLEM

Many industrial processes result in the natural development of a foam layer. At times, foams are intentionally formed for a variety of industrial applications. Foams are used in applications such as enhanced oil recovery, the production of cosmetics, textile processing and the development of new insulation and construction materials [1]. Foam is used to fight fires in confined or inaccessible spaces, and use of foam minimizes the water damage that generally occurs when extinguishing a fire. Frequently, the only way to extinguish fires in flammable liquids is to use foam [1]. Due to their low density, their low thermal conductivity and their ability to efficiently fill a space, foams are excellent insulating materials. Foams are used in electric arc furnaces to shield refractory surfaces and to protect the liquid metal from the atmosphere [2].

However, in other applications, the production of foam is an undesirable byproduct of chemical reactions occurring within a liquid. In chemical reactors and food processing applications, foam acts as a barrier to heat and mass transfer. The formation of foam is particularly problematic in the glass manufacturing industry. It is estimated that the formation of even a thin layer of foam on the surface of a glass melt may reduce the radiative heat flux from the combustion space to the melt by as much as 60% or more [3].

This increased resistance to heat transfer results in higher operating temperatures, increased NO_x formation and energy consumption in glass melting furnaces [3]. Much effort has been devoted to performing detailed numerical simulations of the heat and mass transfer occurring in glass furnaces in order to increase the energy efficiency and decrease the environmental impact associated with glass manufacturing [4]. Due to the widespread existence and importance of foam, techniques for characterizing industrial foams are of interest. Methods of characterizing industrial foams are important to more accurately model the heat and mass transfer in glass furnaces and current techniques generally require carefully controlled laboratory conditions [5]. Due to the delicate nature of foams, properties of foam extracted from a process and examined in a lab are unlikely to be representative of the properties in situ. Therefore, furthering the progress of foam characterization methods depends largely on the ability to accurately measure the structural, radiative and thermophysical properties of glass foams in situ.

Knowledge of the structural, radiative and thermophysical properties of foams will lead to greater ability to model the formation and stability of foams and to model the heat and mass transfer in foams. Clarification of the physical mechanisms involved in the production of foam and transport phenomena occurring in foam will lead to the ability to mitigate their undesirable effects and enhance their desirable characteristics in numerous industrial applications.

The goal of this thesis is to develop and illustrate a method for characterizing foam layer properties. A technique for determining radiative and structural properties of industrial foams will be developed using an inverse approach. The properties to be determined include the thickness of the layer (*L*) and the reduced scattering coefficient (μ_s'). The reduced scattering coefficient is simply a term that combines the scattering property of the medium (μ_s) with an asymmetry parameter (*g*) that characterizes the angular distribution of scattering in the medium. The reduced scattering coefficient is related to these two properties by $\mu_s' = \mu_s\left(1-g\right)$. Determining the properties of the foam requires a relationship between measurable quantities and the desired layer properties. This relationship will be obtained from a model of the radiative transfer within the foam layer. The radiative model will be based on diffusion theory. A number of blind cases were considered in order to validate the method that may be used over a broad range of property values. Monte Carlo simulations were used to generate measurements for specific foam layer properties. The simulated measurements were used as inputs to the inverse algorithm. The inverse technique employed a conjugate gradient method to minimize the error between the simulated measurements and their corresponding predicted values and thereby gave estimates for the layer properties. It should be noted that the Monte Carlo generated data was provided without knowledge of the properties used to generate the data. A design of experiments approach was used to improve the layer property predictions. The optimal range of detector locations and the frequency resulting in improved property prediction will be determined. Levels of error will be assessed to define a property range over which the proposed approach is acceptable.

1.2 Spectroscopy of Turbid Media

Spectroscopy is a widely accepted method of determining properties of turbid media. Spectroscopy is defined as the study of spectra produced when matter interacts with energy, usually electromagnetic radiation. In 1859 Gustav Robert Kirchhoff and Robert Bunsen first observed that each element has specific properties with regards to the light it emits [6]. This observation paved the way for the development of various spectroscopic techniques. A variety of techniques have been introduced which differ according to the type of energy used to probe the material (x-rays, visible light, infrared radiation, ultrasound, heat, etc.), or the nature of the physical interaction (emission, absorption, or scattering). Imaging techniques are based on similar principles, but imaging techniques and spectroscopic techniques are distinguished by their objectives. Imaging techniques are used to create an image of a substance by localizing property variations, and spectroscopic techniques are used to determine the bulk or average properties of the substance. Spectroscopy is a type of parameter estimation problem, which is a subset of the broad area of inverse problems [7, 8].

Spectroscopic methods can typically be categorized into three categories; spatially-resolved (also referred to as steady-state), time-domain, and frequency-domain. Spatially resolved methods use a continuous source and spatially-resolved measurements to infer the properties of the medium. Time-resolved methods use a pulsed laser source, and time-resolved measurements made at a limited number of detector locations to predict the layer properties. Frequency-domain methods use an amplitude-modulated source, and characteristics of the reflected flux to estimate the properties of the layer. A typical setup for a frequency domain experiment is illustrated in Fig 1-1.

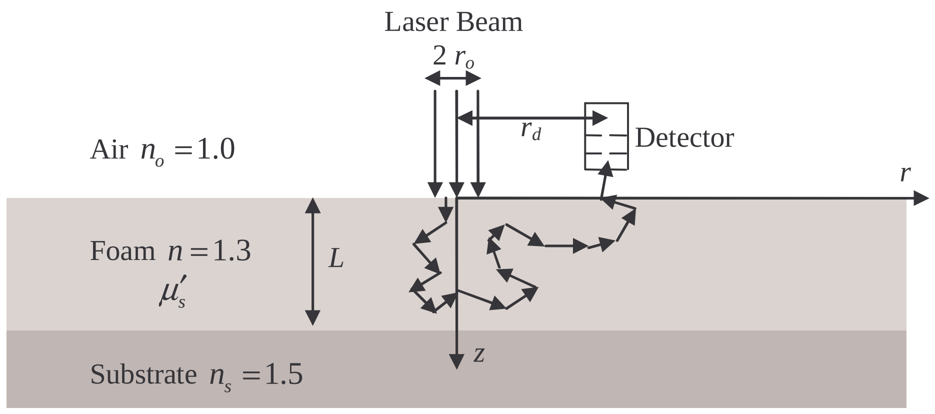
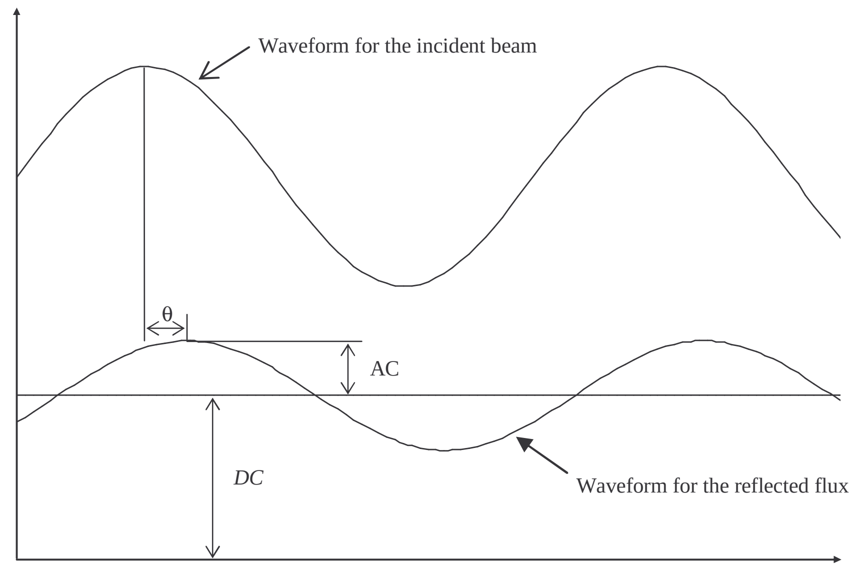


Figure 1-1: Illustrative schematic of experimental setup.

A Gaussian distributed laser beam is amplitude modulated at an angular frequency of ω_o. The beam is incident on a semi-infinite foam layer with a thickness of L. A detector located a distance r_d from the center of the beam is used to measure the time-dependent reflected radiative flux. It is envisioned that the detector can be moved freely along the surface and that measurements can be obtained at an arbitrary number of locations. Characteristics of the reflected flux such as the *DC*-offset, the *AC* component and the phase shift (θ) relative to the incident beam are related via a radiative transfer model to the desired properties of the medium. Diffusion theory is often used to model the radiative transfer in frequency domain methods [9-12]. The measurable characteristics of the reflected flux are illustrated graphically in Fig 1-2.



Waveform for the incident beam
θ
AC
DC
Waveform for the reflected flux

Figure 1-2: Illustration of the possible measurements.

Analytical expressions for the measurable quantities are obtained from a diffusion model. The diffusion approximation is obtained from simplifying approximations to the governing transport equation, and details of the derivation of the diffusion approximation equation will be given in Chapter 3.

A key component in the diffusion approximation is a model of the source term. Various source terms have been studied [13], and it will be shown in Chapter 4 that proper selection of the source term results in good agreement with Monte Carlo simulations for conditions of interest.

2 - LITERATURE REVIEW

2.1 PREVIOUS WORK

Numerous investigators have proposed spectroscopic techniques for characterizing thick turbid media. These investigations have focused primarily on determining the reduced scattering coefficient and the absorption coefficient of biological tissues. The proposed techniques fall into one of three categories depending on whether the source is continuous, pulsed or modulated. A continuous source is used in spatially resolved methods, and measurements of the reflected flux at multiple locations are used to determine the radiative properties [14-16]. Although spatially resolved methods are simple to implement, they have not proven to be as successful as the time-dependent methods [17]. Kienle and Patterson [16] employed the use of spatially resolved methods to predict the scattering coefficient and absorption coefficient to within 10% and 15% respectively. A more recent attempt at using spatially resolved methods was performed by Hayakawa et al. [18]. Implementation of what is referred to as the δ-P_1 approximation in the derivation of the diffusion approximation allowed for good results over a broader range of tissue properties. Use of the δ-P_1 approximation differs from the standard diffusion approximation in that it includes an additional term, in the form of a δ function, to both the phase function and the radiance approximations. This approach yielded predictions of the absorption coefficient (μ_a) and reduced scattering coefficient to within 16% and 23% respectively over a range of (μ_s'/μ_a) spanning nearly 3 orders of magnitude. The improved model also showed the capabilities of successfully separating the asymmetry parameter (*g*) from the reduced scattering coefficient to within 11% over a smaller range of (μ_s'/μ_a).

Time domain methods use a pulsed source, and time-resolved measurements of the attenuation and the temporal broadening of the pulse at a single location are used to determine the radiative properties [16,19,20]. This approach provides the most information, but requires expensive laboratory equipment (ultra-fast laser and streak camera). Patterson et al. [19] showed the potential of this method. Measurements were simulated using Monte Carlo generated data and these data were fitted to the calculated measurements as predicted by diffusion theory. The model predictions proved to be in good agreement with the results of the Monte Carlo simulations. An additional contribution of their research was the investigation of the how measurements of the diffusely reflected light are affected by a finite tissue layer. They concluded that the finite tissue geometry approximation can have a significant effect on the signal if the observation time is long enough. Therefore, caution should be exercised when assuming that the signal is independent of the layer thickness. Madsen et al. [20] also illustrated the capabilities of a time-domain approach. The purpose of their study was to assess the accuracy of tissue property estimates that were made by fitting diffusion theory for the pulse shape to experimental observations. Consideration was also given to the sensitivity of the accuracy of the predictions to the predicted tissue properties. When considering semi-infinite media, the predictions of the absorption coefficient and reduced scattering coefficient were within 10% or less. It was also determined that large errors were incurred for finite slabs less than 50 mm. It is therefore necessary to account for the loss of energy at the boundaries for geometries of decreasing depths.

Significant attention has been given to the use of frequency-domain techniques. In frequency domain methods, the reflected flux due to an amplitude modulated laser beam is measured. Characteristics of the reflected flux are measured at different source-detector separations or at different source frequencies and the radiative properties of the medium can be deduced from these measurements [17]. In early frequency domain studies the use of phase shift and modulation measurements were utilized to make predictions of the tissue properties [9, 19]. In a more recent publication, Fishkin et al. [12] illustrated how the use of phase shift in conjunction with either the *AC* or *DC* component of the reflectance resulted in improved predictions as compared to using the modulation with the phase shift. They also concluded that the use of multiple frequencies in the inverse algorithm offered little to no improvement in the property estimation. Alexandrakis et al. [21] considered the potential of determining the optical properties of a two-layer turbid medium as well as the thickness of the top layer by using diffusion theory. Frequency domain Monte Carlo data were fitted to the model. Their results showed the ability to predict the properties of the bottom layer to within 10% however; the results for the top layer were less accurate. They also studied the effects of different frequencies, and concluded that the diffusion approximation becomes less accurate as frequency is increased. In a more recent study, some of the same authors proposed a modified approach to the previously mentioned ideas [22]. A hybrid Monte Carlo diffusion model was used to predict the properties of a two-layer medium. Spatially resolved frequency-domain measurements were fitted to the hybrid model and it was found that the model could predict μ_s' and μ_a for both layers as well as the top layer thickness to within 5%. It is important to note that in their study, the two layers were considered to have the same value for the refractive index.

The development of the different methods of turbid media spectroscopy over time has progressed in the direction of frequency-domain studies. They have proven to be effective in property estimation and in addition to requiring less expensive laboratory equipment, frequency domain methods are more tolerant of noise [23,24]. Due to these advantages, frequency domain spectroscopic methods will be the technique employed in this research.

2.2 Research Overview

A vast majority of previous research has focused primarily on predicting the absorption and reduced scattering coefficients using diffusion theory for semi-infinite media. Attempts have been made to predict the layer thickness of two-layer turbid media, with acceptable results. However, for this research more representative boundary conditions for the diffusion model will be used to account for the refractive index mismatch at both surface interfaces. While some of the previous research was interested in properties of both layers, the current study will only be interested in the effects of the interaction at the interfaces and not what might occur outside of those boundaries. In other words, if energy escapes the layer it is assumed that it is lost and will have no contribution to the reflected flux unless it escapes at a detector location. At this stage the effects of absorption will not be included. Future development of this technique will incorporate the effects of absorption.

Chapter 3 will focus on the development of the forward problem, including the derivation of the diffusion equation. Selection of an appropriate source term for the diffusion equation will be discussed. The solution of this equation will be obtained using integral transforms in both the time domain and frequency domain.

Chapter 4 will then demonstrate the ability of diffusion theory to model Monte Carlo simulated measurements for ten cases that cover a range of property values. Comparisons are made in both the time domain and frequency domain.

The inverse problem will be described in Chapter 5. The Monte Carlo simulated measurements for ten blind cases with properties spanning a wide range of values are used as inputs to the inverse algorithm. The sensitivities of these measurements to the targeted properties play an important role in the inverse problem. Francoeur et al. [25] concluded that investigation of the sensitivities can lead to optimal measurement conditions. Another aspect of the current research will be to perform a sensitivity study for the various cases to assist in a design of optimal experiments. The design of optimal experiments will attempt to improve the predicted layer properties by identifying improved measurement conditions such as source frequency and the range of detectors over which to make the measurements.

3 - DIFFUSION MODELING OF THE RADIATIVE TRANSFER IN A NON-ABSORBING FOAM LAYER

3.1 DIFFUSION APPROXIMATION

The propagation of radiation is typically modeled according to transport theory, which is based on the principle of conservation of energy. The fundamental equation in transport theory is the radiative transfer equation (RTE). This section outlines the key points of the derivation of the diffusion approximation and likewise identifies conditions where diffusion theory breaks down.

The RTE represents an energy balance along a line of sight, and may be written in the following form.

$$\left\{\frac{1}{c}\frac{\partial}{\partial t}+\hat{s}\cdot\nabla+\mu_t\right\}I\left(\vec{r},\hat{s},t\right)=S\left(\vec{r},\hat{s},t\right)+\mu_s\int_{4\pi}\Phi\left(\hat{s}\cdot\hat{s}'\right)I\left(\vec{r},\hat{s}',t\right)d\Omega' \tag{3.1}$$

In Eq. (3.1), $I\left(\vec{r},\hat{s},t\right)$ represents the scattered or diffuse intensity, and it is frequently referred to as the radiance [26]. For the problem of interest (see Fig 1-1), emission is negligible, so the source term, $S\left(\vec{r},\hat{s},t\right)$ represents the rate at which radiative energy is scattered from the collimated beam into direction $\hat{s}$ at the specified location and time.

The diffusion approximation is obtained by assuming that the diffuse intensity varies linearly with the cosine of the scattering direction.

$$I\left(\vec{r},\hat{s},t\right)=\frac{\varphi\left(\vec{r},t\right)}{4\pi}+\hat{s}\cdot\frac{3}{4\pi}\vec{J}\left(\vec{r},t\right) \tag{3.2}$$

The radiant energy fluence rate $\varphi\left(\vec{r},t\right)$ and the radiant energy flux vector, $\vec{J}\left(\vec{r},t\right)$, are defined as $\varphi\left(\vec{r},t\right)=\int_{4\pi}I\left(\vec{r},\hat{s},t\right)d\Omega$ and $\vec{J}\left(\vec{r},t\right)=\int_{4\pi}\hat{s}I\left(\vec{r},\hat{s},t\right)d\Omega$. Substituting Eq. (3.2) into Eq. (3.1) results in the following equation.

$$\left\{\frac{1}{c}\frac{\partial}{\partial t}+\hat{s}\cdot\nabla+\mu_t\right\}\left(\frac{\varphi}{4\pi}+\hat{s}\cdot\frac{3}{4\pi}\vec{J}\right)=S+\mu_s\int_{4\pi}\Phi\left(\hat{s}\cdot\hat{s}'\right)\left(\frac{\varphi}{4\pi}+\hat{s}\cdot\frac{3}{4\pi}\vec{J}\right)d\Omega' \tag{3.3}$$

Integrating each term in Eq. (3.3) over all directions yields

$$\left\{\frac{1}{c}\frac{\partial}{\partial t}+\mu_s'+\mu_a\right\}\varphi\left(\vec{r},t\right)+\nabla\cdot\vec{J}\left(\vec{r},t\right)=S_o\left(\vec{r},t\right) \tag{3.4}$$

where the isotropic source, $S_o\left(\vec{r},t\right)$ is simply the source term integrated over all directions. Equation (3.3) is then multiplied by $\hat{s}$ and each term is integrated over all directions resulting in Eq. (3.5).

$$\left\{\frac{1}{c}\frac{\partial}{\partial t}+\mu_s'+\mu_a\right\}\vec{J}\left(\vec{r},t\right)=-\frac{1}{3}\nabla\varphi\left(\vec{r},t\right) \tag{3.5}$$

In many cases, the temporal variation in the radiant energy flux vector is small compared to the other terms on the left hand side of Eq. (3.5) [27].

$$\frac{1}{\left|\vec{J}(\vec{r},t)\right|}\frac{\partial\left|\vec{J}(\vec{r},t)\right|}{\partial t} << (\mu_s' + \mu_a)c \tag{3.6}$$

Under such conditions, Eq. (3.5) can be simplified further, resulting in Eq. (3.7).

$$\vec{J}(\vec{r},t) = -\frac{\nabla\varphi(\vec{r},t)}{3(\mu_s' + \mu_a)} \tag{3.7}$$

Substitution of Eq. (3.7) into Eq. (3.4) results in the standard governing equation for diffusion processes including the effects of absorption.

$$\frac{1}{c}\frac{\partial\varphi}{\partial t} = \frac{1}{3\mu_t'}\nabla^2\varphi - \mu_a\varphi + S_o(\vec{r},t) \tag{3.8}$$

Generally speaking, the approximations used in the derivation of Eq. (3.8) are satisfied at locations far from the source in media which are strongly scattering and weakly absorbing. Chapter 4 will demonstrate the ability of diffusion theory to model the radiative transport of non-absorbing foam layers over a broad range of μ_s' and varying values of L. It is expected that the ability of diffusion theory to accurately predict radiative behavior will decrease with decreasing μ_s' and such modeling discrepancies will lead to less accurate predictions of foam layer properties.

3.1.1 Source Model

A key component in Eq. (3.8) is the source term representation. The ability of diffusion theory to accurately represent foam behavior is largely dependent upon the source term. Jones et al. [13] considered a number of different source term representations and investigated their ability to accurately model simulated foam behavior. They showed that use of a modified point source (MPS) resulted in good agreement over a broad range of μ_s'. The modified point source representation is simply a single isotropic source located at a specified distance into the medium, z_o' and is given by Eq. (3.9),

$$S_o\left(r,z,t\right)=P\frac{\delta\left(r\right)}{2\pi r}\delta\left(z-z_o'\right)\tilde{f}\left(t\right) \tag{3.9}$$

where P is the nominal of the laser and $\tilde{f}\left(t\right)$ represents the dependence of the source on time. Both pulsed and sinusoidal time profiles are considered. The use of a pulsed input will allow for comparison of diffusion theory to simulated measurements in the time domain, and the use of a sinusoidal input will result in the ability to compare in the frequency domain.

3.2 Solution Using Integral Transforms

For the current problem, the foam is assumed to be non-absorbing at the wavelength of the laser. In addition, the problem is assumed to be azimuthally symmetric. Given these approximations, the combination of Eq. (3.8) and Eq. (3.9) simplifies to

$$\frac{1}{c}\frac{\partial\varphi}{\partial t}=\frac{1}{3\mu_s'}\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\varphi}{\partial r}\right)+\frac{\partial^2\varphi}{\partial z^2}\right]+P\frac{\delta(r)}{2\pi r}\delta\left(z-z_o'\right)\tilde{f}\left(t\right) \tag{3.10}$$

Although the representation for $\tilde{f}\left(t\right)$ is different for the pulsed input and sinusoidal input, the procedure for arriving at their corresponding solutions is identical for the majority of the process. Thus, the time dependent portion of the source term will be left arbitrary until the solution requires that it be specified

It is convenient to non-dimensionalize the governing equation using the definitions given in the nomenclature section. Applying this notation to Eq. (3.10) results in the following equation.

$$\frac{\partial\psi}{\partial\tau}=D\left[\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial\psi}{\partial\rho}\right)+a^2\frac{\partial^2\psi}{\partial\varsigma^2}\right]+\frac{\delta(\rho)}{2\rho}\delta\left(\varsigma-\varsigma_o'\right)f\left(\tau\right) \tag{3.11}$$

$f\left(\tau\right)$ is the non-dimensional representation of $\tilde{f}\left(t\right)$. It is necessary to define appropriate initial and boundary conditions using the non-dimensional notation. Prior to the energy source entering the layer the fluence rate is zero, so the initial condition is given by

$$\psi\left(\rho,\varsigma,0\right)=0 \tag{3.12}$$

The fluence rate also vanishes at locations far from the source, so the following conditions are imposed as $r \to \infty$.

$$\rho\psi\left(\rho \to \infty, \varsigma, \tau\right) = 0 \text{ and } \rho \left.\frac{\partial \psi}{\partial \rho}\right|_{\rho \to \infty} = 0 \tag{3.13}$$

The fluence rate at $\rho = 0$ must be finite, and the fluence rate is symmetric about the *z*-axis. Thus the following conditions are applied at $\rho = 0$.

$$\rho\psi\left(0, \varsigma, \tau\right) = 0 \text{ and } \rho \left.\frac{\partial \psi}{\partial \rho}\right|_{\rho = 0} = 0 \tag{3.14}$$

The boundary conditions for the *z*-direction are more complicated because of the refractive index mismatch at both interfaces. Previous researchers [17,22] have not accounted for this mismatch when attempting to predict layer thicknesses. Allowing for this mismatch leads to the following Robin boundary conditions in the *z*-direction.

$$\left.\frac{\partial \psi}{\partial \varsigma}\right|_{\varsigma = 0} = H_o L \psi\left(\rho, 0, \tau\right) \tag{3.15}$$

$$\left.\frac{\partial \psi}{\partial \varsigma}\right|_{\varsigma = 1} = -H_L L \psi\left(\rho, 1, \tau\right) \tag{3.16}$$

The boundary coefficient is given by $H = 3\mu_s'(1-\Re)/2(1+\Re)$, where the effective reflection coefficients $\Re$ are obtained using the empirical correlation developed by Egan and Hilgeman [28].

A method of integral transforms is used to solve Eq. (3.11) subject to Eqs. (3.12) - (3.16). A series of transforms is used to remove the spatial dependences from the second order partial differential equation. The result of these transforms is a first order ordinary differential equation in time which can be solved using an integrating factor. The inverse transforms are then applied to return to the original space. Figure 3-1 shows the progression of the integral transform method for arriving at a solution to the governing equations. Full details of the transforms will not be given in the body of the text, but rather can be found in Appendix A. Only the main points of the derivation are presented here in the text.

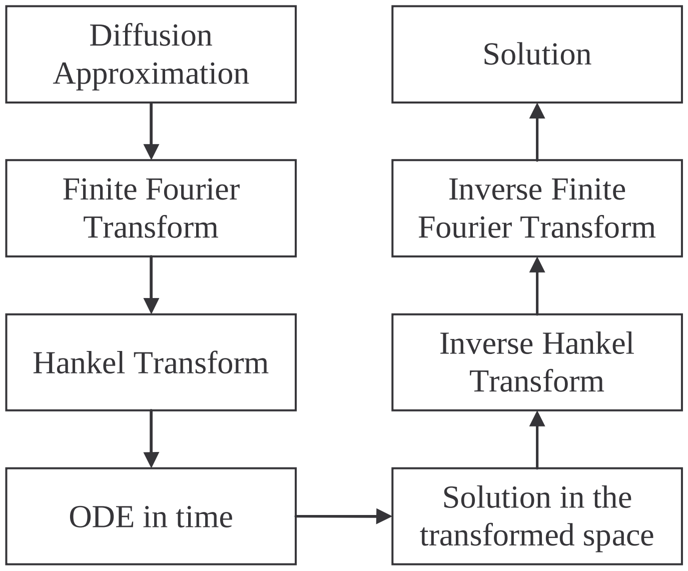


Figure 3-1: Flow chart outlining the solution procedure.

The axial portion of the problem is removed using a finite Fourier transform that is constructed on the basis of the corresponding Sturm-Liouville problem. Based on the defined conditions, the appropriate finite Fourier transform and corresponding inverse finite Fourier transform are given in Eqs. (3.17) and (3.18) respectively.

$$\bar{\psi}(\rho,\beta_n,\tau)=\int_0^1 k_n(\varsigma)\psi(\rho,\varsigma,\tau)\,d\varsigma \tag{3.17}$$

$$\psi(\rho,\varsigma,\tau)=\sum_{n=1}^{\infty} k_n(\varsigma)\bar{\psi}(\rho,\beta_n,\tau) \tag{3.18}$$

The kernel of transform $k_n(\varsigma)$ is obtained from consideration of the corresponding Sturm-Liouville problem in ς.

$$k_n(\varsigma)=\sqrt{2}\frac{\beta_n\cos(\beta_n\varsigma)+H_oL\sin(\beta_n\varsigma)}{\left(\beta_n^2+H_o^2L^2\right)\left(1+\frac{H_LL}{\beta_n^2+H_L^2L^2}\right)+H_oL} \tag{3.19}$$

Applying Eq. (3.17) to Eq. (3.11) results in the following transformed equation.

$$\frac{\partial\bar{\psi}}{\partial\tau}=D\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial\bar{\psi}}{\partial\rho}\right)-Da^2\beta_n^2\bar{\psi}+\frac{\delta(\rho)}{2\rho}k_n(\varsigma_o')\,f(\tau) \tag{3.20}$$

The radial portion of the problem is treated using a Hankel transform. The appropriate Hankel transform and corresponding inverse Hankel transform are given in Eqs. (3.21) and (3.22) respectively.

$$\bar{\bar{\psi}}(\lambda,\beta_n,\tau)=\int_0^\infty \rho J_o(\lambda\rho)\bar{\psi}(\rho,\beta_n,\tau)\,d\rho \tag{3.21}$$

$$\bar{\psi}(\rho,\beta_n,\tau)=\int_0^\infty \lambda J_o(\lambda\rho)\bar{\bar{\psi}}(\lambda,\beta_n,\tau)\,d\lambda \tag{3.22}$$

Application of the Hankel transform eliminates the radial dependence, and reduces the problem to an ordinary differential equation in τ.

$$\frac{\partial\bar{\bar{\psi}}}{\partial\tau}+D\left(\lambda^2+a^2\beta_n^2\right)\bar{\bar{\psi}}=\frac{k_n(\varsigma_o')}{2}f(\tau) \tag{3.23}$$

Solution of Eq. (3.23) requires a definition of the time dependent portion of the source. Therefore, at this point each input type will be considered separately and solutions for each will be given.

3.2.1 Time Domain

For a pulsed input, the time dependent portion of the source term is represented by the following equation.

$$f(\tau)=\delta(\tau-\tau_o') \tag{3.24}$$

Substituting Eq. (3.24) into Eq. (3.23) and solving for the transformed fluence rate yields

$$\bar{\bar{\psi}}(\lambda,\beta_n,\tau)=\frac{k_n(\varsigma_o')}{2}\exp\left\{-p(\tau-\tau_o')\right\} \tag{3.25}$$

where $p(\lambda,\beta_n)=D\left(\lambda^2+a^2\beta_n^2\right)$. Finally, the solution is obtained by application of the inverse finite Fourier transform and the inverse Hankel transform.

$$\psi(\rho,\varsigma,\tau)=\frac{1}{4D(\tau-\tau_o')}\exp\left\{-\frac{\rho^2}{4D(\tau-\tau_o')}\right\}\sum_{n=1}^{\infty}k_n(\varsigma_o')k_n(0)\exp\left\{-a^2D\beta_n^2(\tau-\tau_o')\right\} \tag{3.26}$$

The local non-dimensional reflected flux is obtained from the gradient of the fluence rate at the surface of the foam layer and is obtained using Eq. (3.15).

$$R(\rho,\tau)=\frac{1}{3\mu_s'L}\frac{\partial\psi}{\partial\varsigma}\bigg|_{\varsigma=0}=h_o\psi(\rho,0,\tau) \tag{3.27}$$

Where h_o is related to the boundary coefficient by $h_o=H_o/3\mu_s'$. A relationship for the reflected flux can then be obtained.

$$R(\rho,\tau)=\frac{h_o}{4D(\tau-\tau_o')}\exp\left\{-\frac{\rho^2}{4D(\tau-\tau_o')}\right\}\sum_{n=1}^{\infty}k_n(\varsigma_o')k_n(\varsigma_o)\exp\left\{-a^2D\beta_n^2(\tau-\tau_o')\right\} \tag{3.28}$$

In Chapter 4, Eq. (3.28) will be used to compare Monte Carlo simulated measurements to diffusion theory. Although time domain measurements will not be used in the inverse problem, Monte Carlo measurements are simulated in the time domain and therefore Eq. (3.28) serves as a good point of comparison before the simulated measurements are transformed into the frequency domain.

3.2.2 Frequency Domain

For a sinusoidal input, the time dependent portion of the source term in Eq. (3.23) is represented by the following equation.

$$f(\tau) = 1 + A\sin(\tau - \tau_o') \tag{3.29}$$

Inputting Eq. (3.29) into Eq. (3.23) and solving for the transformed variable results in the following.

$$\begin{aligned}\bar{\bar{\psi}}(\lambda, \beta_n, \tau) = {} & \frac{k_n(\varsigma_o')}{2}\left(\frac{A}{p^2+1}(\cos\tau_o' + p\sin\tau_o') - \frac{1}{p}\right)\mathrm{e}^{-p\tau} \\ & + \frac{k_n(\varsigma_o')}{2}\left[\frac{Ap}{p^2+1}\sin(\tau - \tau_o') - \frac{A}{p^2+1}\cos(\tau - \tau_o') + \frac{1}{p}\right]\end{aligned} \tag{3.30}$$

The exponential term decays rapidly for $\tau > 0$ and is therefore neglected and only the steady-periodic portion of the solution is retained.

$$\bar{\bar{\psi}}_p(\lambda, \beta_n, \tau) = \bar{\bar{S}}(\lambda, \beta_n) + \bar{\bar{T}}(\lambda, \beta_n)\sin(\tau - \tau_o') - \bar{\bar{U}}(\lambda, \beta_n)\cos(\tau - \tau_o') \tag{3.31}$$

where $\bar{\bar{S}}(\lambda, \beta_n)$, $\bar{\bar{T}}(\lambda, \beta_n)$, and $\bar{\bar{U}}(\lambda, \beta_n)$ are defined by the following equations.

$$\bar{\bar{S}}\left(\lambda,\beta_n\right)=\frac{k_n\left(\varsigma_o'\right)}{2p\left(\lambda,\beta_n\right)} \tag{3.32}$$

$$\bar{\bar{T}}\left(\lambda,\beta_n\right)=\frac{A}{2}\frac{k_n\left(\varsigma_o'\right)p\left(\lambda,\beta_n\right)}{p\left(\lambda,\beta_n\right)^2+1} \tag{3.33}$$

$$\bar{\bar{U}}\left(\lambda,\beta_n\right)=\frac{A}{2}\frac{k_n\left(\varsigma_o'\right)}{p\left(\lambda,\beta_n\right)^2+1} \tag{3.34}$$

The non-dimensional reflected flux for the frequency domain is obtained by application of the inverse finite Fourier transform and the inverse Hankel transform used in conjunction with Eq. (3.27).

$$\tilde{R}\left(\rho,\tau\right)=h_oS\left(\rho,\beta_n\right)+h_o\sqrt{T\left(\rho,\beta_n\right)^2+U\left(\rho,\beta_n\right)^2}\sin\left(\tau-\tilde{\theta}\right) \tag{3.35}$$

where the phase shift $\tilde{\theta}$ is defined by

$$\tilde{\theta}=\tau_o'+\tan^{-1}\left[\frac{U\left(\rho,0\right)}{T\left(\rho,0\right)}\right] \tag{3.36}$$

and the other terms are defined below.

$$S\left(\rho,\beta_n\right)=\sum_{n=1}^{\infty}\frac{k_n\left(\varsigma_o'\right)k_n\left(0\right)}{2}\int_0^{\infty}\frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)}d\lambda=\sum_{n=1}^{\infty}\frac{k_n\left(\varsigma_o'\right)k_n\left(0\right)}{2D}K_o\left(a\beta_n\rho\right) \tag{3.37}$$

$$T\left(\rho,\beta_n\right)=\sum_{n=1}^{\infty}A\frac{k_n\left(\varsigma_o'\right)k_n\left(0\right)}{2}\int_0^{\infty}\frac{p\left(\lambda,\beta_n\right)\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2+1}d\lambda \tag{3.38}$$

$$U\left(\rho,\beta_n\right)=\sum_{n=1}^{\infty}A\frac{k_n\left(\varsigma_o'\right)k_n\left(0\right)}{2}\int_0^{\infty}\frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2+1}d\lambda \tag{3.39}$$

Equations (3.38) and (3.39) both contain an infinite integral for which there is no closed form solution. Numerical evaluation of the integrals requires a significant amount of computation time, and this time is magnified because the integrals are nested inside of an infinite sum. However, recognizing that slight changes in the integral terms result in an expression to which there is a closed form solution leads to a more efficient way of evaluating the integrals. As λ increases, then $p\left(\lambda,\beta_n\right)^2>>1$. Therefore, at a critical value of λ_c, the second term in the denominator can be neglected. The integral portions of Eqs. (3.38) and (3.39) are then re-written in an approximate form.

$$\bar{T}\left(\rho,\beta_n\right)=\int_0^{\infty}\frac{p\left(\lambda,\beta_n\right)\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2+1}d\lambda\approx\int_0^{\lambda_c}\frac{p\left(\lambda,\beta_n\right)\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2+1}d\lambda+\int_{\lambda_c}^{\infty}\frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)}d\lambda \tag{3.40}$$

$$\bar{U}\left(\rho,\beta_n\right)=\int_0^{\infty}\frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2+1}d\lambda\approx\int_0^{\lambda_c}\frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2+1}d\lambda+\int_{\lambda_c}^{\infty}\frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2}d\lambda \tag{3.41}$$

Now consider the integral term of Eq. (3.37) and its exact solution in order to simplify Eq. (3.40).

$$\bar{S}\left(\rho,\beta_n\right)=\frac{K_o\left(a\beta_n\rho\right)}{D}=\int_0^{\lambda_c}\frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)}d\lambda+\int_{\lambda_c}^{\infty}\frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)}d\lambda \tag{3.42}$$

Solving for the last term in Eq. (3.42) and substituting for the last term in Eq. (3.40) yields the simplified result for the integral term of Eq. (3.38).

$$\bar{T}\left(\rho,\beta_n\right) \approx \int_0^{\lambda_c} \frac{p\left(\lambda,\beta_n\right)}{p\left(\lambda,\beta_n\right)^2+1} \lambda J_o\left(\lambda\rho\right) d\lambda + \frac{K_o\left(a\beta_n\rho\right)}{D} - \int_0^{\lambda_c} \frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)} d\lambda \tag{3.43}$$

Equation (3.38) can then be written in its simplified form.

$$T\left(\rho,\beta_n\right) \approx \sum_{n=1}^{\infty} A \frac{k_n\left(\varsigma_o'\right) k_n\left(0\right)}{2} \bar{T}\left(\rho,0\right) \tag{3.44}$$

The numerical integration only has to be performed over a narrow range from 0 to λ_c as opposed to the infinite nature of the previous version. To simplify Eq. (3.41) consider the following integral and its exact solution.

$$\int_0^{\infty} \frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2} d\lambda = \frac{\rho}{2\beta_n} K_1\left(a\beta_n\rho\right) = \int_0^{\lambda_c} \frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2} d\lambda + \int_{\lambda_c}^{\infty} \frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2} d\lambda \tag{3.45}$$

Solving for the last term in Eq. (3.45) and substituting for the last term in Eq. (3.41) yields the simplified result for the integral term of Eq. (3.39).

$$\bar{U}\left(\rho,\beta_n\right) \approx \int_0^{\lambda_c} \frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2+1} d\lambda + \frac{\rho}{2\beta_n} K_1\left(a\beta_n\rho\right) - \int_0^{\lambda_c} \frac{\lambda J_o\left(\lambda\rho\right)}{p\left(\lambda,\beta_n\right)^2} d\lambda \tag{3.46}$$

Equation (3.39) can then be written in its approximate form.

$$U(\rho,\beta_n) \approx \sum_{n=1}^{\infty} A \frac{k_n(\varsigma_o')k_n(0)}{2} \bar{U}(\rho,0) \tag{3.47}$$

Evaluation of Eq. (3.44) and Eq. (3.47) require significantly less computation time than Eq. (3.38) and Eq. (3.39). In order to asses the time improvement of the approximation, a test case was considered wherein *T* and *U* were calculated for both the approximated and the exact forms of their respective equations. The cutoff value for λ_c was chosen to be the point when $p(\lambda,\beta_n)^2 > 10^4$. *T* and *U* were calculated at ten detector locations for a single frequency and a single set of layer properties. Both comparisons of the values and the time to calculate those values were considered. For the exact case, the computation took approximately three days, whereas for the approximated case the computations took approximately two minutes. Thus a significant improvement is observed with regards to computation time. The values of the approximated form as compared to the exact form were all within 0.007% relative error. It can therefore be concluded that the approximation is in very good agreement and can be used in place of the exact form.

4 - COMPARISON OF DIFFUSION THEORY RESULTS WITH MONTE CARLO SIMULATIONS

4.1 MONTE CARLO SIMULATION DEVELOPMENT

Monte Carlo simulations are often used as benchmarks to which other methods of solving the RTE are compared [29]. The algorithm used for the current comparisons is based on the method outlined by Jacques and Wang [23], and simulates the random path that a photon may take as it traverses through the medium. The propagation of photons are based on probability distributions for the path length between scattering events, the deflection angles resulting from a scattering event, and the probability of reflectance or transmittance at a boundary.

As was illustrated in Fig 1-1, photons are normally incident on a foam layer. The beam is Gaussian distributed with a radius of r_o. The Monte Carlo method simulates the path that a photon takes as it undergoes multiple scattering interactions. Throughout the simulations, the reflected radiative flux from the medium is recorded, at select locations measured from the center of the source, as a function of time. Due to the randomness of the photon paths, it is necessary to launch a large number of photons to obtain clean reflectance signals.

For each of the scenarios presented in this research, the number of photons released into the medium is 10^7. Ten runs, resulting in a total of 10^8 photons, will be conducted in order to allow for some variation in the measurements. From the ten independent runs, the uncertainty for each simulated measurement can then be estimated. For the ten runs, the average and standard deviation of each measured value can be calculated. Using finite statistic theory a precision interval for each measurement is calculated [30].

$$M' = \bar{M} \pm t_{\nu,P} S_{\bar{x}} \quad (P\ \%) \tag{4.1}$$

Where M' is the true mean, $\bar{M}$ is the sample mean, $t_{\nu,P}$ is the *t*-estimator with degrees of freedom ν and probability C (%). $S_{\bar{x}}$ is the standard deviation of the means calculated by $S_{\bar{x}} = S_x / \sqrt{\mathrm{N}}$. Here N represents the sample size. The uncertainties were calculated using a 95% confidence interval and the corresponding *t*-estimator for that probability and the ten runs that were made is 2.262 [30]. The error levels presented in the next section were calculated using this approach.

The time resolution is set to 20 ps, and the resolution in the radial direction is 1 mm. The selection of an appropriate time resolution affects the reflected radiative flux simulations, as well as the Fast Fourier Transform (FFT). If the time resolution is set too large, then the location of the peak reflectance is not appropriately resolved. The time resolution also affects the range of frequency data when performing the FFT. The time resolution of 20 ps has been found to be a sufficient resolution for most scenarios of this nature [23]. Because the radial resolution has some effect on the speed of the Monte Carlo calculations, a brief study was performed to identify the maximum radial resolution that could be used without compromising the resulting signal. Initially, a small radial step was used to obtain a benchmark solution. Solutions corresponding to increased values of the radial resolution were calculated and compared to the benchmark. It was concluded that a radial resolution of 1mm could be used without affecting the benchmark beyond the noise evident in Monte Carlo simulations. A radial resolution beyond 1 mm was not considered because measurements were to be taken at 1 mm increments, and an increased value in the radial step would cause overlap between detectors in the Monte Carlo code.

The Monte Carlo simulations are performed in the time-domain and corresponding frequency-domain measurements can be calculated through the use of a FFT. Details of this transform, pertaining to the current problem, are described in Section 4.3. The following sections will show the comparisons of Monte Carlo simulated measurements to the approximate solution resulting from diffusion theory. Comparisons of ten cases, each with randomly generated properties, were performed in both the time-domain and frequency-domain. The results of three representative cases are presented in the next section, and the remaining results are presented in Appendix B. Table 4-1 summarizes the properties used to generate the Monte Carlo simulated data. It should be noted that although the property values are listed here, these values were not known until after the inverse problem was solved.

Table 4-1. Summary of the properties used for the ten blind cases that are considered in the following sections.

Case #	μ_s *(m^{-1})*	*g*	μ'_s *(m^{-1})*	*L (m)*
1	1384	0.7666	323	0.0940
2	6934	0.9254	517	0.1376
3	6276	0.4460	3477	0.0395
4	2432	0.9650	85	0.0655
5	9313	0.1154	8238	0.1794
6	7840	0.5069	3866	0.1298
7	10125	0.7302	2731	0.1838
8	13895	0.0387	13358	0.0086
9	589	0.6120	229	0.1803
10	2210	0.4638	1185	0.1529

4.2 TIME DOMAIN COMPARISON

The output of the Monte Carlo simulations results in non-dimensional reflected flux measurements as a function of time generated by a simulated pulsed input. The reflectance was recorded at detector locations ranging from 15 mm to 60 mm at uniform increments of 1 mm. However, results at a smaller number of detector locations will be presented, and these results illustrate the trends over a range of layer properties. In the figures that follow, comparisons are made between Monte Carlo simulated reflectance profiles in the time-domain and measurements predicted by diffusion theory. Three specified detector locations are considered in order to illustrate the trends in not only agreement, but also with regards to the noise. For each figure, the symbols represent the average value for the ten independent runs, and the levels of error were determined as previously described.

<u>*Case 1*</u>

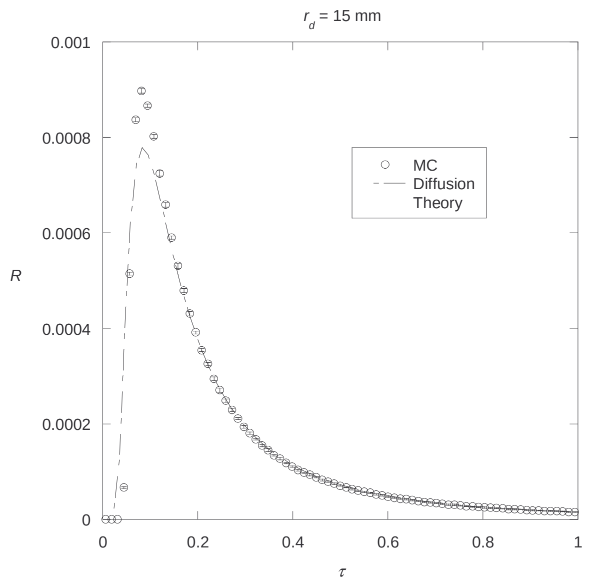


Figure 4-1: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #1 ($\mu_s' = 323$ m^{-1}, $L = 0.094$ m) at a source-detector separation of 15 mm.

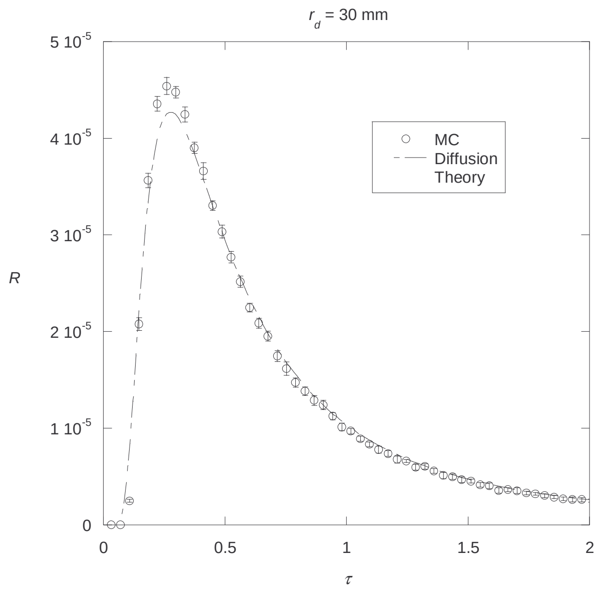


Figure 4-2: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #1 ($\mu_s' = 323$ m^{-1}, $L = 0.094$ m) at a source-detector separation of 30 mm.

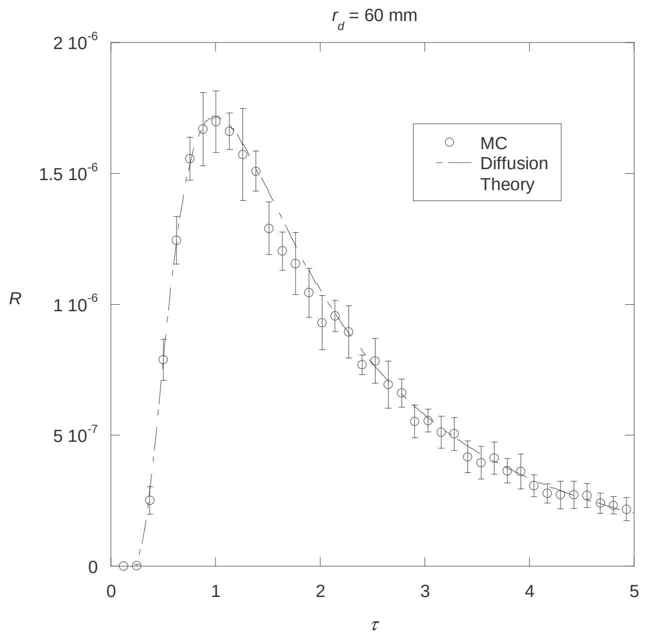


Figure 4-3: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #1 ($\mu_s' = 323$ m^{-1}, $L = 0.094$ m) at a source-detector separation of 60 mm.

Case 3

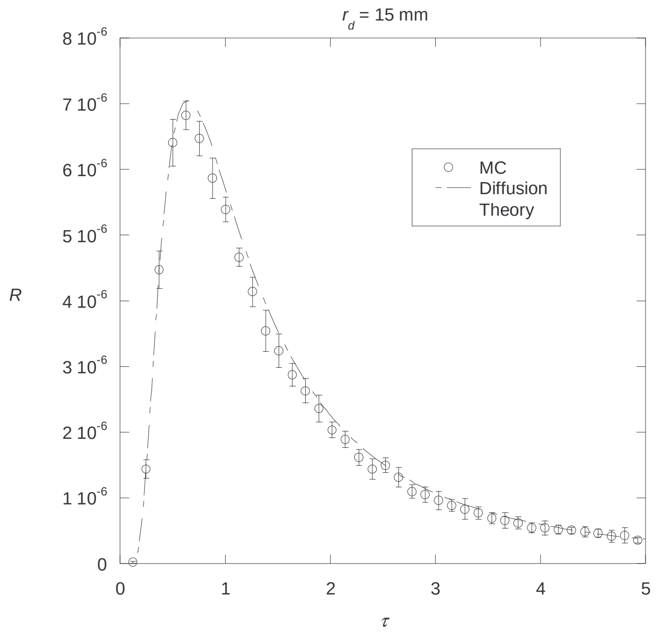


Figure 4-4: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #3 ($\mu_s' = 3477$ m^{-1}, $L = 0.0395$ m) at a source-detector separation of 15 mm.

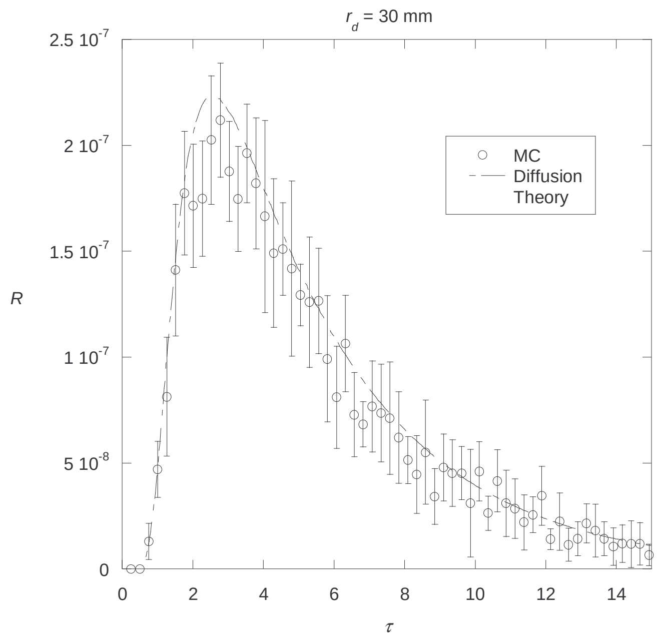


Figure 4-5: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #3 ($\mu_s' = 3477$ m^{-1}, $L = 0.0395$ m) at a source-detector separation of 30 mm.

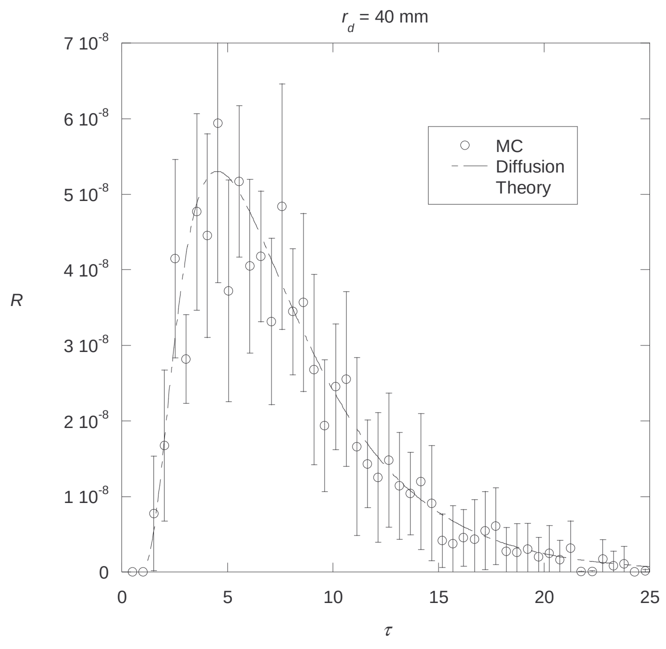


Figure 4-6: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #3 ($\mu_s' = 3477$ m^{-1}, $L = 0.0395$ m) at a source-detector separation of 40 mm.

Case 5

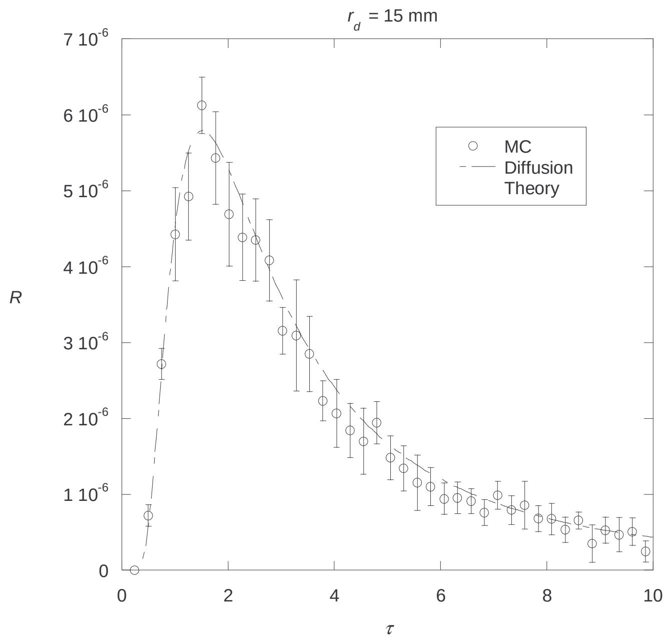


Figure 4-7: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #5 ($\mu_s' = 8238$ m^{-1}, $L = 0.1794$ m) at a source-detector separation of 15 mm.

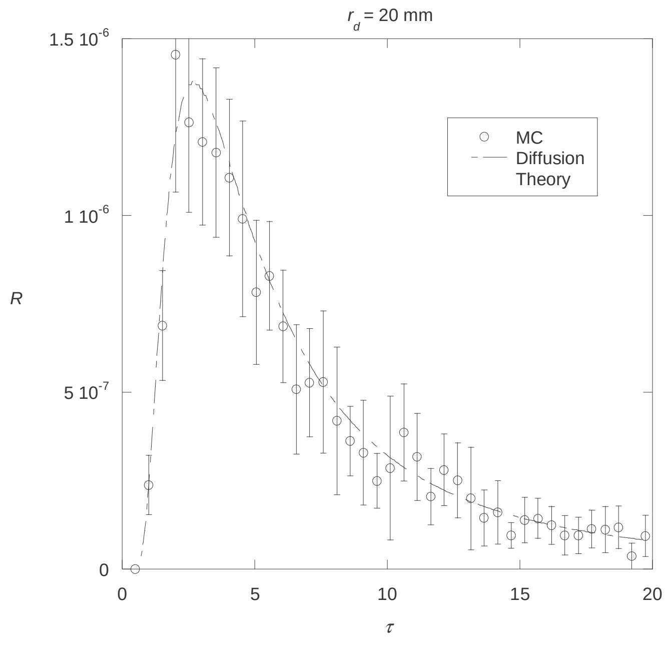


Figure 4-8: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #5 ($\mu_s' = 8238$ m^{-1}, $L = 0.1794$ m) at a source-detector separation of 20 mm.

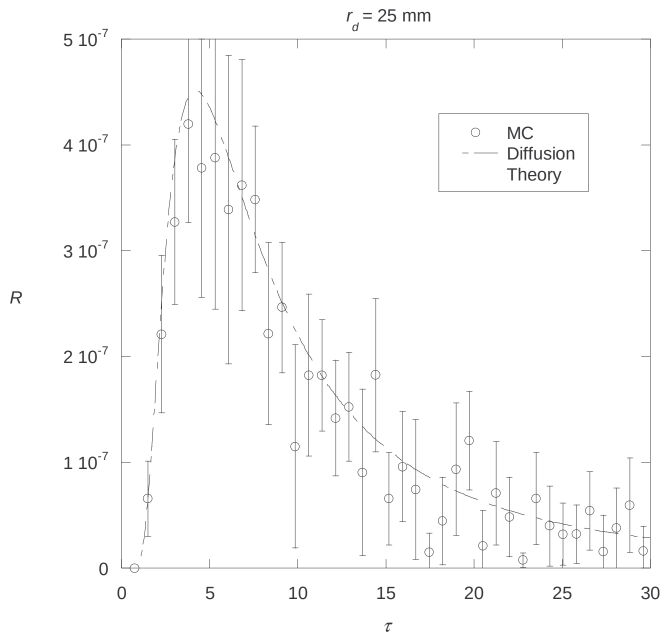


Figure 4-9: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #5 ($\mu_s' = 8238$ m^{-1}, $L = 0.1794$ m) at a source-detector separation of 25 mm.

Some common trends can be noted when considering all of the cases presented. Generally speaking, the ability of diffusion theory to accurately predict the reflectance profile increases with larger source-detector separation. This is consistent with the approximations made in the development of diffusion theory. Another observation is the consistent ability of diffusion theory to accurately model radiation propagation at later times. For some of the cases, diffusion theory was unable to predict the magnitude near the peak at detector locations nearer to the source. Although diffusion theory becomes more accurate at detector locations further from the source, the level of noise in the Monte Carlo simulations also increases. The fact that frequency-domain measurements are less susceptible to noise levels will be illustrated in the next section.

4.3 Relating Pulsed Monte Carlo Data to Modulated Monte Carlo Data Through the Use of the FFT

As was previously mentioned, simulated measurements in the frequency-domain are obtained from the time-domain simulated profile through the use of a Fast Fourier Transform (FFT). The Monte Carlo code can be treated as a system that converts the input (the time-dependent source) into the output (the time-dependent reflectance profile). Figure 4-10 shows the flow for the conversion of a time-dependent source representation into the corresponding reflectance profile. The time-dependent portion may be either pulsed or frequency-modulated, and the corresponding reflectance profile is the output.

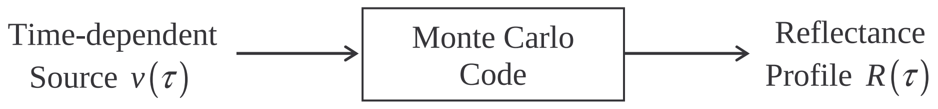

Time-dependent Source $v(\tau)$
Monte Carlo Code
Reflectance Profile $R(\tau)$

Figure 4-10: Illustration of the Monte Carlo "system".

Because the Monte Carlo data was simulated in the time-domain (corresponding to a pulsed input), use of the FFT is used in place of re-running the Monte Carlo simulations to represent a sinusoidal time-dependent input. Results for any ω_o can be produced from the same time-dependent profile. The FFT algorithm used was taken from Numerical Recipes [31] and is incorporated into the Monte Carlo code included in the Appendix D. In order to relate the reflectance profile from the time-domain to the corresponding reflectance profile in the frequency-domain, it is necessary to introduce the

Fourier transform and related notation. Full details of the derivation are not included here but can be found in Appendix C.

$$\hat{V}(f) = \int_{-\infty}^{\infty} v(\tau) e^{i\frac{f}{f_o}\tau} d\tau = F\{v(\tau)\} \tag{4.2}$$

The modulation frequency f_o is related to the angular source frequency ω_o used previously by $f_o = \omega_o / 2\pi$. The transfer function for this system is defined as

$$z(f) = \frac{\hat{R}(f)}{\hat{V}(f)} \tag{4.3}$$

where $\hat{R}(f)$ is the Fourier transform of the reflectance profile and is defined below.

$$\hat{R}(f) = F\{R(\tau)\} = \int_{-\infty}^{\infty} R(\tau) e^{i\frac{f}{f_o}\tau} d\tau \tag{4.4}$$

To find the transfer function, let $v^*(\tau) = \delta(\tau)$ be the time-dependent portion of the source and $R^*(\tau)$ be the corresponding reflectance profile. $R^*(\tau)$ is obtained from the Monte Carlo simulation with a pulse input and $\hat{R}^*(f)$ is the corresponding Fourier transform of the reflectance. However, because $R^*(\tau)$ contains data at discrete time intervals, it is convenient to calculate $\hat{R}^*(f)$ through the use of the FFT. Equation (4.2) can be used to evaluate the Fourier transform of the input $v^*(\tau)$.

$$\hat{F}^{*}(f) = \int_{-\infty}^{\infty} \delta(\tau) e^{i\frac{f}{f_O}\tau} d\tau = 1 \tag{4.5}$$

Therefore the transfer function simply becomes $z(f) = \hat{R}^{*}(f)$ and can be used for any arbitrary input $v(\tau)$ to obtain the corresponding reflectance profile. Solving Eq. (4.3) for $\hat{R}(f)$ and substituting the result of the transfer function, the Fourier transformed reflectance profile for an arbitrary source is given by

$$\hat{R}(f) = \hat{R}^{*}(f)\hat{F}(f) \tag{4.6}$$

Applying the inverse Fourier transform to Eq. (4.6) results in a reflectance profile corresponding to the respective input $v(\tau)$.

$$R(\tau) = \int_{-\infty}^{\infty} \hat{R}^{*}(f)\hat{F}(f) e^{-i\frac{f}{f_O}\tau} d\tau \tag{4.7}$$

Consider the case of a sinusoidal modulated source input $v(\tau) = 1 + A\sin\tau$.

$$\hat{F}(f) = \int_{-\infty}^{\infty} (1 + A\sin\tau) e^{i\frac{f}{f_O}\tau} d\tau \tag{4.8}$$

In order to evaluate Eq. (4.8), a definition of the exponential portion of the integral is needed [32].

$$\delta(x-\xi)=\int_{-\infty}^{\infty} e^{\pm i2\pi f(x-\xi)}df \tag{4.9}$$

Applying the definition of Eq. (4.9) and recalling the complex form of the sine function, $\sin\tau=\left(e^{i\tau}-e^{-i\tau}\right)/2i$ Eq. (4.8) can be evaluated.

$$\hat{F}(f)=\delta\left(\frac{f}{2\pi f_o}\right)-\frac{Ai}{2}\delta\left(\frac{f}{2\pi f_o}+\frac{1}{2\pi}\right)+\frac{Ai}{2}\delta\left(\frac{f}{2\pi f_o}-\frac{1}{2\pi}\right) \tag{4.10}$$

Substituting Eq. (4.10) into Eq. (4.7) and evaluating the integral yields

$$\tilde{R}(\tau)=\hat{R}^*(0)-\frac{Ai}{2}\left[\hat{R}^*(-f_o)e^{i\tau}-\hat{R}^*(f_o)e^{-i\tau}\right] \tag{4.11}$$

It is important to note that $\hat{R}^*$, obtained from the FFT, is a complex number and therefore contains both real and imaginary components as a function of frequency. At this point it is necessary to introduce some notation that will be used to simplify Eq. (4.11).

$$\hat{R}^*(f_o)=\mathrm{Re}(f_o)+i\,\mathrm{Im}(f_o) \tag{4.12}$$

Re() and Im() are used to simply represent the real and imaginary components respectively of the complex number at the specified frequencies. The following properties of the FFT allow for simplification of the previous equations [32].

$$\begin{aligned}&\mathrm{Re}(-f_o)=\mathrm{Re}(f_o)\\&\mathrm{Im}(-f_o)=-\mathrm{Im}(f_o)\\&\mathrm{Im}(0)=0\end{aligned}\tag{4.13}$$

Substituting Eqs. (4.12) and (4.13) into Eq. (4.11) and performing some simplifying algebra yields the following reflectance profile corresponding to a sinusoidal input.

$$\tilde{R}(\tau)=\mathrm{Re}(0)+A\sqrt{\mathrm{Re}(f_o)^2+\mathrm{Im}(f_o)^2}\,\sin(\tau-\theta)\tag{4.14}$$

Where θ represents the phase shift between the input and the output and is defined by the following equation.

$$\theta=\tan^{-1}\left[\frac{\mathrm{Im}(f_o)}{\mathrm{Re}(f_o)}\right]\tag{4.15}$$

Equation (4.14) takes the same form as Eq. (3.35) and therefore makes for easy frequency-domain comparisons between diffusion theory predictions and transformed Monte Carlo simulations.

4.4 FREQUENCY DOMAIN COMPARISON

Comparisons between diffusion theory and simulated data in the frequency-domain can be made two different ways; the overall sinusoidal reflectance and the individual characteristics of the reflected flux. Both were previously illustrated in Fig 1-2.

The sinusoidal profiles are presented for the same three cases that were shown in the previous section at a single detector location and a single modulation frequency plotted as a function of time. The sinusoidal profiles for the other seven cases can be found in the Appendix B. Equation (3.35) and Eq. (4.14) are used to generate the modulated reflectance curves for diffusion theory and the simulated measurements respectively. The characteristics of the reflected flux (*DC*, *AC*, θ) are obtained from these same equations, where the general form is given by the following equation.

$$\tilde{R}(\tau) = DC + AC\sin(\tau - \theta) \tag{4.16}$$

For each case, the *AC* component and phase shift of the sinusoidal profile are also plotted as a function of the detector location for two frequencies. The *DC* offset is also plotted as a function of the detector location. However, it is only plotted at a single frequency because the *DC* offset is simply scaled between different frequencies. The *AC* component, and phase shift for the other seven cases can be found in the Appendix B. Although a variety of detector locations and modulation frequencies could be considered for each case, only a select few are presented here to illustrate the accuracy trends associated with diffusion theory modeling. It should be noted that although measured phase shift values can only vary between 0 and 2π, at times a factor of 2π was added to the simulated phase shift values in order to maintain a continuous curve.

Case1

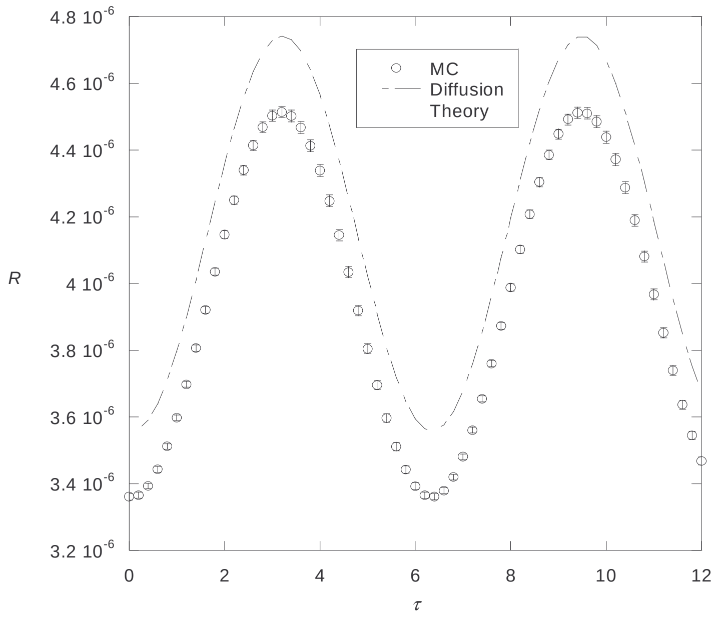


Figure 4-11: Frequency-domain comparison between diffusion theory and Monte Carlo simulations for case #1 ($\mu_s' = 323$ m^{-1}, $L = 0.094$ m) at $r_d = 60$ mm and $f_o = 100$ MHz.

As compared to the time-domain plot for the same case (see Fig 4-3), the level of noise in the measurements in Fig 4-11 is significantly reduced in the frequency-domain. This illustrates the claim made earlier that frequency-domain techniques are less susceptible to the noise levels than are those of the time-domain. The trends of the sinusoidal curves at other frequencies are very similar and therefore results at additional frequencies offer no further useful information. Figure 4-11 shows good agreement between diffusion theory and the simulated measurements. The shapes are very similar; however, the *DC* offset appears to be slightly off. The plot is somewhat misleading, and the *DC* offset agreement is verified in Fig 4-12. Consideration of each individual characteristic shows the detailed agreement between diffusion theory and Monte Carlo simulations.

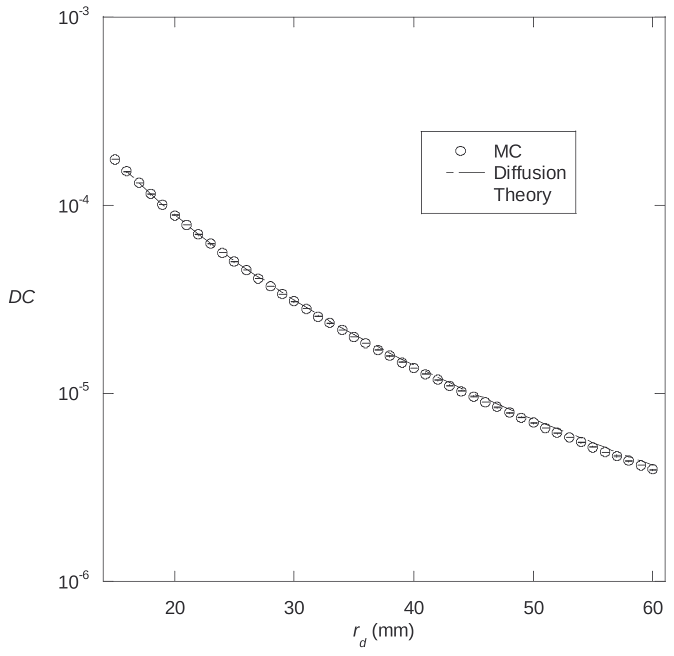
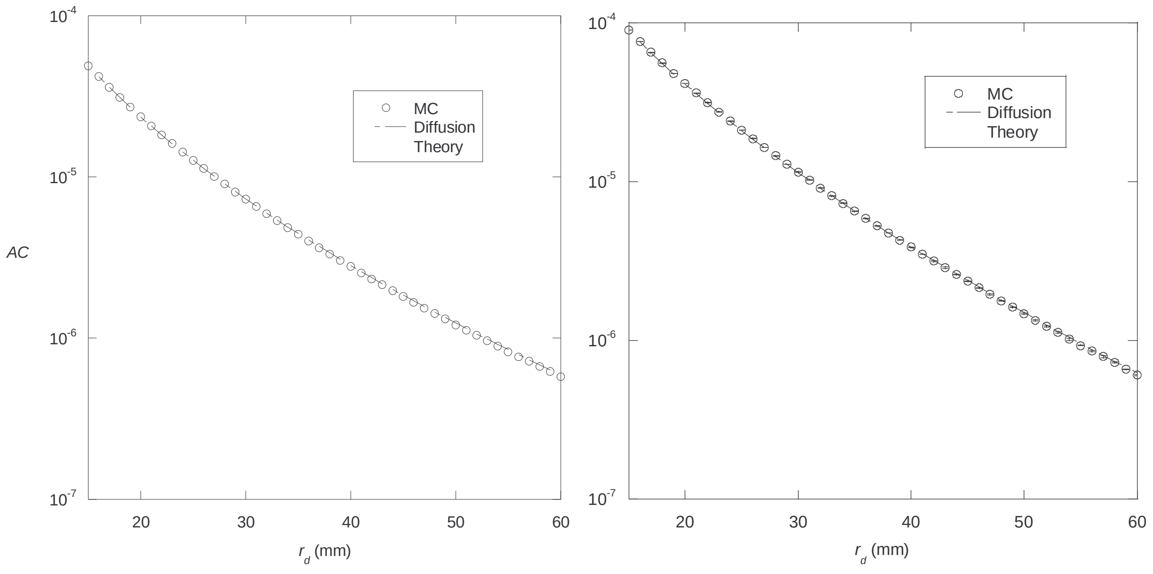
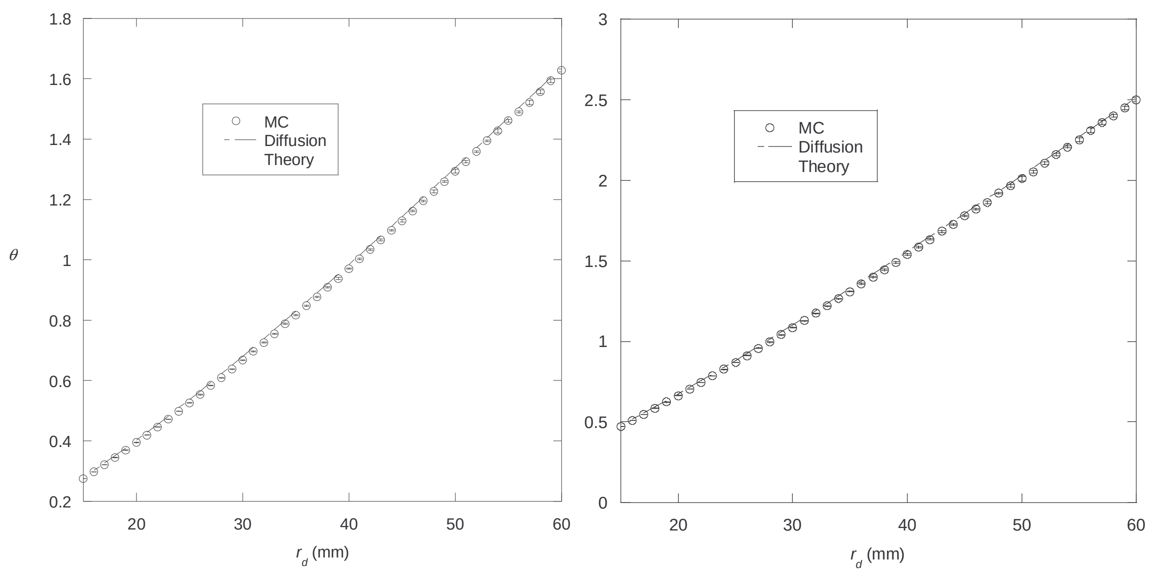


Figure 4-12: Comparison between diffusion theory and Monte Carlo simulations of *DC* values for case #1 ($\mu_s' = 323$ m^{-1}, $L = 0.094$ m) at $f_o = 100$ MHz.



(a) **(b)**

Figure 4-13: Comparison between diffusion theory and Monte Carlo simulations of *AC* values for case #1 ($\mu_s' = 323$ m^{-1}, $L = 0.094$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.



(a) **(b)**

Figure 4-14: Comparison of the phase shift between diffusion theory and Monte Carlo simulations for case #1 ($\mu_s' = 323$ m^{-1}, $L = 0.094$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

The results for case 1 show excellent agreement and relatively small noise levels over the full range of detector locations. The difference in noise levels between time-domain and frequency-domain measurements are again illustrated. Similar results are seen for the other nine cases with increased noise for the higher scattering cases. Based on the noise levels for each case, some discretion was used to select an upper limit of the detector location for each case. This upper limit was selected when either the noise level was on the same order of magnitude as the measurement, or if the maximum detector location was reached. In some cases the reflectance signal was non-existent at large source-detector separations; for these reasons not all cases have the same upper detector limit. The ranges presented in these comparisons are ultimately the ranges that are used in the inverse problem. The remainder of the cases is presented in succession and some general trends are noted at the end.

Case 3

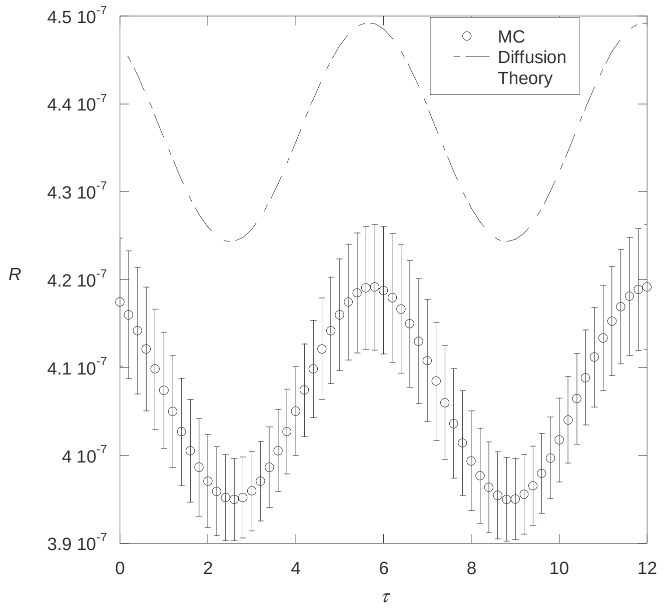


Figure 4-15: Frequency-domain comparison between diffusion theory and Monte Carlo simulations for case #3 ($\mu_s' = 3477$ m^{-1}, $L = 0.0395$ m) at $r_d = 40$ mm and $f_o = 100$ MHz.

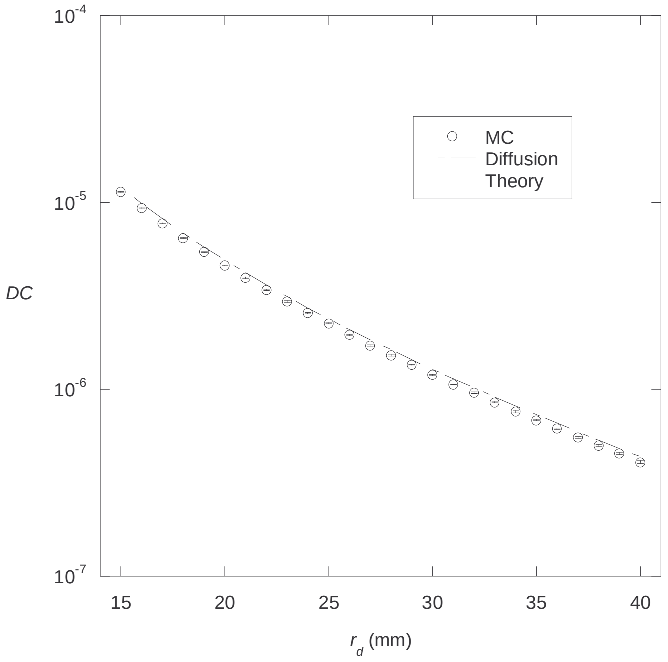
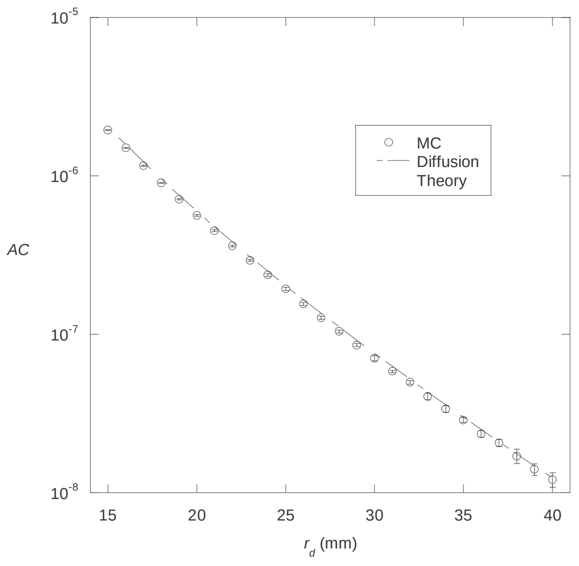
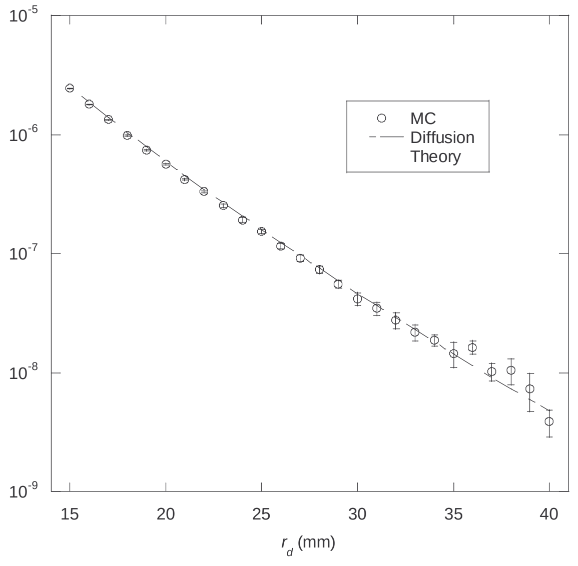


Figure 4-16: Comparison between diffusion theory and Monte Carlo simulations of *DC* values for case #3 ($\mu_s' = 3477$ m^{-1}, $L = 0.0395$ m) at $f_o = 100$ MHz.

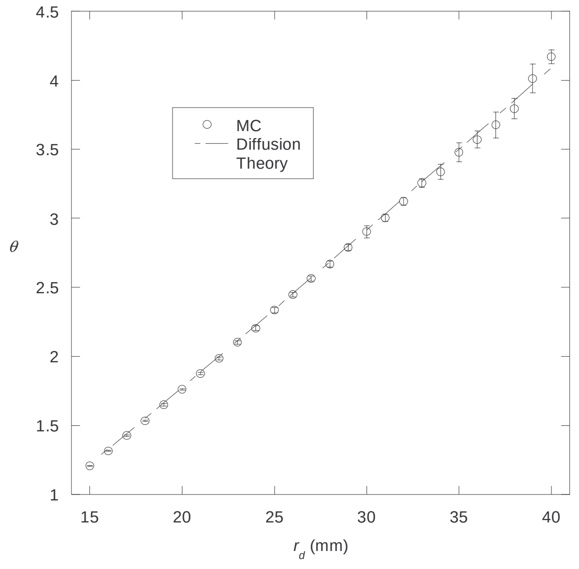


(a)

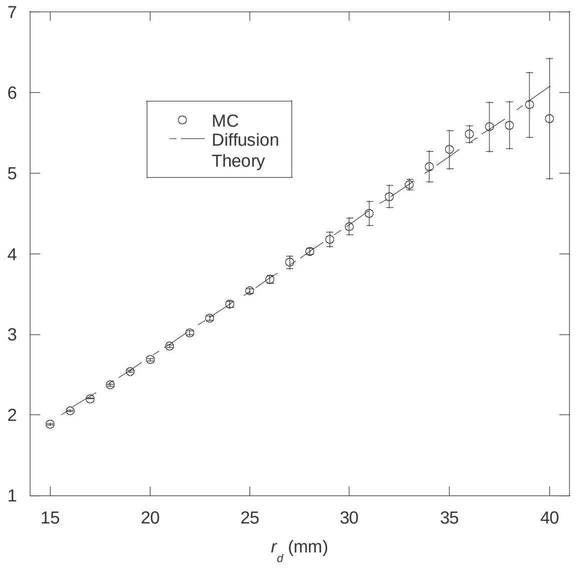


(b)

Figure 4-17: Comparison between diffusion theory and Monte Carlo simulations of *AC* values for case #3 ($\mu_s' = 3477$ m^{-1}, $L = 0.0395$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.



(a)



(b)

Figure 4-18: Comparison of the phase shift between diffusion theory and Monte Carlo simulations for case #3 ($\mu_s' = 3477$ m^{-1}, $L = 0.0395$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

Case 5

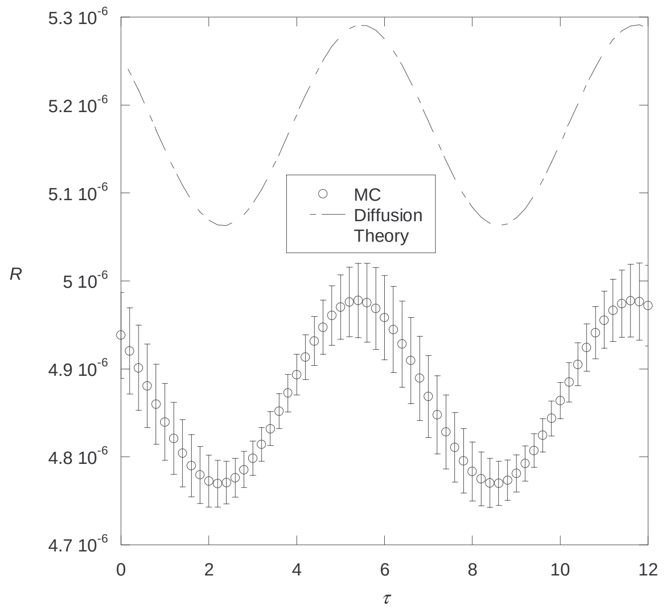


Figure 4-19: Frequency-domain comparison between diffusion theory and Monte Carlo simulations for case #5 ($\mu_s' = 8238$ m^{-1}, $L = 0.1794$ m) at $r_d = 25$ mm and $f_o = 100$ MHz.

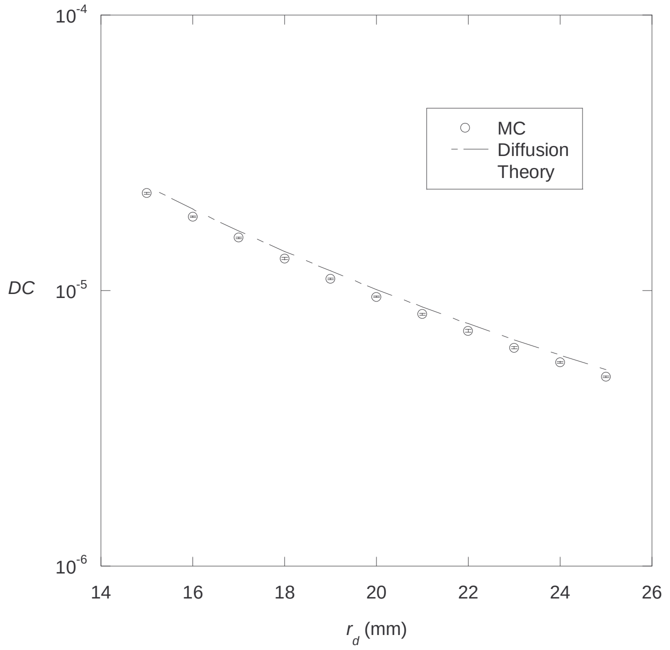
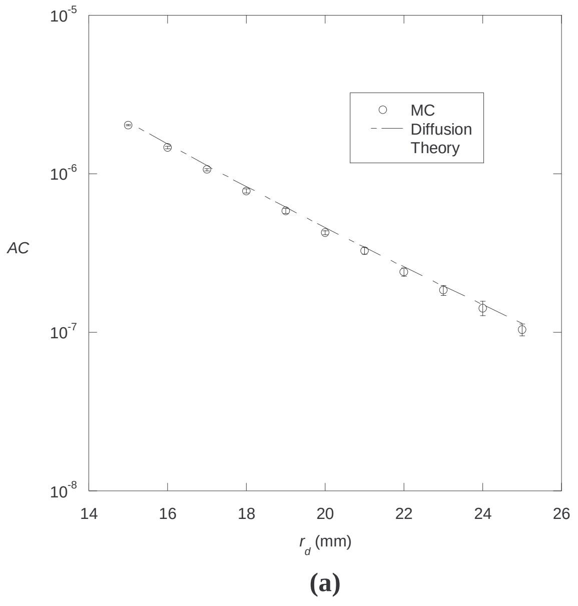


Figure 4-20: Comparison between diffusion theory and Monte Carlo simulations of *DC* values for case #5 ($\mu_s' = 8238$ m^{-1}, $L = 0.1794$ m) at $f_o = 100$ MHz.



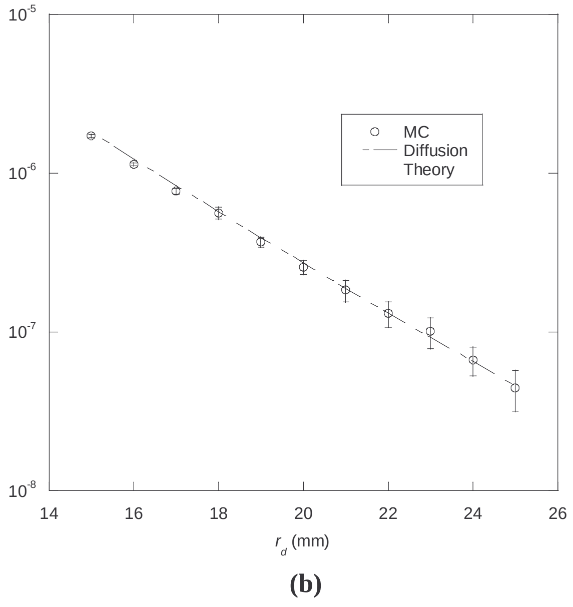
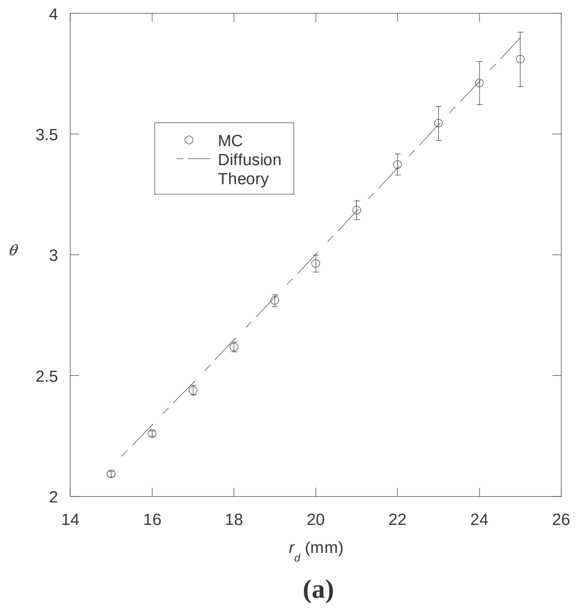


Figure 4-21: Comparison between diffusion theory and Monte Carlo simulations of *AC* values for case #5 ($\mu_s' = 8238$ m^{-1}, $L = 0.1794$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.



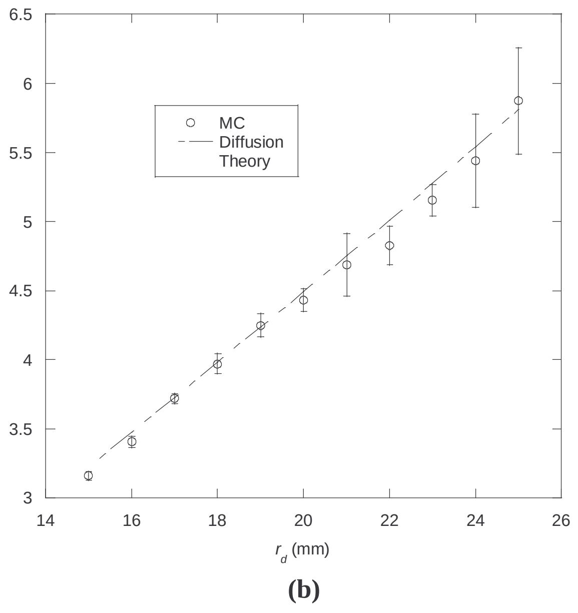


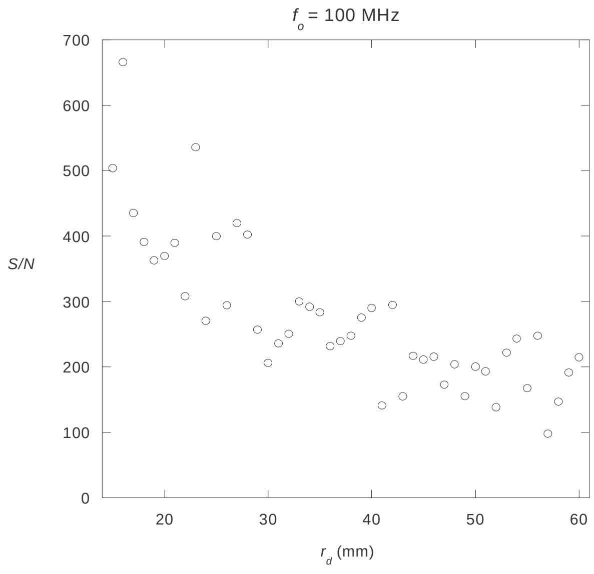
Figure 4-22: Comparison of the phase shift between diffusion theory and Monte Carlo simulations for case #5 ($\mu_s' = 8238$ m^{-1}, $L = 0.1794$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

For all of the cases presented, some recurring trends are observed. As the source-detector separation distance increases, the *AC* and *DC* components decrease. Conversely, as the detector locations get further from the source, the phase shift values increase. Intuitively this is an expected result. As the source-detector separation increases, the overall signal decreases, therefore causing the *AC* and *DC* components to decrease. Also, photons that are detected further from the source generally have undergone more scattering interactions and the time required to arrive at that location is generally higher than locations closer to the source. This leads to an increased value in the phase shift. Trends with respect to frequency are also observed. As the frequency is increased, the phase shift also increases. The same trend is seen when considering the *AC* component and *DC* offset of the reflected flux.

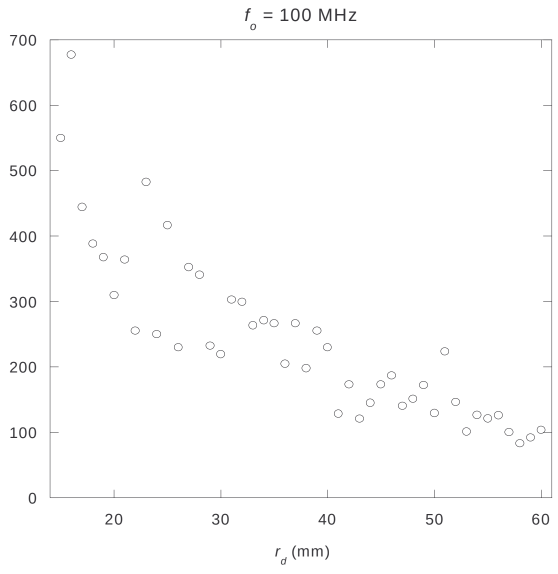
For some of the cases (1, 2, and 9) the level of noise is negligible over the full range of detector locations. However, for the other cases an increasing degree of noise is evident as source-detector separation distance and frequency increases. For all of these cases where the noise is significant, the value of μ_s' is greater than 1000 m^{-1}. This is expected since as μ_s' increases, the mean free path between scattering events decreases, thereby decreasing the number of photons that reach the detectors furthest from the source; hence the signal-to-noise ratio also decreases.

When adapting the inverse method, that is presented in Chapter 5, to actual experimental conditions, it is important to present the signal-to-noise ratios that allowed for successful predictions of the foam properties in order select proper instrumentation.

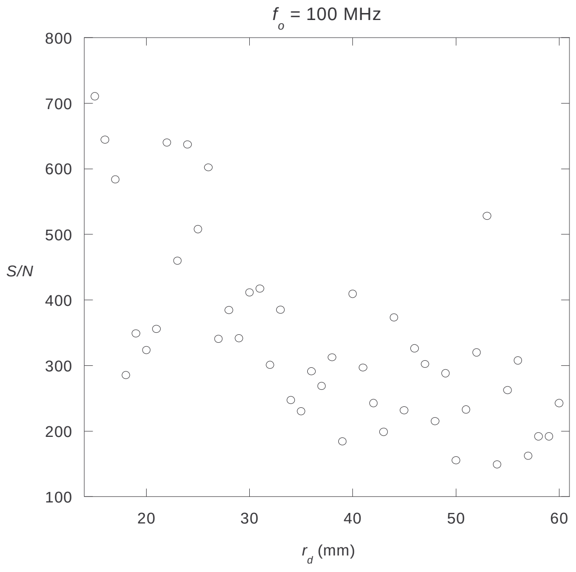
The signal-to-noise ratios for the three cases previously considered are presented as a function of the detector location for a single modulation frequency.



(a)

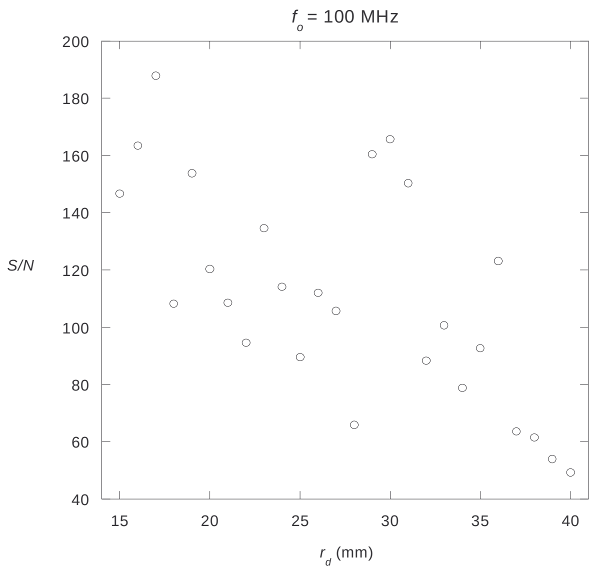


(b)

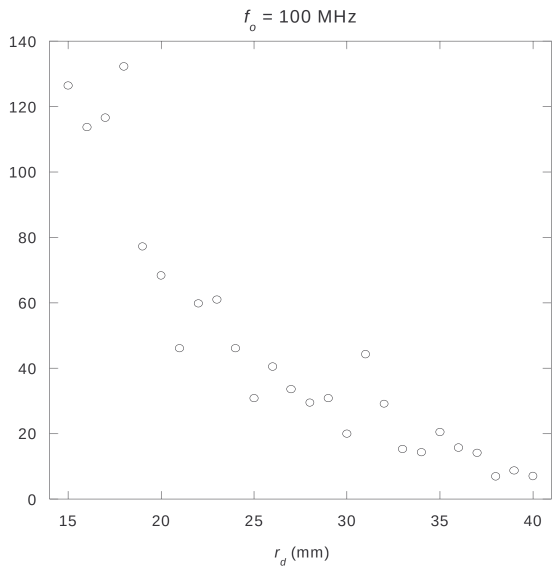


(c)

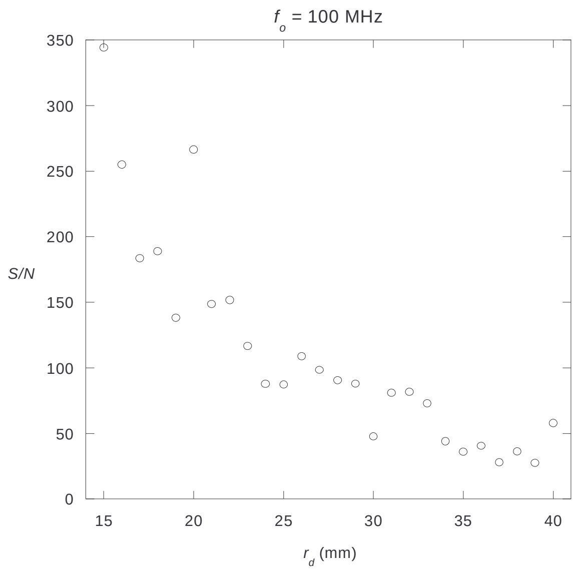
Figure 4-23: Case #1 signal-to-noise ratios (*S/N*) for the (a) *DC* offset, (b) *AC* component, and (c) phase shift plotted as a function of detector location.



(a)



(b)



(c)

Figure 4-24: Case #3 signal-to-noise ratios (*S/N*) for the (a) *DC* offset, (b) *AC* component, and (c) phase shift plotted as a function of detector location.

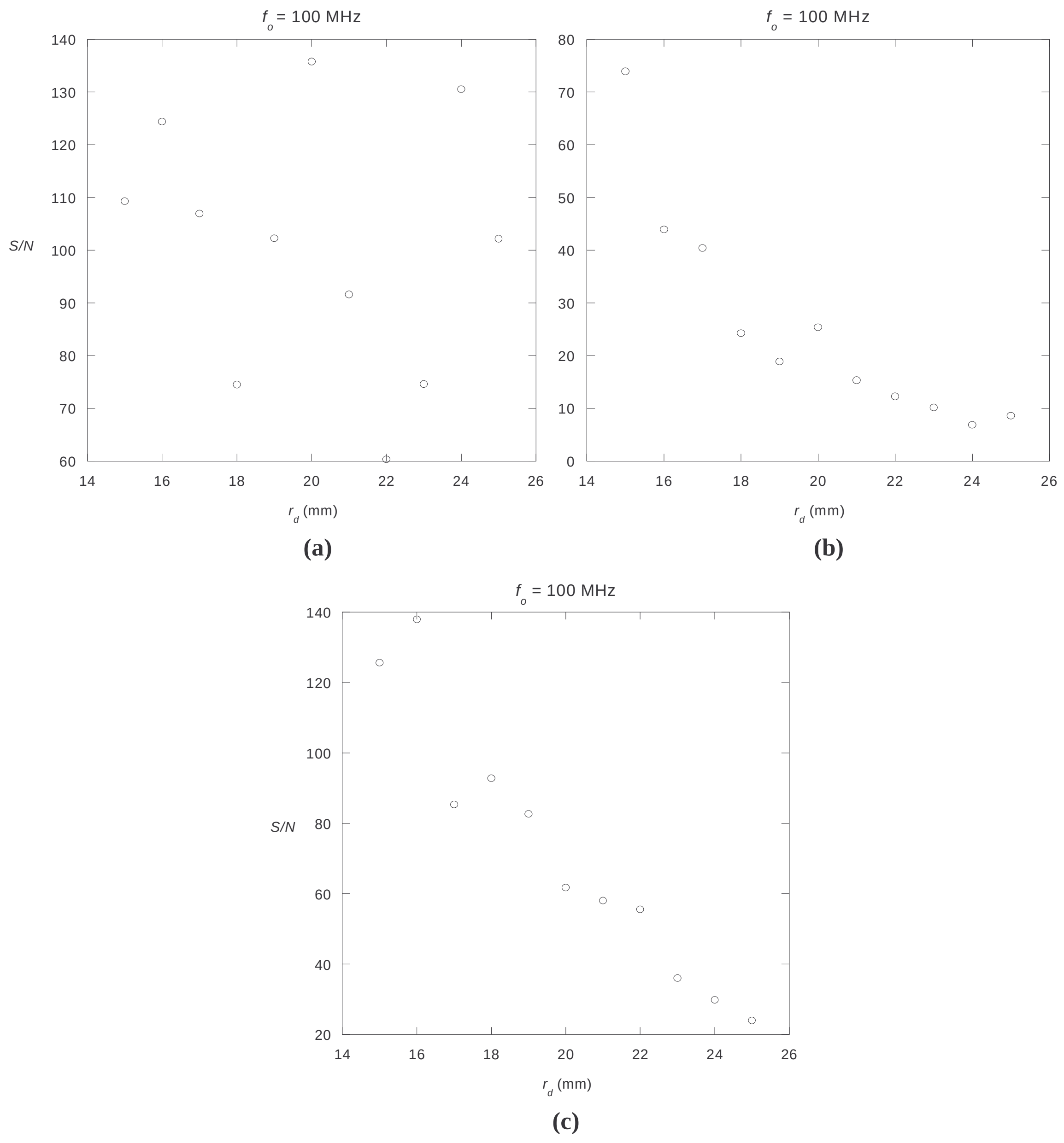


Figure 4-25: Case #5 signal-to-noise ratios (*S/N*) for the (a) *DC* offset, (b) *AC* component, and (c) phase shift plotted as a function of detector location.

Average signal-to-noise ratios of the various ranges of detector locations of all ten cases are presented in Table 4-2 for each measurement type for a single modulation frequency.

Table 4-2. Average signal-to-noise ratios of each measurement type at a modulation frequency of 100 MHz and the combined average.

Case #	$(S/N)_{DC}$	$(S/N)_{AC}$	$(S/N)_{\theta}$	$(S/N)_{comb}$
1	276	243	344	288
2	247	179	276	234
3	111	46	112	90
4	331	321	426	359
5	101	25	72	66
6	120	39	95	85
7	127	62	120	103
8	24	17	61	34
9	316	273	353	314
10	176	88	160	85

5 - DETERMINATION OF THE FOAM LAYER PROPERTIES

5.1 CONJUGATE-GRADIENT ALGORITHM

The use of gradient-based algorithms to solve inverse problems is common practice. For the current research, a conjugate-gradient algorithm was chosen to solve the inverse problem of interest. The basis for this approach was taken from the algorithm development of Özişik and Orlande [8]. The methodology of the conjugate-gradient algorithm is the use of gradients to determine the search direction and a suitable step size is used in order to minimize the objective function. As with any inverse problem, an efficient way of solving the forward problem is necessary. The diffusion theory solution presented in Chapter 3 is used to solve the forward problem.

Each iteration of the inverse algorithm requires the calculation of diffusion theory predicted measurements. These predictions are compared to the Monte Carlo simulated measurements in the form of a normalized objective function. It is necessary to normalize the objective function because of the varying degrees of magnitudes between the measurement types. Estimation of the desired parameters is obtained by minimizing the difference between the simulated and predicted measurement values.

$$X(\mathbf{P}) = \left[\frac{\mathbf{M}-\mathbf{\Psi}(\mathbf{P})}{\mathbf{M}}\right]^T \left[\frac{\mathbf{M}-\mathbf{\Psi}(\mathbf{P})}{\mathbf{M}}\right] \tag{5.1}$$

In Eq. (5.1), **M** represents a vector of simulated measurements obtained from the Monte Carlo method and **Ψ** is the vector of predicted measurements, based on diffusion theory, for the current values of the desired parameters **P**. The iterative procedure for the conjugate-gradient method for the minimization of Eq. (5.1) is given by

$$\mathbf{P}^{k+1} = \mathbf{P}^{k} - \beta^{k}\mathbf{d}^{k} \tag{5.2}$$

where β^k is the search step size, $\mathbf{d}^k$ is the direction of descent and the superscript k represents the number of iterations. The direction of descent combines the current gradient direction $\nabla X\left(\mathbf{P}^k\right)$, and the direction of descent of the previous iteration.

$$\mathbf{d}^{k} = \nabla X\left(\mathbf{P}^{k}\right) + \gamma^{k}\mathbf{d}^{k-1} \tag{5.3}$$

γ^k is known as the conjugation coefficient and assures that the negative gradient direction is less than 90° and is defined as

$$\gamma^{k} = \frac{\sum_{j=1}^{N}\left\{\left[\nabla X\left(\mathbf{P}^{k}\right)\right]_j \left[\nabla X\left(\mathbf{P}^{k}\right) - \nabla X\left(\mathbf{P}^{k-1}\right)\right]_j\right\}}{\sum_{j=1}^{N}\left[\nabla X\left(\mathbf{P}^{k}\right)\right]_j^2} \qquad \text{for } k = 1, 2, \ldots \tag{5.4}$$

with $\gamma^0 = 0$ for $k = 0$ and N represents the number of unknown parameters. Equation (5.5) is known as the Polak-Ribiere expression [33,34]. Another possible expression for the conjugation gradient is the Fletcher-Reeves expression [33,34,35]. The Polak-Ribiere expression is used for this problem because there is some evidence that it provides improved convergence in nonlinear estimation problems [31,34]. $\nabla X\left(\mathbf{P}^k\right)$ is an Nx1 matrix, and the j^{th} component of the gradient direction evaluated at iteration k is simply the j^{th} entry of the column vector. The expression for the gradient direction is obtained by differentiating Eq. (5.1) with respect to the vector of unknown parameters **P**.

$$\nabla X\left(\mathbf{P}^k\right) = -2\left(\mathbf{J}^k\right)^T \left[\frac{\mathbf{M}-\mathbf{\Psi}\left(\mathbf{P}^k\right)}{\mathbf{M}}\right] \tag{5.6}$$

$\mathbf{J}^k$ is the normalized sensitivity matrix given by the following equation.

$$\mathbf{J}\left(\mathbf{P}\right) = \begin{bmatrix} \frac{P_1}{M_1}\frac{\partial \Psi_1}{\partial P_1} & \frac{P_2}{M_1}\frac{\partial \Psi_1}{\partial P_2} & \cdots & \frac{P_N}{M_1}\frac{\partial \Psi_1}{\partial P_N} \\ \frac{P_1}{M_2}\frac{\partial \Psi_2}{\partial P_1} & \frac{P_2}{M_2}\frac{\partial \Psi_2}{\partial P_2} & \cdots & \frac{P_N}{M_2}\frac{\partial \Psi_2}{\partial P_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{P_1}{M_I}\frac{\partial \Psi_I}{\partial P_1} & \frac{P_2}{M_I}\frac{\partial \Psi_I}{\partial P_2} & \cdots & \frac{P_N}{M_I}\frac{\partial \Psi_I}{\partial P_N} \end{bmatrix} \tag{5.7}$$

I represents the total number of measurements (including all measurement types) used in the inverse problem. Due to the complex nature of the diffusion theory solution, the derivatives of the measurements with respect to the parameters were calculated numerically using a central difference scheme.

Finally the search step size β^k can be determined by minimizing $X\left(\mathbf{P}^{k+1}\right)$ with respect to β^k. Details of the derivation are presented in detail elsewhere [8].

$$\beta^k = \frac{\left[\mathbf{J}^k \mathbf{d}^k\right]^T \left[\mathbf{\Psi}\left(\mathbf{P}^k\right) - \mathbf{M}\right]}{\left[\mathbf{J}^k \mathbf{d}^k\right]^T \left[\mathbf{J}^k \mathbf{d}^k\right]} \tag{5.8}$$

Having calculated Eqs. (5.3) - (5.8), the iterative procedure given in Eq. (5.2) can be implemented until the updated parameters cease to change from one iteration to the next (within some level of error).

Because many of the necessary calculations are computationally expensive, and in order to accommodate a large measurement data set, the inverse problem was coded in the C++ programming language and executed on various supercomputers available through CAEDM at BYU. A copy of the complete code can be found in Appendix D. The following section shows the results of the inverse problem for the ten cases previously discussed.

5.2 Inverse Solution for Blind Cases

This section is dedicated to illustrating the ability to accurately predict foam layer properties through the use of a conjugate-gradient inversion algorithm. The inverse algorithm is based on diffusion theory and uses Monte Carlo simulated measurements as inputs. For the same three cases that have been used up to this point, the conjugate-gradient algorithm is used to predict the layer properties with no prior knowledge of their values. The results of the other seven cases are given in Appendix B. The inverse algorithm was performed blindly with respect to the properties to be estimated. As was previously mentioned, the actual values of the properties were not known before performing the inverse algorithm. The Monte Carlo method was used to simulate the measurement set (*DC*, *AC*, and θ) at three specified frequencies over a range of detector locations. The range of detector locations used in the inverse algorithm for the individual cases was 15-60 mm, corresponding to the range used in the previous chapter. The use of a large measurement set usually improves the parameter estimation in inverse problems. Ten initial guesses for the property values were randomly generated as starting points for the inverse algorithm for each case. Considering different start values and ensuring that the algorithm converges to the same point, helps to make certain that the algorithm is not trapped at a local minimum. The iteration history is presented for a few of the start values to illustrate common convergence for different initial guesses.

<u>*Case 1*</u>

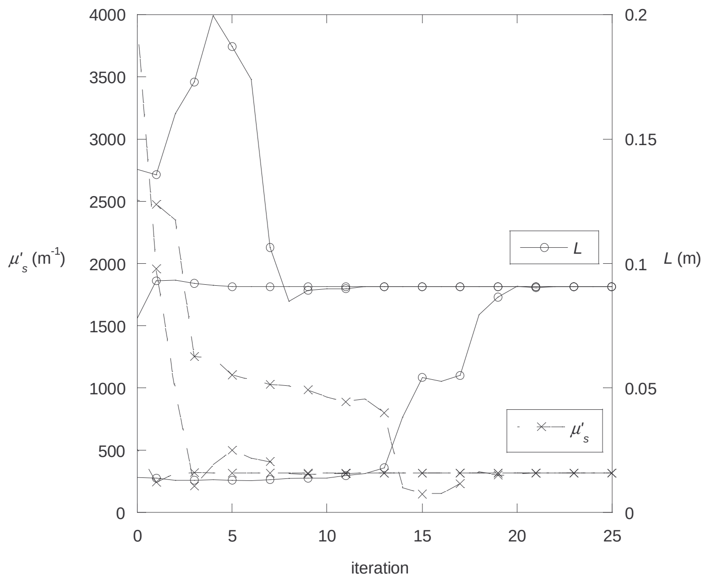


Figure 5-1: Iteration history of the inverse algorithm for case #1.

The iteration history shows convergence for two of the start values after about 12 iterations. In each case, the algorithm converged to the same values. Similar trends are observed for the other cases. The number of iterations necessary to achieve convergence varied from 5 to 25. Typically, a start value far from the converged value required more iterations to reach convergence.

Case 3

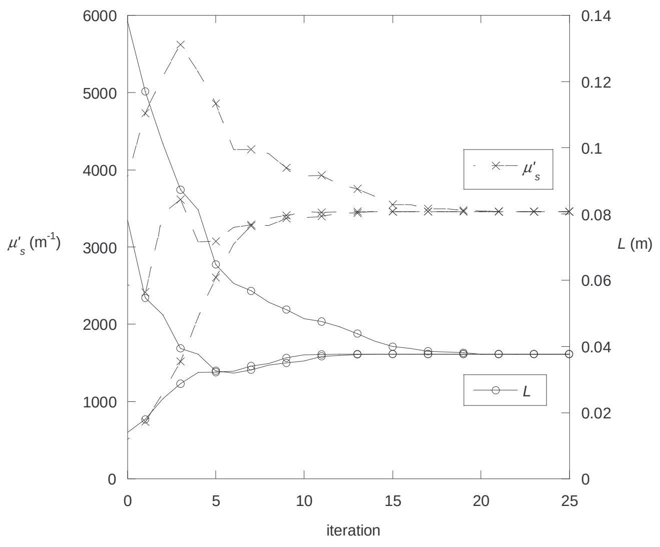


Figure 5-2: Iteration history of the inverse algorithm for case #3.

Case 5

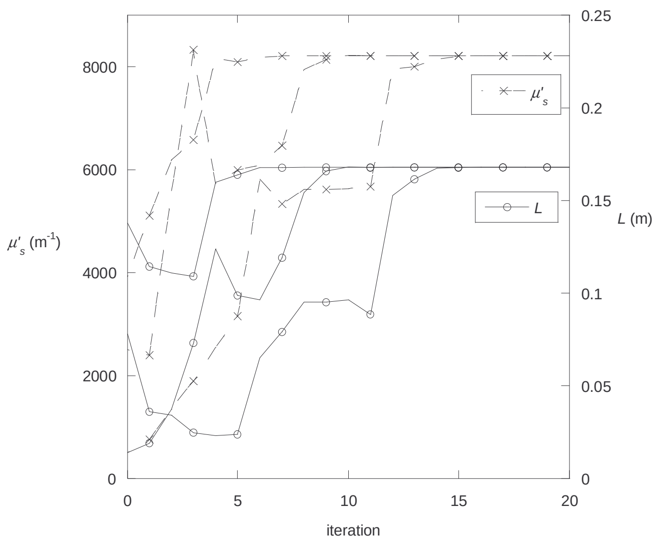


Figure 5-3: Iteration history of the inverse algorithm for case #5.

For each case, the iteration histories converged to the same values. As was mentioned earlier, a total of ten random start guesses were used to assure that a global optimum was reached. With all ten of the start values for each of the ten cases, convergence was achieved. Generally speaking convergence was attained by about twenty iterations. A couple of exceptions were observed to this commonality. For case 8, about 40 iterations were needed for both parameters to reach converged values (see Appendix B). Nevertheless, the algorithm proved to be consistent in the prediction of foam layer properties. At this point the predicted values can be compared to the actual property values that were used as inputs to the Monte Carlo simulation.

Table 5-1. Summary of the predicted values as compared to the actual values for the ten blind cases and the corresponding relative error for each parameter.

	Actual Values		**Predicted Values**		**Relative Errors**	
Case #	μ'_s **(m^{-1})**	L **(m)**	μ'_s **(m^{-1})**	L **(m)**	$\varepsilon_{\mu'_s}$	ε_L
1	323	0.0940	317.0	0.0907	1.9%	3.5%
2	517	0.1376	511.1	0.1307	1.2%	5.0%
3	3477	0.0395	3460.1	0.0378	0.5%	4.4%
4	85	0.0655	82.6	0.0661	3.0%	0.9%
5	8238	0.1794	8125	0.1655	1.4%	7.8%
6	3866	0.1298	3745	0.1164	3.1%	10.3%
7	2731	0.1838	2688	0.1690	1.6%	8.1%
8	13358	0.0086	13308	0.00856	0.4%	0.5%
9	229	0.1803	216.2	0.1709	5.4%	5.2%
10	1185	0.1529	1172	0.1420	1.1%	7.1%

As can be seen in the columns representing the errors, the predictions of the layer properties agree very well with the actual values. The predictions for μ_s' resulted in less error than the predictions for the layer thickness L. The case with the worst prediction of μ_s' is number 9. The scattering coefficient (μ_s) for this case is the smallest of the ten cases (see Table 4-1). The case with the best predictions of μ_s' is number 8, where μ_s is the largest. These results are consistent with the observation that diffusion theory more accurately models radiative transport as the scattering coefficient increases [13]. Figure 5-4 shows $\varepsilon_{\mu_s'}$ as a function of μ_s. This figure shows the upper limit of the error decreasing as μ_s increases for all ten cases. Similar trends are seen in Fig 5-5 where $\varepsilon_{\mu_s'}$ is plotted as a function of μ_s'. This is expected since the reduced scattering coefficient is directly proportional to the scattering coefficient.

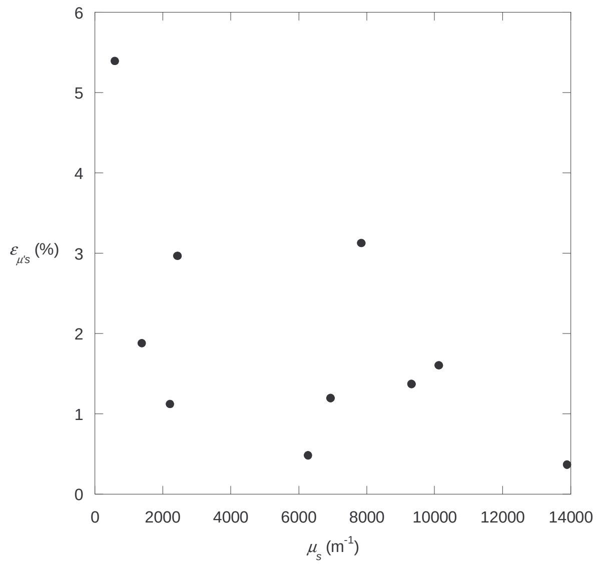


Figure 5-4: The error of the reduced scattering coefficient predictions as a function of the scattering coefficient for all ten cases.

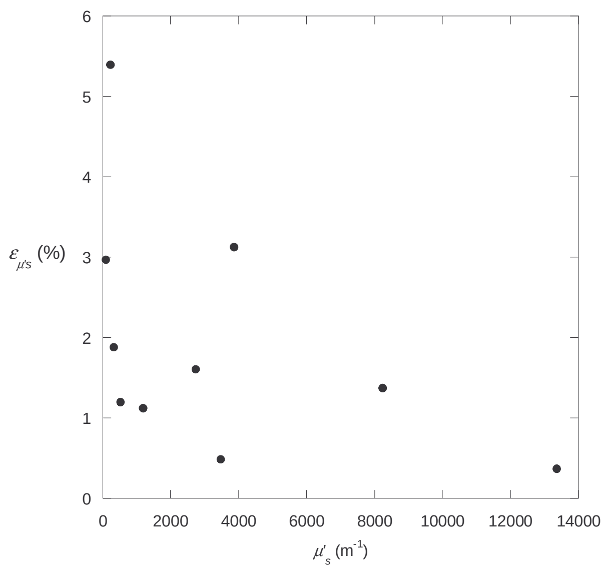


Figure 5-5: The error of the reduced scattering coefficient predictions as a function of the reduced scattering coefficient for all ten cases.

The trends regarding the error in the layer thickness predictions as a function of L are illustrated in Fig 5-6. It appears that generally speaking, ε_L tends to increase with increasing layer thickness.

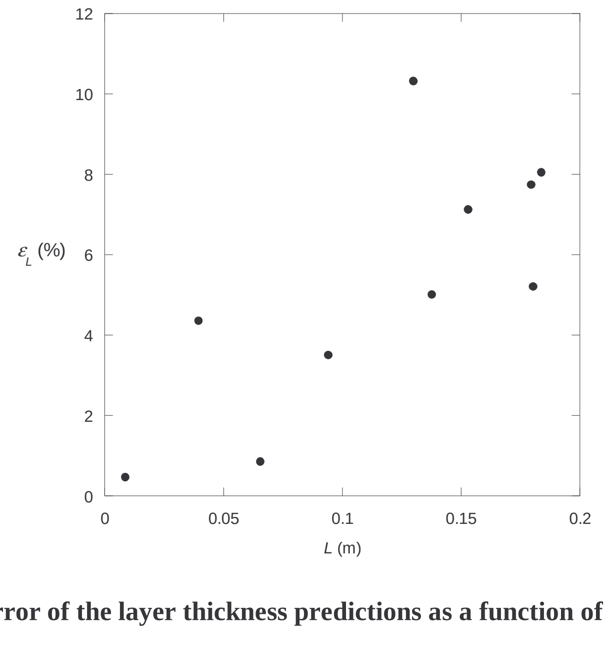


Figure 5-6: The error of the layer thickness predictions as a function of *L* for all ten cases.

As has been shown in comparisons between diffusion theory and Monte Carlo simulations, various levels of modeling accuracy have been observed depending on the source frequency and the source-detector separation distance. These measurement conditions also greatly affect the sensitivities of the measurements with respect to the layer properties. The sensitivities play a significant role in the ability of an inverse algorithm to estimate parameters [7]. These observations indicate that an optimal combination of measurements exists. Although it would be nearly impossible to investigate every combination of such conditions, a method for determining measurement conditions that improves layer property predictions is attempted.

5.3 Determining Optimal Measurement Conditions

The idea of determining optimal measurement conditions in spectroscopic methods is an area of research that, to the author's knowledge, has received little attention. In a recent study, Francoeur et al. [25] provided some insightful information on the subject of inverse problems. By use of the discrete-ordinates finite-volume method for modeling the radiation transport, a preliminary sensitivity analysis was performed to determine some optimal parameters for the design of frequency-modulation based optical diagnostic techniques. An important aspect of inverse problems involves the sensitivities of the measurements to the unknown parameters and plays a key role in parameter estimation. The authors gave a general criterion that states estimation of the targeted property is feasible when the normalized sensitivity coefficients are greater than 0.1.

Accurate estimation of the targeted property is difficult when the normalized sensitivities are between 0.01 and 0.1, and essentially impossible for sensitivities less than 0.01. They concluded the use of frequency domain measurement techniques is very promising to characterize participating media for a wide spectrum of engineering applications and that consideration of the sensitivities will make it possible to select optimal measurement conditions for measurement setups, thereby improving the ability to estimate layer properties.

Özişik and Orlande [8] present a method of designing for optimum experiments, where the idea is to maximize the determinant of the sensitivity matrix **J** (see Section 5.1). However, at times the location of the maximum sensitivities may not provide enough information regarding the optimal measurement conditions. Because diffusion theory is an approximate model, an assessment of the modeling accuracy should be included. Often times these two factors behave in an opposite manner as shown in Fig 5-7. Therefore, a method that balances the effects of the sensitivities and modeling accuracy is proposed.

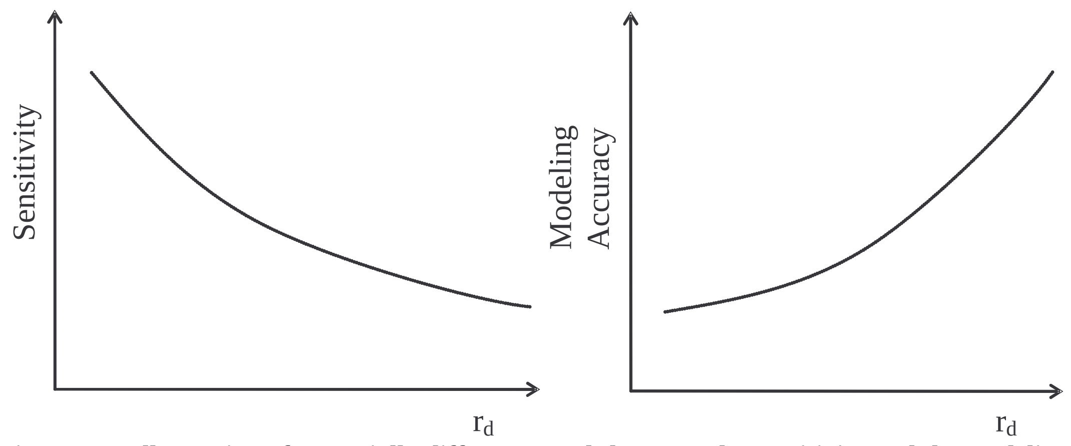


Figure 5-7: Illustration of potentially different trends between the sensitivity and the modeling accuracy.

It is desirable to combine the two effects into a single objective function. This is a challenging task since their values are on different orders of magnitude. The difficulty arises in proper scaling of the two effects in order to appropriately combine them into a single objective. The combined objective (Γ), is set to be minimized and the general form of the objective can be written as shown in the following equation.

$$\Gamma = \frac{f_1 - \tilde{f}_1}{f_1^* - \tilde{f}_1} + \frac{f_2 - \tilde{f}_2}{f_2^* - \tilde{f}_2} \tag{5.9}$$

The term Γ is used to denote the combined effects of the sensitivities and the modeling error. The modeling error is simply the relative error between the simulated measurements and the predicted measurements. The subscript *1* represents the sensitivities and the subscript *2* denotes the modeling error. The different symbols represent user specified indifference values (~) and allowable values (*). The indifference value is defined as the point beyond which the user is no longer concerned with whether or not improvements are made. The allowable value is exactly what it says, the point at which if no further improvements are made, then the results would be allowable. The selection of these values is somewhat arbitrary and was chosen by considering the maximum and minimum values for the investigated conditions. Equation (5.9) is defined such that, at the indifference values, the combined objective is equal to zero and at the allowable values the overall objective is equivalent to two.

The individual objectives are defined in the following equations. The first part of the overall objective regards the sensitivities. The determinant of the sensitivity matrix is very important when determining optimal measurement conditions. In the absence of modeling error, the determinant may be used to determine the optimal measurement set.

$$f_1 = \frac{1}{I}\left|\mathbf{J}^T\mathbf{J}\right| \tag{5.10}$$

The term *I* represents the number of measurements being considered. The portion of the objective that includes the effects of modeling error is given in Eq. (5.11).

$$f_2 = \sqrt{\frac{1}{I}\sum_{j=1}^{I}\left(\frac{M_j - \Psi_j}{M_j}\right)^2} \tag{5.11}$$

In this equation, M_j represents the j^{th} component of the simulated measurement vector, and Ψ_j represents the j^{th} component of the predicted measurement vector based on the current estimate of the properties of the foam layer.

Now that the objective is defined, the procedure for determining the optimal measurement set is set forth. Because this is a non-linear problem (the sensitivities are a function of the unknown parameters), only a local optimum experimental design is possible by using some prior information regarding the expected values for the unknown parameters. Therefore, the first step in this procedure is to obtain an initial estimate of the expected values as was done in Section 5.2. Using those predictions, Γ can then be calculated as a function of both frequency and detector range to be used. As was mentioned previously, there is an immense amount of measurement set combinations that could be considered. For this study three frequencies are considered, along with a varying number of detectors. A minimum and maximum detector location is set for each case (the same limits that were used for previous comparisons). Either the minimum or the maximum is used as a reference detector and additional detectors are added either away from or toward the source respectively. The minimum value of Γ is located and the corresponding frequency and range of detectors is then used to re-calculate the layer properties via the inverse algorithm. This procedure is repeated until the measurement conditions, and therefore the predicted layer properties, cease to change. This procedure is illustrated in detail for one of the ten cases. The preliminary initial property predictions for case 3 (see Table 5-1 for property values) are used to calculate Γ for the three frequencies and range of detectors. For case 3, the reference detector was at 40 mm and additional detectors were added towards the source up to a maximum of 26 detectors.

From the results shown in Fig 5-8, the minimum error occurs at 100 MHz using about 4 detectors which correspond to the range of 40-37 mm. Using those conditions for

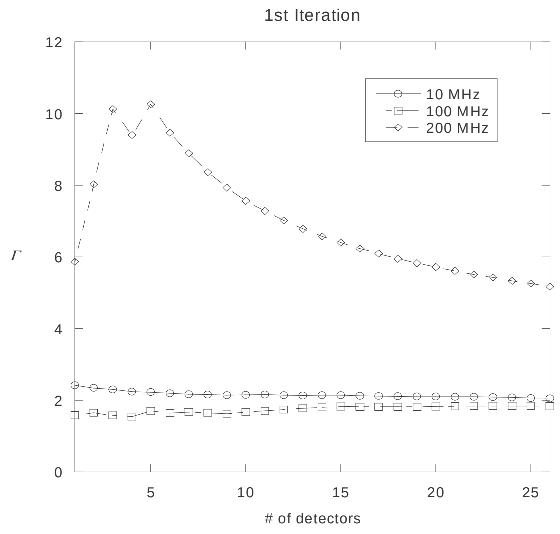


Figure 5-8: The objective function Γ plotted as a function of frequency and the number of detectors for case 3 for the first iteration.

the input measurements into the inverse algorithm, new property predictions are obtained. The original and updated property values are listed below.

$$\mu_s' : 3460.1 \rightarrow 3477.6\ m^{-1}$$

$$L : 0.0378 \rightarrow 0.0381\ m$$

Using these updated values, the objective function is re-calculated to check for any further changes in the measurement set. Figure 5-9 clearly shows that the minimum occurs at 100 MHz when using a single detector location at 40 mm. Re-calculating the layer property predictions yields the following.

$$\mu_s' : 3477.6 \rightarrow 3480.2\ m^{-1}$$

$$L : 0.0381 \rightarrow 0.0382\ m$$

Following the same procedure, the objective is again calculated for the updated values of the properties.

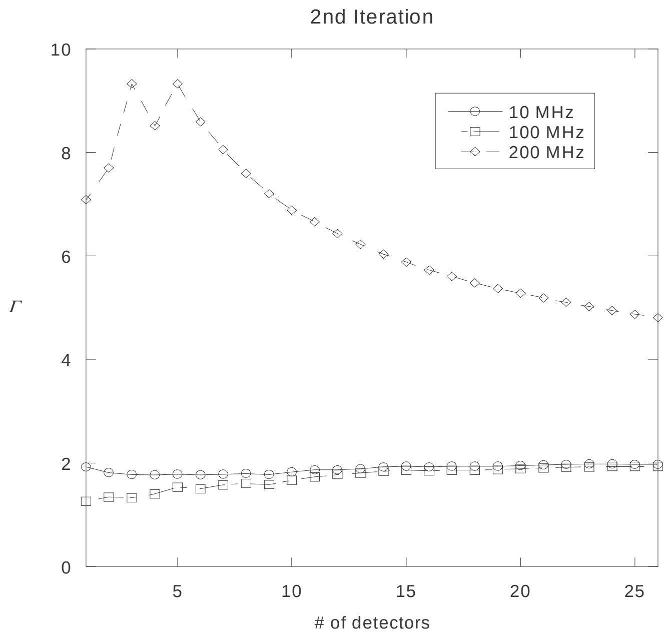


Figure 5-9: The objective function Γ plotted as a function of frequency and the number of detectors for case 3 for the second iteration.

At this point, the optimum has been reached. These results say that the optimum, over the range studied, occurs when using a 100 MHz source frequency at a single detector location of 40 mm.

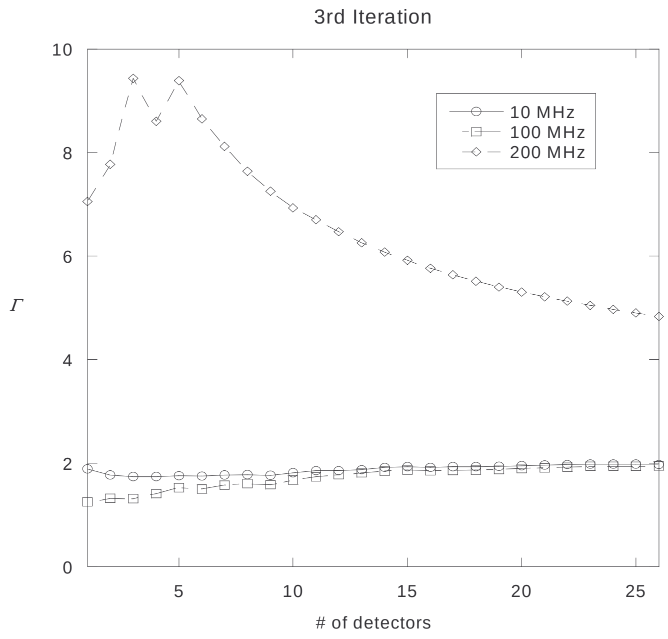


Figure 5-10: The objective function Γ plotted as a function of frequency and the number of detectors for case 3 for the third iteration.

A way of validating this method is necessary. The inverse algorithm can be run for the different combinations of frequencies and detector locations over the range being studied. Figure 5-11 shows the total error for the layer property predictions as a function of both frequency and the number of detectors. Once the actual property inputs used in the blind cases have been disclosed, the total error defined by Eq. (5.12) could be calculated.

$$\varepsilon = 100\sqrt{\left(1-\mu'_{s,pred} / \mu'_{s,act}\right)^2 + \left(1-L_{pred} / L_{act}\right)^2} \tag{5.12}$$

The total error is plotted to verify that the optimal measurement conditions predicted are in fact the optimal measurement conditions.

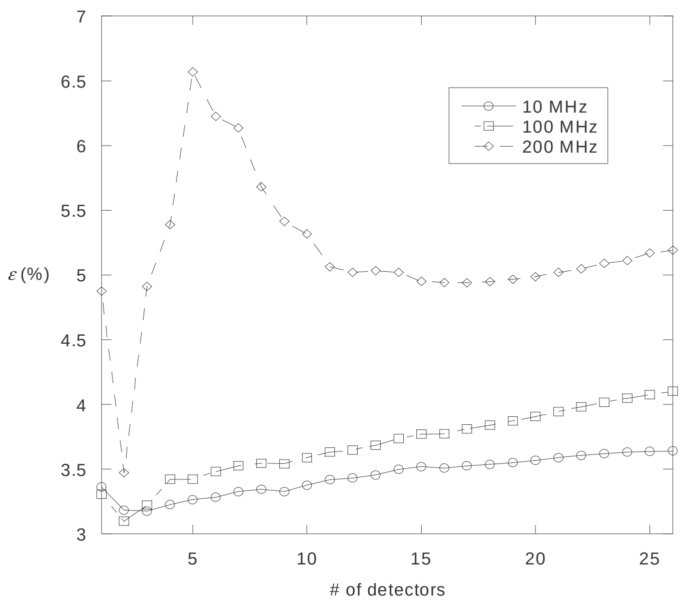


Figure 5-11: Verification of the optimal measurement conditions selected.

This verification plot would indicate that using 2 detectors ranging from 39-40 mm at a source frequency of 100 MHz would result in the smallest error for the range investigated. Thus the proposed method effectively improved the predictions and nearly found the optimum in this case. However, the results for the other nine cases were inconsistent and unreliable. For some of these cases, the total error of the layer property predictions increased after the optimal number of detectors was selected. Table 5-2 shows the results both before and after the optimization with their corresponding total errors as calculated from Eq. (5.13).

Table 5-2. Summary of the predicted values before and after the optimization along with their corresponding total errors. The bolded cases indicate those that improved.

	Before Opt.		**After Opt.**		**Total Error**	
Case #	μ'_s **(m^{-1})**	***L*** **(m)**	μ'_s **(m^{-1})**	***L*** **(m)**	**Before**	**After**
1	317.0	0.0907	318.4	0.0895	4.0%	5.0%
2	511.1	0.1307	509.3	0.1280	5.2%	7.2%
3	**3460.1**	**0.0378**	**3480.2**	**0.0382**	**4.4%**	**3.3%**
4	82.6	0.0661	86.8	0.0677	3.1%	3.9%
5	**8125**	**0.1655**	**8133**	**0.1665**	**7.9%**	**7.3%**
6	**3745**	**0.1164**	**3745.4**	**0.1177**	**10.8%**	**9.8%**
7	2688	0.1690	2684.3	0.1673	8.2%	9.1%
8	13308	0.00856	13300	0.00855	0.6%	0.7%
9	**216.2**	**0.1709**	**216.1**	**0.1731**	**7.5%**	**6.8%**
10	**1172**	**0.1420**	**1192**	**0.1422**	**7.2%**	**7.0%**

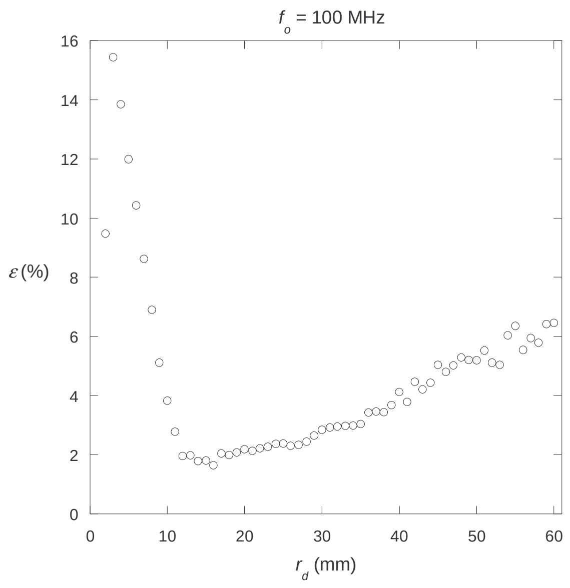
Although half of the cases showed some improvement, the change was very slight. Generally speaking, whether improvements were made or not, the level of change for the property predictions was very small, insignificant enough that the use of this method is not worth the cost of performing the optimization portion of the algorithm.

Further consideration regarding the choice of the objective function may explain the inconsistencies of the optimization. The approach used to include the effects of the modeling error may not lead to improved property predictions. The way the objective is defined, the number of detectors and their locations are selected based partially on how well the measurement predictions agree with the simulated measurements at the current estimate of the properties. However, because diffusion theory may not be in good agreement near the actual values of the parameters, this optimization routine may in fact take the property predictions further from the actual values to a location where the measurements are in better agreement, but where the predicted properties may not be in good agreement. This observation illustrates the need to further investigate more appropriate methods when selecting the number of measurements and their locations.

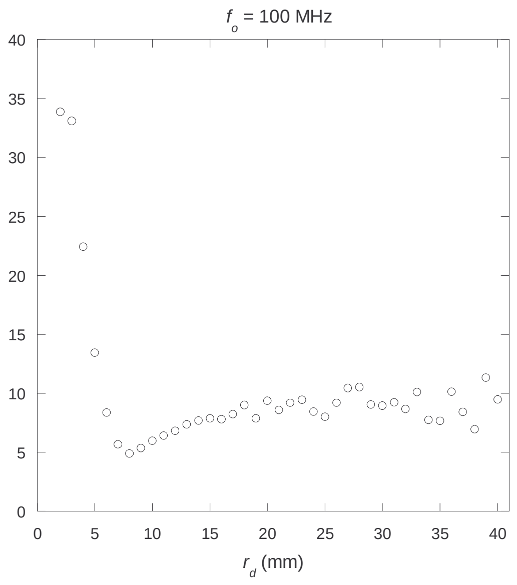
5.4 Systematic Selection of a Minimal Number of Detectors

Up to this point, a large number of measurements have been used to predict properties. Although this approach has proven to be effective in accurately predicting the layer properties, the cost of taking such a large number of measurements in a physical lab is high. Therefore a method for using a select few measurements is discussed.

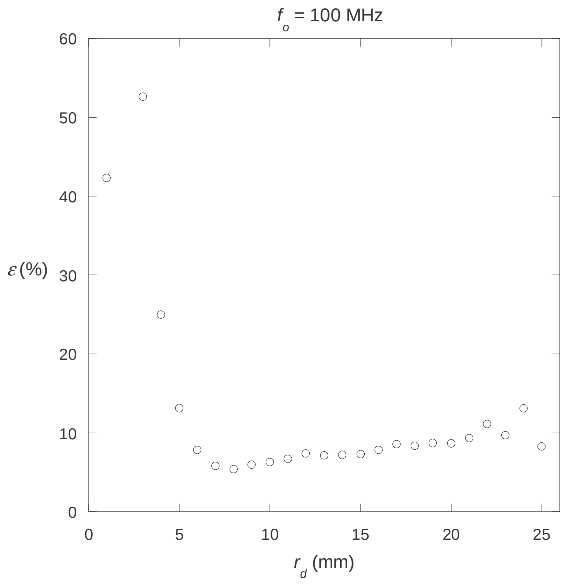
This approach is developed based on the fact that the actual layer properties are already known. However, it is expected that this approach, once developed, can be applied to situations where the actual layer properties are unknown. For a number of cases with varying property values, diffusion theory based measurements were calculated and compared to Monte Carlo simulations (see Section 4.4). The combined relative error, including all three measurement types, was calculated as a function of r_d at a single modulation frequency of 100 MHz. Figure 5-12 shows representative plots of the error for the same three cases that have been considered thus far. As can be seen in Fig 5-12, a clear minimum error occurs at an intermediate value of r_d. This minimum value essentially shows the point where modeling error (dominates left of the minimum) and noise (dominates right of the minimum) are balanced. For the three cases shown here, a clearly defined minimum occurs at different locations. Other cases were also considered and minima were observed for each case.



(a)



(b)



(c)

Figure 5-12: Total measurement error between diffusion theory and Monte Carlo simulations for (a) case #1, (b) case #3, and (c) case #5 plotted as a function of detector location.

The detector location resulting in the combined minimum measurement error was identified for each case and plotted as a function of the product $\mu_s' L$ in order to observe the behavior. Figure 5-13 shows a plot of the detector location at which the combined measurement error plotted as a function of $\mu_s' L$, is a minimum. Recognizing a decaying trend, an exponential curve was fit to the data.

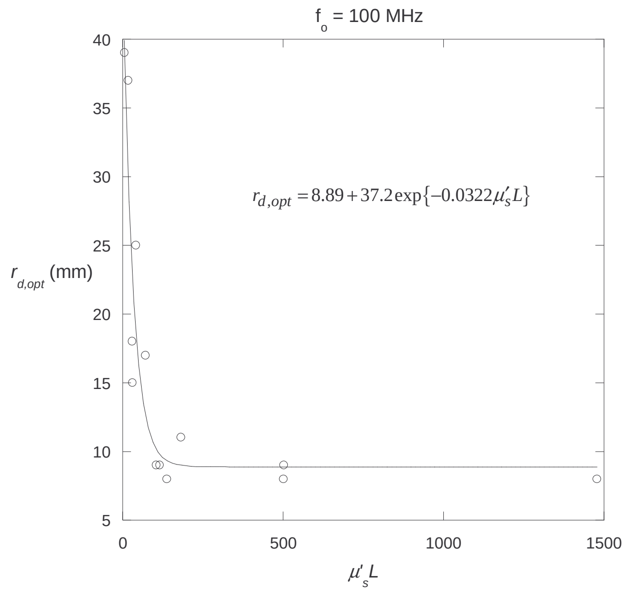


Figure 5-13: Optimal detector location plotted as a function of the product of $\mu_s' L$ with its corresponding curve fit.

It can be seen that the optimal detector location decays very rapidly to a value of approximately 9 mm. Thus for a wide range of $\mu_s' L$, the optimal detector location is roughly the same. Using this information, sequential steps are set forth that lead to accurate property predictions with a reduced number of measurements.

First, the three measurement types at a single detector location of 9 mm are used to obtain the first estimate of the property values based on random initial guesses of the properties. Based on these first estimates, an optimal detector location is calculated from the curve fit shown in Fig 5-13. Measurements at this new location are then used to obtain updated predictions for the properties. These steps are repeated until the calculated optimal detector location stops changing. This procedure was performed for the three cases considered thus far and the iterative progression is summarized in Table 5-3.

Table 5-3. Iterative progression of a method for selecting a reduced number of detector locations.

Case #	$r_{d,opt}$ (mm)	μ'_s (m^{-1})	L (m)	Prediction error
1	9	326.7	0.0906	3.8 %
	23	309.9	0.0916	4.8 %
	24	309.6	0.0914	5.0 %
3	9	3316.5	0.0366	8.7 %
	10	3342.7	0.0366	8.4 %
5	9	7848.7	0.1638	9.9 %

The results in this table clearly show the ability to accurately predict foam layer properties with only a few measurements. For the three cases shown here, the predictions were all within 10%. Although the predictions obtained with a reduced number of measurements may not be as good as when using a large measurement set, the reduction in measurements is significant. For case #1, the progression of optimal detector location selection actually resulted in a higher overall prediction error. This could be caused by two reasons. First, the curve fit for the optimal detector location is not a perfect fit and therefore the property predictions may get slightly worse with the progression of the optimal detector selection. Secondly, the presented technique does not include the effects of sensitivities. As was mentioned previously, the accuracy of property predictions is generally improved by maximizing the sensitivities. For these two reasons, the property predictions may not be the best. Regardless of the less than optimal results, this technique has significantly reduced the number of measurements needed for good property estimation, and the reader should be reminded that the goal of this exercise was to simply demonstrate a technique for which one could systematically select a few detector locations that resulted in good property estimation.

6 - CONCLUSIONS AND RECOMMENDATIONS

6.1 OBJECTIVE SUMMARY

The ability to predict structural and radiative properties of layers through the use of an inverse algorithm has been demonstrated for ten blind cases. A significant part of inverse problems requires a method to accurately solve the forward problem. A diffusion theory model capable of modeling radiative transfer through a non-absorbing, scattering foam layer has been derived by applying simplifying approximations to the RTE. The resulting governing equation was solved through the use of integral transforms in both the time-domain and frequency-domain.

Numerical Monte Carlo methods were used to simulate experimental reflected flux measurements of foam layers. Such measurements were simulated in the time-domain (representative of a pulsed laser source), and reflectance simulations were recorded at various source-detector separations distances. Pulsed simulated measurements were related to representative modulated measurements through the use of a FFT. The development of the FFT came from the consideration of the transfer function for the specified system.

Comparisons of diffusion theory predicted measurements and Monte Carlo simulated measurements were made in both the time-domain and frequency-domain to determine how well diffusion theory modeled simulated measurements and ultimately give insight as to its ability to accurately predict layer properties. Comparisons made in the time-domain showed good agreement for all cases. Agreement between the two tended to improve at larger times and increased source-detector separation distances. However, for such conditions, the signal-to-noise ratio decreases significantly and consequently the uncertainty in the measurements increases. Because measurements in the frequency-domain are less sensitive to noise, comparison between simulated measurements and diffusion theory predictions appears to be better than those of the time-domain. For frequency-domain comparisons, appropriate consideration of the source modulation frequency as well as source-detector separation distance was investigated. Agreement for all ten cases was very good. Evidence of noise is apparent for some of the cases at higher frequencies and for increased source-detector separation. Nevertheless, diffusion theory proved its ability to accurately model radiation propagation over the range of property values considered.

Use of a conjugate-gradient method for solving inverse problems was employed to predict layer properties in ten blind tests. The relative errors in the predicted properties were generally less than 10%, and on average, the errors were approximately 4%. Predictions of μ_s' were all within approximately 5% with the majority being within 3%. This is an impressive result since the actual values of μ_s' range over approximately four orders of magnitude. Generally, the prediction error of μ_s' decreased as μ_s' increased. This was expected since diffusion theory more accurately models radiative transport as

μ_s' increases. Predictions of L were all within approximately 10% with the majority being within 7%. These results are likewise impressive in that the actual values ranged from approximately 8.5 mm to 18 cm.

An attempt at optimizing the modulation frequency and the number of detectors and their locations was made to improve the property predictions. Mixed results were observed with about half of the cases considered showing improved results, while the other half resulted in worse predictions. In either case, the changes in property predictions due to the optimization were minimal.

6.2 RECOMMENDATIONS

It is recommended that the inverse problem be extended to cases where the simulated foam layers include the effects of absorption. The procedure illustrated in this research is easily extended to mildly absorbing foams. An additional problem of interest would be to separate the asymmetry parameter (g) and the scattering coefficient (μ_s) from the reduced scattering coefficient (μ_s'). For this task, a source representation different from the modified Point Source is necessary. Such a representation would require that g and μ_s appear independently so as to satisfy the identifiability criterion [3].

REFERENCES

1. Aubert, J. H., Kraynik, A. M. and Rand, P. B., "Aqueous Foams," *Scientific America*, **254** (5), pp. 74-82 (1986).

2. Pilon, L., Fedorov, A. G. and Viskanta, R., 2002, "Analysis of transient thickness of pneumatic foams," *Chemical Engineering Science*, **57** (6), pp. 977-990.

3. Fedorov, A. G. and Pilon, L., "Glass foams: formation, transport properties, and heat, mass, and radiation transfer," *Journal of Non-Crystalline Solids*, **311** (2), pp. 154-173 (2002).

4. Choudhary, M. K., 2002, "Recent advances in mathematical modeling of flow and heat transfer phenomena in glass furnaces," *Journal of the American Ceramic Society*, **85** (5), pp. 1030 – 1036.

5. Weaire, D. and Hutzler, S., 1999, *The Physics of Foams*, Clarendon Press, Oxford.

6. http://www.thespectroscopynet.com/Educational/Kirchhoff.htm

7. Beck, J. V. and Arnold, K. J., *Parameter Estimation in Engineering and Science*, John Wiley & Sons, New York (1977).

8. Özişik, M. N. and Orlande, H. R. B., *Inverse Heat Transfer*, Taylor & Francis, New York (2000).

9. Pogue, B. W. and Patterson, M. S., "Frequency-domain optical absorption spectroscopy of finite tissue volumes using diffusion theory," *Physics in Medicine and Biology*, **39** (7), pp. 1157-1180 (1994).

10. Fantini, S., Franceschini, M. A. and Gratton, E., “Semi-infinite-geometry boundary problem for light migration in highly scattering media: a frequency-domain study in the diffusion approximation,” *Journal of the Optical Society of America B*, **11** (10), pp. 2128-2138 (1994).

11. Kohl, M., Watson, R. and Cope, M., “Optical properties of highly scattering media determined from changes in attenuation, phase, and modulation Depth,” *Applied Optics*, **36** (1), pp. 105-115 (1997).

12. Fishkin, J. B. and Gratton, E., “Propagation of photon-density waves in strongly scattering media containing an absorbing semi-infinite plane bounded by a straight edge,” *Journal of the Optical Society of America A*, **10** (1), pp. 127-140 (1993).

13. Jones, M. R., Solovjov, V. P., Webb, B. W. and Salisbury, S. M., “Identification of appropriate source models for accurate diffusion modeling of radiative transfer in a non-absorbing foam layer,” *Journal of Quantitative Spectroscopy and Radiative Transfer*, **93** (1-3), pp. 125 – 137 (2005).

14. Farrell, T. J., Patterson, M. S. and Wilson, B., “A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties in vivo”, *Medical Physics*, **19**, pp. 879-888 (1992).

15. Bevilacqua, F., Piguet, D., Marquet, P., Gross, J. D., Tromberg, B. J. and Depeursinge, C., “In vivo determination of tissue optical properties: applications to human brain”, *Applied Optics*, **38** (22), pp. 4939-4950 (1999).

16. Kienle, A. and Patterson, M. S., “Improved solutions of the steady-state and the time-resolved diffusion equations for reflectance from a semi-infinite turbid medium”, *J. Opt. Soc. Am. A*, **14** (1), pp. 246-254 (1997).

17. Pham, T. H., Coquoz, O., Fishkin, J. B., Anderson, A. and Tromberg, B. J., “Broad bandwidth frequency domain instrument for quantitative tissue optical spectroscopy,” *Review of Scientific Instruments*, **71** (6), pp. 2500 – 2513 (2000).

18. Hayakawa, C. K., Hill, B. Y., You, J. S., Bevilacqua, F., Spanier, J. and Venugopalan, V., 2004, “Use of the δ-P_1 approximation for recovery of optical absorption, scattering, and asymmetry coefficients in turbid media,” *Applied Optics*, **43** (24), pp. 4677-4684.

19. Patterson, M. S., Chance, B. and Wilson, B. C., “Time resolved reflectance and transmittance for the non-invasive measurement of tissue optical properties”, *Applied Optics*, **28** (12), pp. 2331-2336 (1989).

20. Madsen, S. J., Wilson, B. C., Patterson, M. S., Park, Y. D., Jacques, S. L. and Hefetz, Y., “Experimental tests of a simple diffusion model for the estimation of scattering and absorption coefficients of turbid media from time-resolved disuse reflectance measurements”, *Applied Optics*, **31** (18), pp. 3509-3517 (1992).

21. Alexandrakis, G., Farrell, T. J. and Patterson, M. S., “Accuracy of the diffusion approximation in determining the optical properties of a two-layer turbid medium”, *Applied Optics*, **37** (31), pp. 7401-7409 (1998).

22. Alexandrakis, G., Busch, D. R., Faris, G. W. and Patterson, M. S., “Determination of the optical properties of two-layer turbid media by use of frequency-domain hybrid Monte Carlo diffusion model”, *Applied Optics*, **40** (22), pp. 3810-3821 (2001).

23. Jacques, S. L., Wang, L. and Hielscher, A. H., “Time-Resolved Photon Propagation in Tissues,” *Optical-Thermal Response of Laser-Irradiated Tissue*, A. J. Welch and M. J. C. van Gemert, editors, Plenum Press, New York, pp. 305- 332 (1995).

24. Patterson, M. S., “Principles and Applications of Frequency-Domain Measurements of Light Propagation,” *Optical-Thermal Response of Laser-Irradiated Tissue*, A. J. Welch and M. J. C. van Gemert, editors, Plenum Press, New York, pp. 333-364 (1995).

25. Francoeur, M., Vaillon, R. and Rousse, D. R., “Theoretical analysis of frequency and time-domain methods for optical characterization of absorbing and scattering Media,” *Journal of Quantitative Spectroscopy and Radiative Transfer*, **93** (1 – 3), pp. 139 – 150 (2005).

26. Svaasand, L. O., “Physics of Laser-Induced Hyperthermia,” *Optical-Thermal Response of Laser-Irradiated Tissue*, A. J. Welch and M. J. C. van Gemert, editors, Plenum Press, New York, pp. 765-787 (1995).

27. Duderstadt, J. J. and Hamilton, L. J, *Nuclear Reactor Analysis*, John Wiley & Sons, New York (1976).

28. Egan, W. G. and Hilgeman, T. W., *Optical Properties of Inhomogeneous Materials, Applications to Geology, Astronomy, Chemistry and Engineering*, Academic Press, New York (1979).

29. Flock, S. T., Patterson, M. S., Wilson, B. C. and Wyman, D. R., "Monte Carlo Modeling of Light Propagation in Highly Scattering Tissues – I: Model Predictions and Comparison with Diffusion Theory," *IEEE Transactions on Biomedical Engineering*," **36** (12), pp. 1162-1167 (1989).

30. Figliola, R. S., and Beasley, D. E., *Theory and Design for Mechanical Measurements*, John Wiley & Sons Inc., New York (2000).

31. Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P., *Numerical Recipes in C*, Cambridge University Press, New York (1992).

32. Wylie, C. R. and Barrett, L. C., *Advanced Engineering Mathematics*, McGraw-Hill Inc., New York (1995).

33. Alifanov, O. M., *Inverse Heat Transfer Problems*, Springer-Verlag, New York (1994).

34. Daniel, J. W., *The Approximate Minimization of Functionals*, Prentice-Hall Inc., Englewood Cliffs (1971).

35. Fletcher, R. and Reeves, C. M., "Function Minimization by Conjugate Gradients," *Computer J.*," **7**, pp. 149-154 (1964).

APPENDIX

APPENDIX A– DETAILED SOLUTION TO THE DIFFUSION APPROXIMATION USING INTEGRAL TRANSFORMS

Beginning with the non-dimensional form of the diffusion equation and its corresponding boundary conditions, the detailed derivation of the diffusion theory model in presented here.

$$\frac{\partial\psi}{\partial\tau}=D\left[\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial\psi}{\partial\rho}\right)+a^2\frac{\partial^2\psi}{\partial\varsigma^2}\right]+\frac{\delta(\rho)}{2\rho}\delta(\varsigma-\varsigma_o')\,f(\tau) \tag{A.1}$$

$$\psi(\rho,\varsigma,0)=0 \tag{A.2}$$

$$\rho\psi(\rho\to\infty,\varsigma,\tau)=0 \text{ and } \rho\frac{\partial\psi}{\partial\rho}\bigg|_{\rho\to\infty}=0 \tag{A.3}$$

$$\rho\psi(0,\varsigma,\tau)=0 \text{ and } \rho\frac{\partial\psi}{\partial\rho}\bigg|_{\rho=0}=0 \tag{A.4}$$

$$\frac{\partial\psi}{\partial\varsigma}\bigg|_{\varsigma=0}=H_oL\psi(\rho,0,\tau) \tag{A.5}$$

$$\frac{\partial\psi}{\partial\varsigma}\bigg|_{\varsigma=1}=-H_LL\psi(\rho,1,\tau) \tag{A.6}$$

Consider the associated Sturm-Liouville problem in ζ.

$$\frac{d^2Z}{d\varsigma^2}+\beta^2Z=0\to Z(\varsigma)=c_1\cos\beta\varsigma+c_2\sin\beta\varsigma \tag{A.7}$$

Applying the two boundary conditions in ζ, yields the following transcendental function for the eigenvalues.

$$\beta L\left(H_o + H_L\right)\cos\beta - \left(\beta^2 - H_o H_L L^2\right)\sin\beta = 0 \tag{A.8}$$

where the eigenfunctions are given by

$$Z_n\left(\varsigma\right) = c_1 \cos\beta_n\varsigma + c_2 \sin\beta_n\varsigma \tag{A.9}$$

The eigenvalues are the positive roots of Eq.(A.8), for n=1,2,…,∞. The normalization factor for the eigenfunction is given by

$$N_n = \int_0^1 Z_n^2\left(\varsigma\right)d\varsigma = \int_0^1 \frac{-1}{\beta_n^2} Z_n \frac{d^2 Z_n}{d\varsigma^2} d\varsigma \tag{A.10}$$

Integrating the previous equations by parts yields the following equation.

$$N_n = \frac{-1}{\beta_n^2}\left[Z_n \frac{dZ_n}{d\varsigma}\right]_0^1 + \int_0^1 \left(\frac{1}{\beta_n}\frac{dZ_n}{d\varsigma}\right)^2 d\varsigma \tag{A.11}$$

Consider the term inside of the integral of Eq. (A.11) and the squared representation of the eigenfunction.

$$\left(\frac{1}{\beta_n}\frac{dZ_n}{d\varsigma}\right)^2 = \sin^2\beta_n\varsigma - \frac{2H_o L}{\beta_n}\sin\beta_n\varsigma\cos\beta_n\varsigma + \frac{2H_o^2 L^2}{\beta_n^2}\cos^2\beta_n\varsigma \tag{A.12}$$

$$Z_n^2 = \cos^2\beta_n\varsigma + \frac{2H_o L}{\beta_n}\sin\beta_n\varsigma\cos\beta_n\varsigma + \frac{H_o^2 L^2}{\beta_n^2}\cos^2\beta_n\varsigma \tag{A.13}$$

Adding the two previous equations together results in the following equation.

$$Z_n^2+\left(\frac{1}{\beta_n}\frac{dZ_n}{d\varsigma}\right)^2=1+\frac{H_o^2L^2}{\beta_n^2} \tag{A.14}$$

Integrating each term of Eq. (A.14) over ζ from 0 to 1 yields the following succession of equations.

$$\int_0^1 Z_n^2 d\varsigma+\int_0^1\left(\frac{1}{\beta_n}\frac{dZ_n}{d\varsigma}\right)^2 d\varsigma=\left(1+\frac{H_o^2L^2}{\beta_n^2}\right)\int_0^1 d\varsigma \tag{A.15}$$

$$N_n+\int_0^1\left(\frac{1}{\beta_n}\frac{dZ_n}{d\varsigma}\right)^2 d\varsigma=\left(1+\frac{H_o^2L^2}{\beta_n^2}\right) \tag{A.16}$$

Solving Eq. (A.11) for the integral term, substituting into Eq. (A.16), and applying the appropriate boundary conditions in ζ, results in the following equation after some simplifying algebra.

$$N_n=\frac{1}{2}\left[\frac{\beta_n^2+H_o^2L^2}{\beta_n^2}\left(1+\frac{H_LL}{\beta_n^2+H_L^2L^2}\right)+\frac{H_oL}{\beta_n^2}\right] \tag{A.17}$$

The kernel for the integral transform can now be defined as follows.

$$k_n(\varsigma)=\frac{Z_n(\varsigma)}{\sqrt{N_n}} \tag{A.18}$$

The Fourier Transform and its inverse can now be introduced and is subsequently used to remove the spatial dependence of ζ.

$$\overline{\psi}(\rho,\beta_n,\tau)=\int_0^1 k_n(\varsigma)\psi(\rho,\varsigma,\tau)d\varsigma \tag{A.19}$$

$$\psi(\rho,\varsigma,\tau)=\sum_{n=1}^{\infty}k_n(\varsigma)\overline{\psi}(\rho,\beta_n,\tau) \tag{A.20}$$

Application of the Fourier Transform to Eq. (A.1) results in the following term-by-term integration.

$$\int_0^1 \frac{\partial \psi}{\partial \tau} k_n(\varsigma)\, d\varsigma = \frac{\partial \overline{\psi}}{\partial \tau} \tag{A.21}$$

$$\int_0^1 \frac{1}{\rho}\frac{\partial}{\partial \rho}\left(\rho \frac{\partial \psi}{\partial \rho}\right) k_n(\varsigma)\, d\varsigma = \frac{1}{\rho}\frac{\partial}{\partial \rho}\left(\rho \frac{\partial \overline{\psi}}{\partial \rho}\right) \tag{A.22}$$

$$\int_0^1 \frac{\partial^2 \psi}{\partial \varsigma^2} k_n(\varsigma)\, d\varsigma = -\beta_n^2 \overline{\psi}(\rho, \beta_n, \tau) \tag{A.23}$$

$$\int_0^1 \delta(\varsigma - \varsigma_o') k_n(\varsigma)\, d\varsigma = k_n(\varsigma_o') \tag{A.24}$$

The Fourier Transformed equation can now be defined.

$$\frac{\partial \overline{\psi}}{\partial \tau} = D \frac{1}{\rho}\frac{\partial}{\partial \rho}\left(\rho \frac{\partial \overline{\psi}}{\partial \rho}\right) - Da^2 \beta_n^2 \overline{\psi} + \frac{\delta(\rho)}{2\rho} k_n(\varsigma_o') f(\tau) \tag{A.25}$$

The forward and inverse Hankel Transforms are given in the following equations respectively. The forward transform is used to remove the remaining spatial dependence.

$$\overline{\overline{\psi}}(\lambda, \beta_n, \tau) = \int_0^\infty \rho J_o(\lambda\rho) \overline{\psi}(\rho, \beta_n, \tau) \tag{A.26}$$

$$\overline{\psi}(\rho, \beta_n, \tau) = \int_0^\infty \lambda J_o(\lambda\rho) \overline{\overline{\psi}}(\lambda, \beta_n, \tau) d\lambda \tag{A.27}$$

Applying the Hankel Transform to Eq. (A.25) takes us one step closer to the solution.

$$\int_0^1 \rho J_o(\lambda\rho) \frac{\partial \overline{\psi}}{\partial \tau} d\rho = \frac{\partial \overline{\overline{\psi}}}{\partial \tau} \tag{A.28}$$

$$\int_0^1 \frac{1}{\rho}\frac{\partial}{\partial \rho}\left(\rho \frac{\partial \overline{\psi}}{\partial \rho}\right) \rho J_o(\lambda\rho)\, d\rho = -\lambda^2 \overline{\overline{\psi}}(\lambda, \beta_n, \tau) \tag{A.29}$$

$$\int_0^1 \rho J_o(\lambda\rho)\frac{\delta(\rho)}{\rho}d\rho = J_o(0) = 1 \tag{A.30}$$

Equation (A.25) can be modified to incorporate the results of the Hankel Transform.

$$\frac{\partial\bar{\bar{\psi}}}{\partial\tau} + D\left(\lambda^2 + a^2\beta_n^2\right)\bar{\bar{\psi}} + \frac{k_n(\varsigma_o')}{2}f(\tau) \tag{A.31}$$

For convenience let the following be defined as $p(\lambda,\beta_n) = D\left(\lambda^2 + a^2\beta_n^2\right)$. Equation (A.31) can be solved using an integrating factor approach, and the solution is listed below.

$$\bar{\bar{\psi}}(\lambda,\beta_n,\tau) = \frac{k_n(\varsigma_o')e^{-p\tau}}{2}\int_0^\tau e^{-p\tau'}f(\tau')d\tau' \tag{A.32}$$

At this point it is necessary to introduce the time-dependent portion of the source term. First consider the case when $f(\tau) = \delta(\tau - \tau_o')$ which corresponds to a pulsed input. Solving Eq. (A.32) for the pulsed input and using a * notation to distinguish its solution from the modulated source input solution that will follow the current discussion yields

$$\bar{\bar{\psi}}^*(\lambda,\beta_n,\tau) = \frac{k_n(\varsigma_o')}{2}e^{-p(\tau-\tau_o')} \tag{A.33}$$

Applying the inverse Hankel Transform given in Eq. (A.27)

$$\bar{\psi}^*(\rho,\beta_n,\tau) = \frac{k_n(\varsigma_o')}{2}\int_0^\infty \lambda J_o(\lambda\rho)\exp\left\{-D\left(\lambda^2 + a^2\beta_n^2\right)(\tau-\tau_o')\right\}d\lambda \tag{A.34}$$

$$\bar{\psi}^*(\rho,\beta_n,\tau) = \frac{k_n(\varsigma_o')}{4D(\tau-\tau_o')}\exp\left\{-a^2\beta_n^2(\tau-\tau_o') - \frac{\rho^2}{4D(\tau-\tau_o')}\right\} \tag{A.35}$$

Applying the inverse Fourier Transform given in Eq. (A.20)

$$\psi^*(\rho,\varsigma,\tau) = \sum_{n=1}^{\infty} \frac{k_n(\varsigma_o')k_n(\varsigma)}{4D(\tau-\tau_o')} \exp\left\{-a^2\beta_n^2 D(\tau-\tau_o') - \frac{\rho^2}{4D(\tau-\tau_o')}\right\} \tag{A.36}$$

The local non-dimensional reflected flux is obtained from the gradient of the fluence rate at the surface of the foam layer and is related to the boundary condition given by Eq.(A.5).

$$R(\rho,\tau) = \frac{1}{3\mu_s' L} \frac{\partial\psi}{\partial\varsigma}\bigg|_{\varsigma=0} = h_o \psi(\rho,0,\tau) \tag{A.37}$$

Where h_o is related to the boundary coefficient by $h_o = H_o / 3\mu_s'$. A relationship for the reflected flux can then be obtained.

$$R(\rho,\tau) = \frac{h_o}{4D(\tau-\tau_o')} \exp\left\{-\frac{\rho^2}{4D(\tau-\tau_o')}\right\} \sum_{n=1}^{\infty} k_n(\varsigma_o')k_n(\varsigma_o) \exp\left\{-a^2\beta_n^2 D(\tau-\tau_o')\right\} \tag{A.38}$$

Now consider the case when $f(\tau) = 1 + A\sin(\tau-\tau_o')$ which corresponds to a modulated input. Solving Eq. (A.32) for the modulated input yields

$$\bar{\bar{\psi}}(\lambda,\beta_n,\tau) = \frac{k_n(\varsigma_o')}{2} e^{-p\tau} \int_0^{\tau} e^{-p\tau'} \left[1 + A\sin(\tau-\tau_o')\right] d\tau' \tag{A.39}$$

Evaluating this integral results in the following

$$\begin{aligned}\bar{\bar{\psi}}(\lambda,\beta_n,\tau) &= \frac{k_n(\varsigma_o')}{2}\left(\frac{A}{p^2+1}(\cos\tau_o' + p\sin\tau_o') - \frac{1}{p}\right)\mathrm{e}^{-p\tau} \\ &\quad + \frac{k_n(\varsigma_o')}{2}\left[\frac{Ap}{p^2+1}\sin(\tau-\tau_o') - \frac{A}{p^2+1}\cos(\tau-\tau_o') + \frac{1}{p}\right]\end{aligned} \tag{A.40}$$

The exponential term decays rapidly for $\tau > 0$ and is therefore neglected and only the periodic portion of the solution is retained.

$$\bar{\bar{\psi}}_p(\lambda,\beta_n,\tau)=\bar{\bar{S}}(\lambda,\beta_n)+\bar{\bar{T}}(\lambda,\beta_n)\sin(\tau-\tau_o')-\bar{\bar{U}}(\lambda,\beta_n)\cos(\tau-\tau_o') \tag{A.41}$$

where $\bar{\bar{S}}(\lambda,\beta_n)$, $\bar{\bar{T}}(\lambda,\beta_n)$, and $\bar{\bar{U}}(\lambda,\beta_n)$ are defined by the following equations.

$$\bar{\bar{S}}(\lambda,\beta_n)=\frac{k_n(\varsigma_o')}{2p(\lambda,\beta_n)} \tag{A.42}$$

$$\bar{\bar{T}}(\lambda,\beta_n)=\frac{A}{2}\frac{k_n(\varsigma_o')\,p(\lambda,\beta_n)}{p(\lambda,\beta_n)^2+1} \tag{A.43}$$

$$\bar{\bar{U}}(\lambda,\beta_n)=\frac{A}{2}\frac{k_n(\varsigma_o')}{p(\lambda,\beta_n)^2+1} \tag{A.44}$$

Applying the inverse Hankel Transform as defined in Eq. (A.27)

$$\bar{\psi}_p(\rho,\beta_n,\tau)=\int_0^\infty \lambda J_o(\lambda\rho)\bar{\bar{\psi}}(\lambda,\beta_n,\tau)\,d\lambda \tag{A.45}$$

$$\begin{aligned}\bar{\psi}_p(\rho,\beta_n,\tau)=&\frac{k_n(\varsigma_o')}{2}\bar{S}(\rho,\beta_n)+\frac{A}{2}k_n(\varsigma_o')\bar{T}(\rho,\beta_n)\sin(\tau-\tau_o')\\&-\frac{A}{2}k_n(\varsigma_o')\bar{U}(\rho,\beta_n)\cos(\tau-\tau_o')\end{aligned} \tag{A.46}$$

where $\bar{S}(\lambda,\beta_n)$, $\bar{T}(\lambda,\beta_n)$, and $\bar{U}(\lambda,\beta_n)$ are defined by the following equations.

$$\bar{S}(\lambda,\beta_n)=\int_0^\infty \frac{1}{p(\lambda,\beta_n)}d\lambda \tag{A.47}$$

$$\bar{T}(\lambda,\beta_n)=\int_0^\infty \frac{p(\lambda,\beta_n)}{p(\lambda,\beta_n)^2+1}d\lambda \tag{A.48}$$

$$\bar{U}(\lambda,\beta_n)=\int_0^\infty \frac{1}{p(\lambda,\beta_n)^2+1}d\lambda \tag{A.49}$$

Applying the inverse Fourier Transform given in Eq. (A.20)

$$\psi_p(\rho,\varsigma,\tau)=\sum_{n=1}^\infty S(\rho,\varsigma)+T(\rho,\varsigma)\sin(\tau-\tau_o')-U(\rho,\varsigma)\cos(\tau-\tau_o') \tag{A.50}$$

The non-dimensional reflected flux for the frequency domain is also given by Eq. (A.37). Some algebraic manipulation gives the following results.

$$\tilde{R}(\rho,\tau)=h_o S(\rho,\beta_n)+h_o\sqrt{T(\rho,\beta_n)^2+U(\rho,\beta_n)^2}\,\sin\left(\tau-\tilde{\theta}\right) \tag{A.51}$$

where the phase shift $\tilde{\theta}$ is defined by

$$\tilde{\theta}=\tau_o'+\tan^{-1}\left[\frac{U(\rho,0)}{T(\rho,0)}\right] \tag{A.52}$$

and the other terms are defined below.

$$S(\rho,\beta_n)=\sum_{n=1}^{\infty}\frac{k_n(\varsigma_o')k_n(0)}{2}\int_0^{\infty}\frac{\lambda J_o(\lambda\rho)}{p(\lambda,\beta_n)}d\lambda=\sum_{n=1}^{\infty}\frac{k_n(\varsigma_o')k_n(0)}{2D}K_o(a\beta_n\rho) \tag{A.53}$$

$$T(\rho,\beta_n)=\sum_{n=1}^{\infty}A\frac{k_n(\varsigma_o')k_n(0)}{2}\int_0^{\infty}\frac{p(\lambda,\beta_n)}{p(\lambda,\beta_n)^2+1}\lambda J_o(\lambda\rho)\,d\lambda \tag{A.54}$$

$$U(\rho,\beta_n)=\sum_{n=1}^{\infty}A\frac{k_n(\varsigma_o')k_n(0)}{2}\int_0^{\infty}\frac{\lambda J_o(\lambda\rho)}{p(\lambda,\beta_n)^2+1}d\lambda \tag{A.55}$$

APPENDIX B– RESULTS FOR ADDITIONAL SEVEN CASES

B.1 Time-Domain Comparisons

The contents of this appendix include comparisons between Monte Carlo simulations and diffusion theory predictions for the additional seven cases not included in the body of this work. Both time-domain and frequency-domain comparisons are given.

Case 2

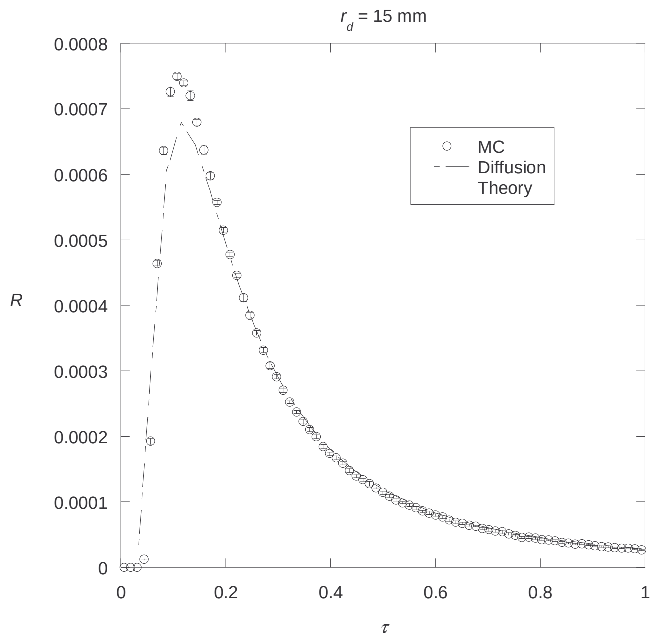


Figure B-1: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #2 ($\mu_s' = 517$ m^{-1}, $L = 0.1376$ m) at a source-detector separation of 15 mm.

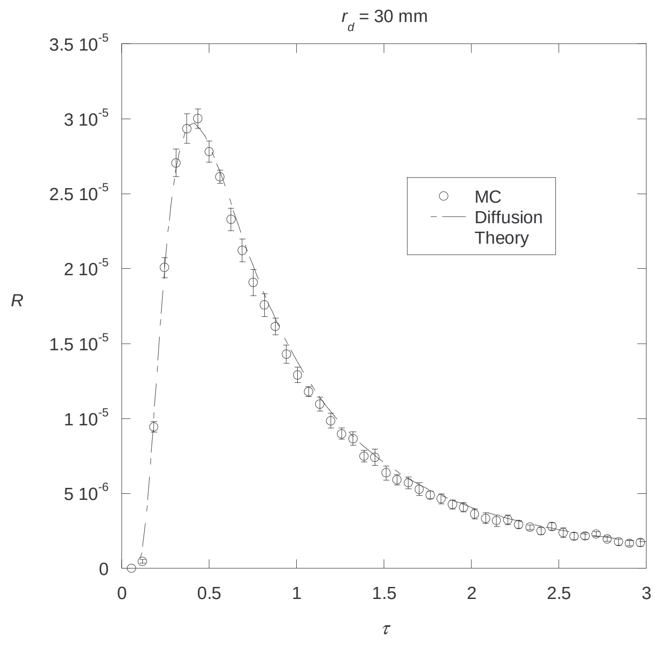


Figure B-2: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #2 ($\mu_s' = 517$ m^{-1}, $L = 0.1376$ m) at a source-detector separation of 30 mm.

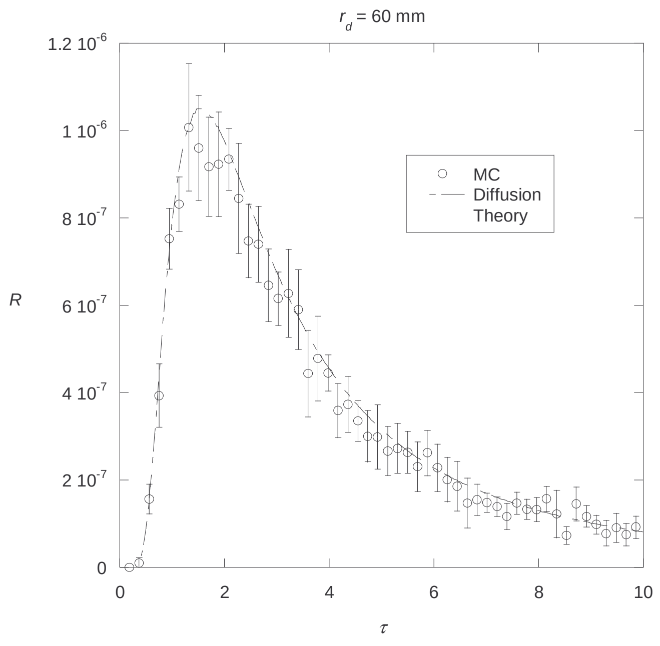


Figure B-3: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #2 ($\mu_s' = 517$ m^{-1}, $L = 0.1376$ m) at a source-detector separation of 60 mm.

Case 4

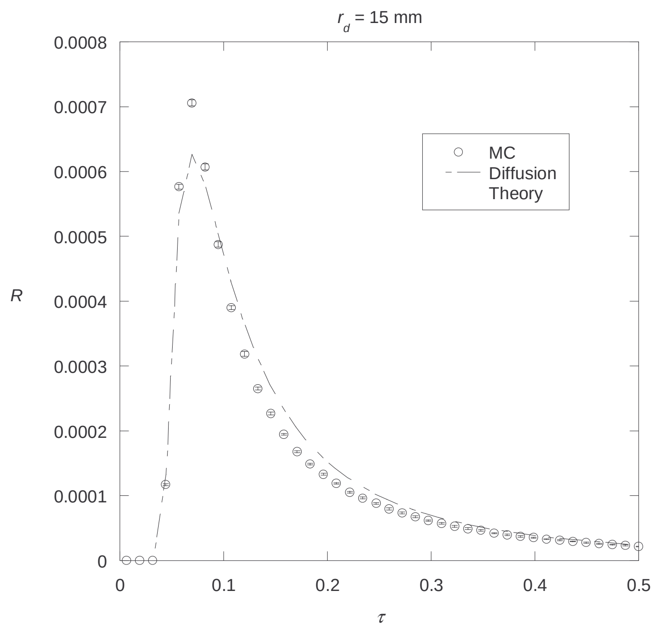


Figure B-4: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #4 ($\mu_s' = 85$ m^{-1}, $L = 0.0655$ m) at a source-detector separation of 15 mm.

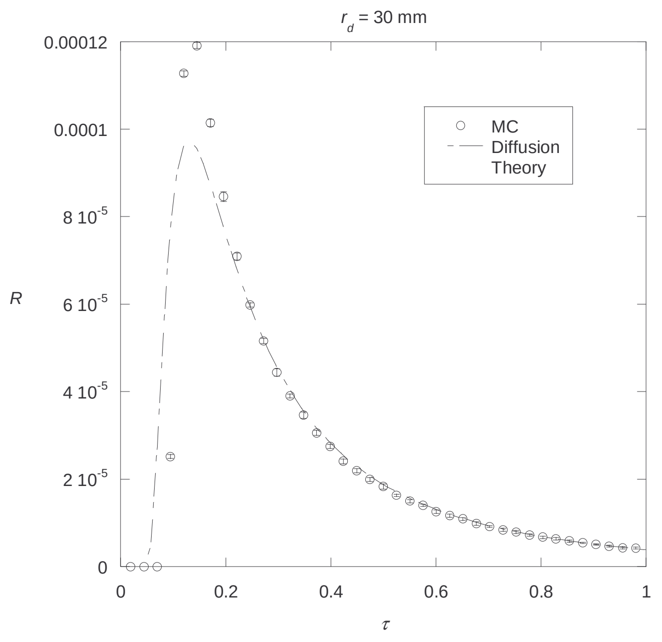


Figure B-5: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #4 ($\mu_s' = 85$ m^{-1}, $L = 0.0655$ m) at a source-detector separation of 30 mm.

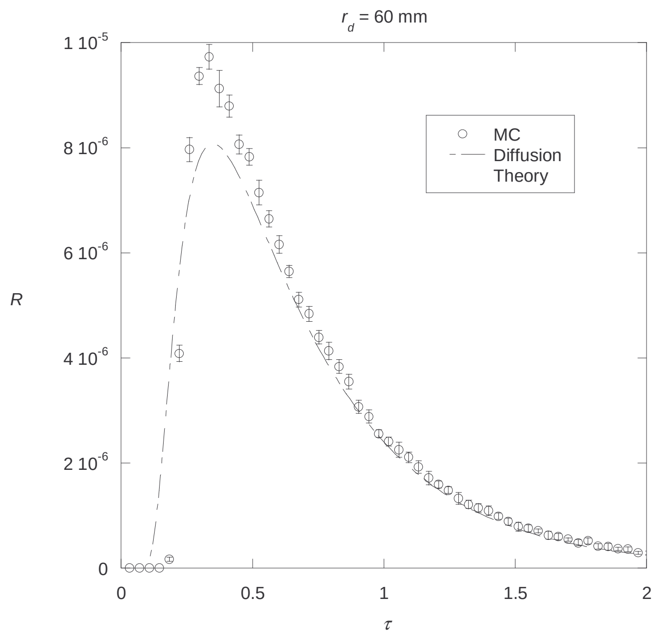


Figure B-6: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #4 ($\mu_s' = 85$ m^{-1}, $L = 0.0655$ m) at a source-detector separation of 60 mm.

<u>*Case 6*</u>

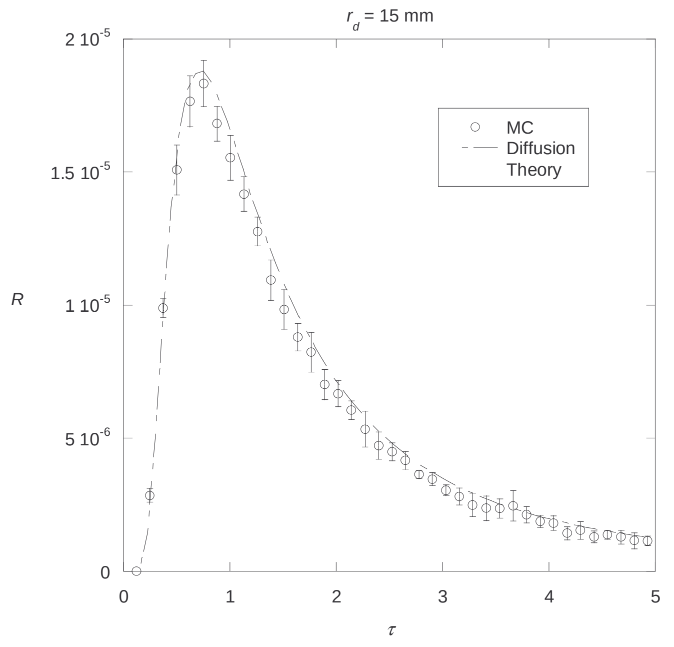


Figure B-7: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #6 ($\mu_s' = 3866$ m^{-1}, $L = 0.1298$ m) at a source-detector separation of 15 mm.

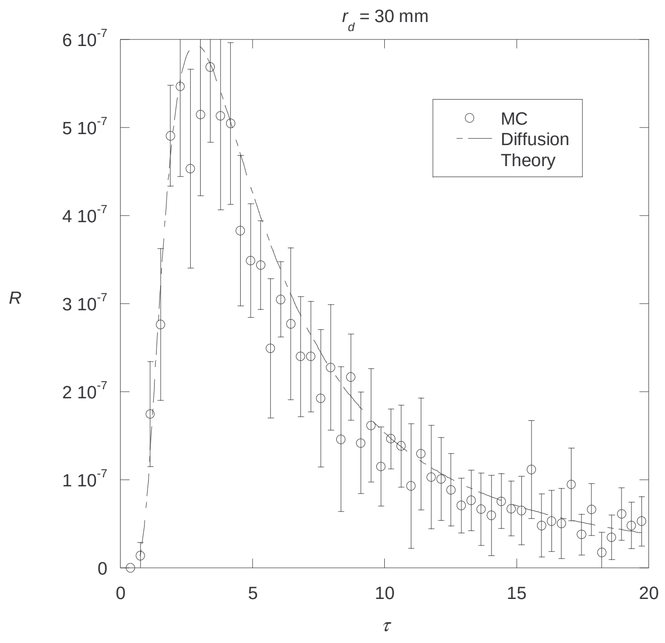


Figure B-8: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #6 ($\mu_s' = 3866$ m^{-1}, $L = 0.1298$ m) at a source-detector separation of 30 mm.

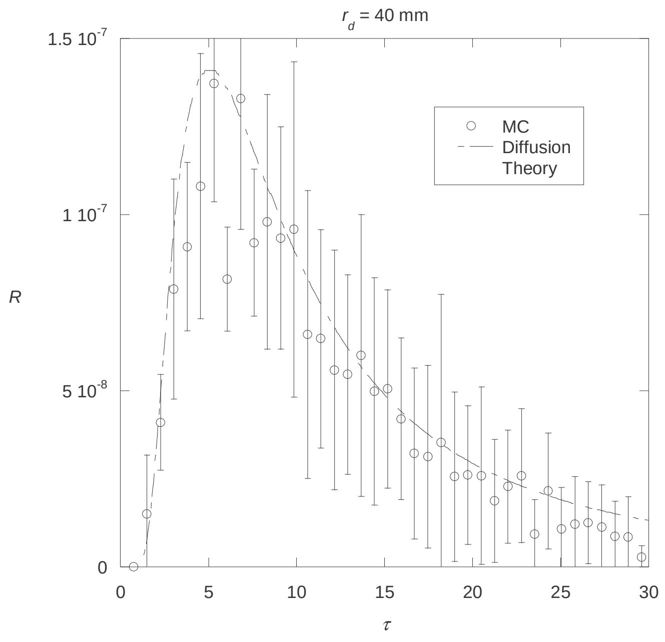


Figure B-9: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #6 ($\mu_s' = 3866$ m^{-1}, $L = 0.1298$ m) at a source-detector separation of 40 mm.

Case 7

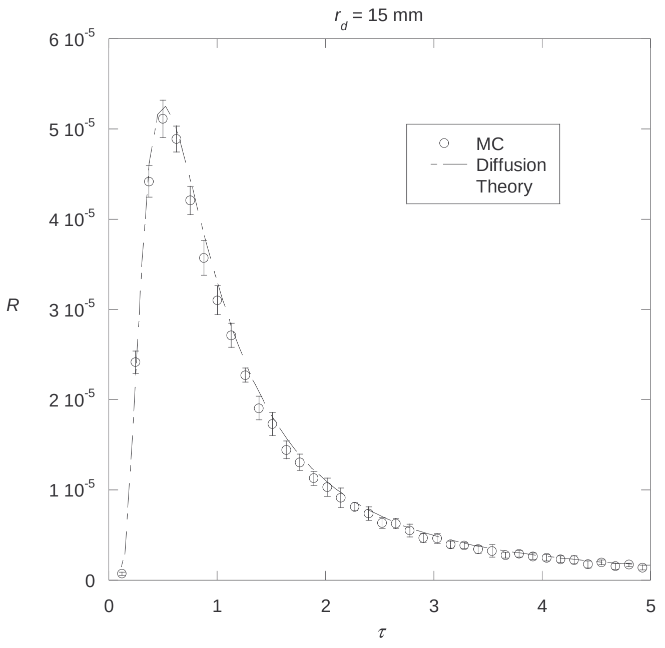


Figure B-10: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #7 ($\mu_s' = 2731$ m^{-1}, $L = 0.1838$ m) at a source-detector separation of 15 mm.

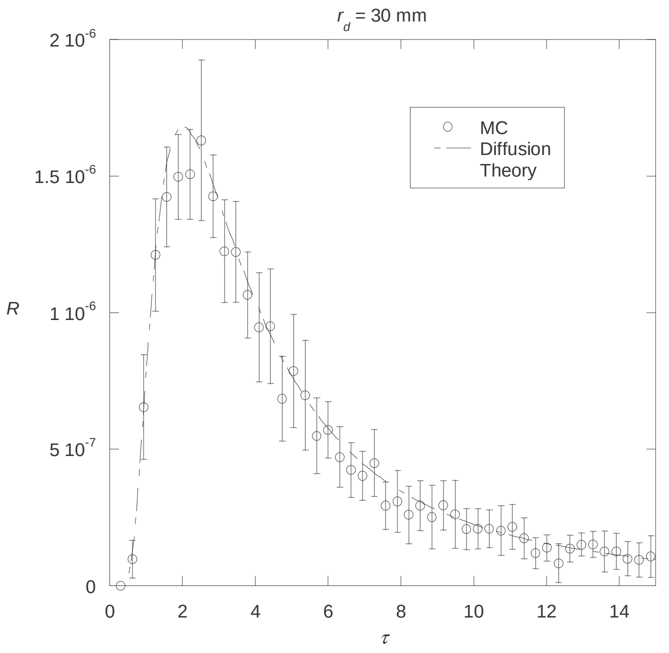


Figure B-11: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #7 ($\mu_s' = 2731$ m^{-1}, $L = 0.1838$ m) at a source-detector separation of 30 mm.

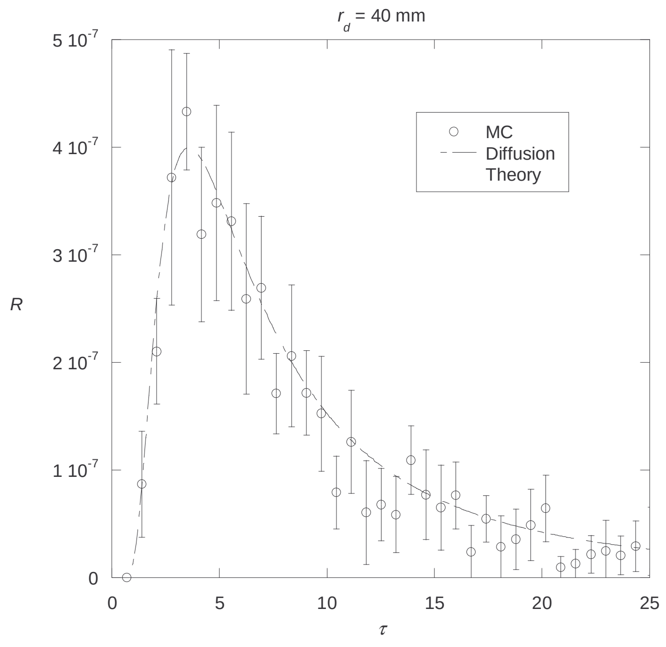


Figure B-12: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #7 ($\mu_s' = 2731$ m^{-1}, $L = 0.1838$ m) at a source-detector separation of 40 mm.

<u>Case 8</u>

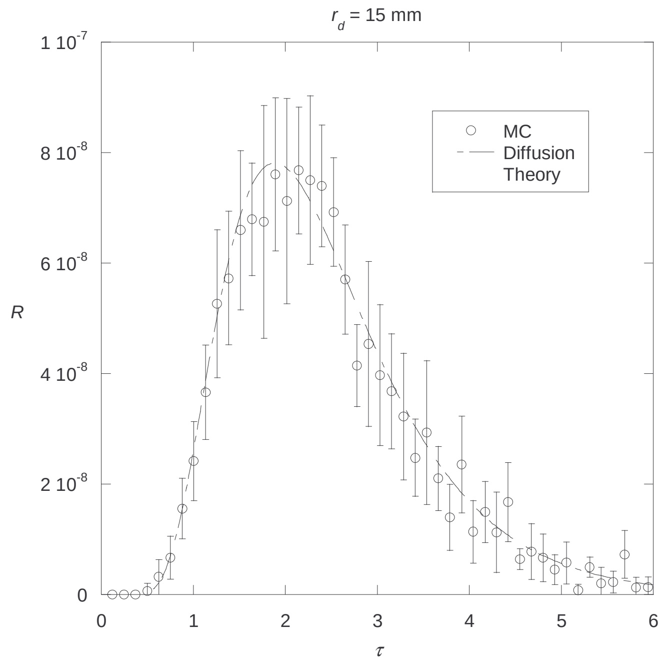


Figure B-13: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #8 ($\mu_s' = 13358$ m^{-1}, $L = 0.0086$ m) at a source-detector separation of 15 mm.

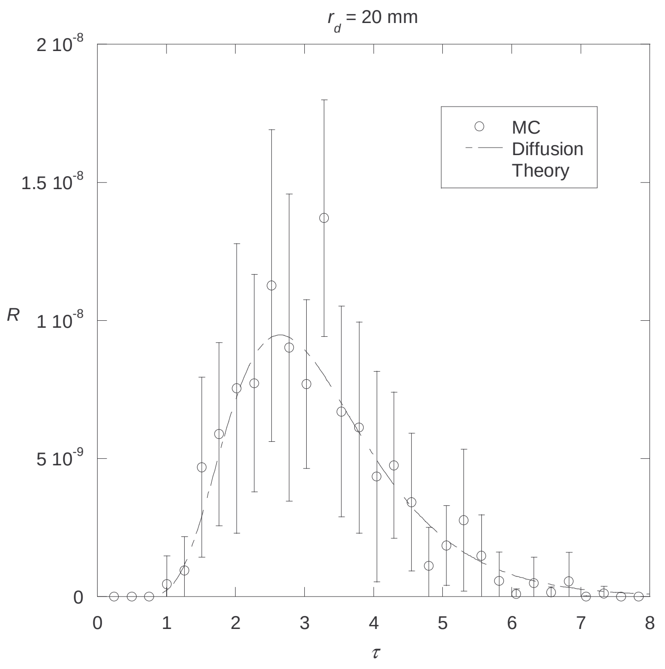


Figure B-14: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #8 ($\mu_s' = 13358$ m^{-1}, $L = 0.0086$ m) at a source-detector separation of 20 mm.

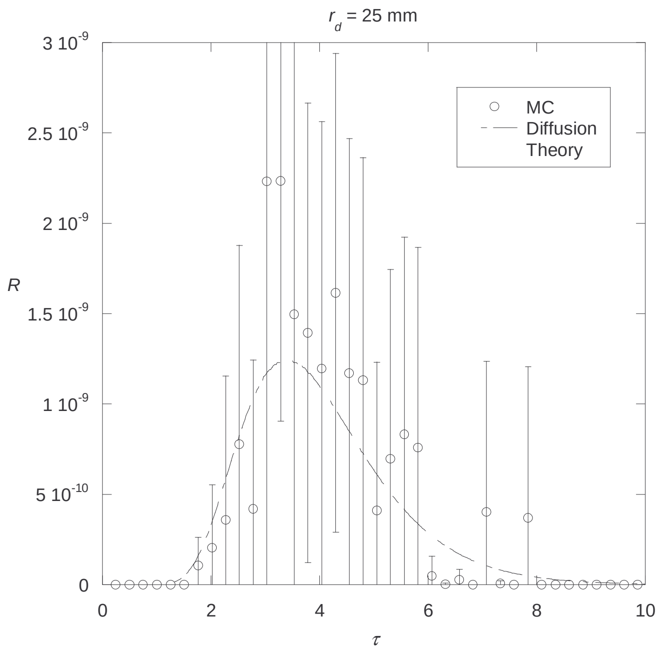


Figure B-15: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #8 ($\mu_s' = 13358$ m^{-1}, $L = 0.0086$ m) at a source-detector separation of 25 mm.

Case 9

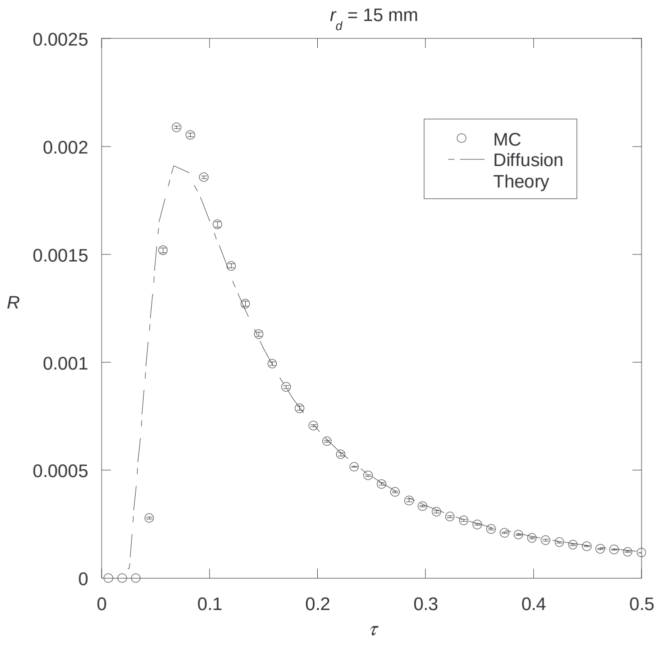


Figure B-16: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #9 ($\mu_s' = 229$ m^{-1}, $L = 0.1803$ m) at a source-detector separation of 15 mm.

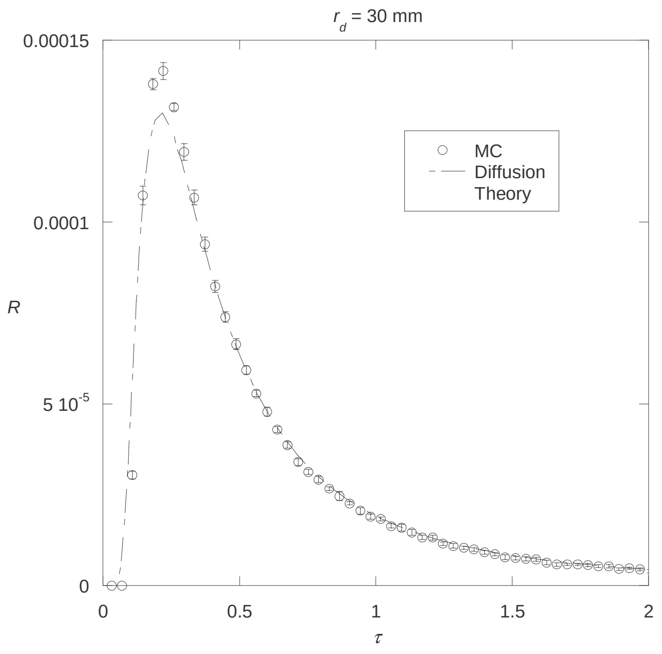


Figure B-17: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #9 ($\mu_s' = 229$ m^{-1}, $L = 0.1803$ m) at a source-detector separation of 30 mm.

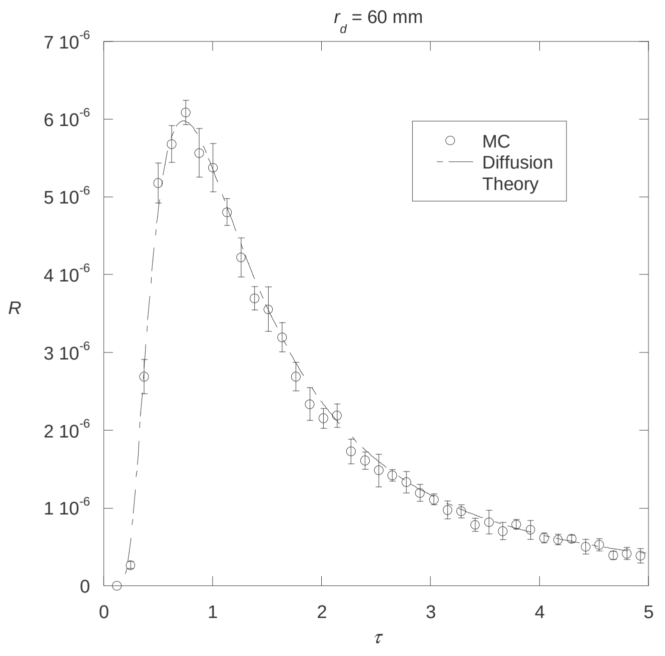


Figure B-18: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #9 ($\mu_s' = 229$ m^{-1}, $L = 0.1803$ m) at a source-detector separation of 60 mm.

Case 10

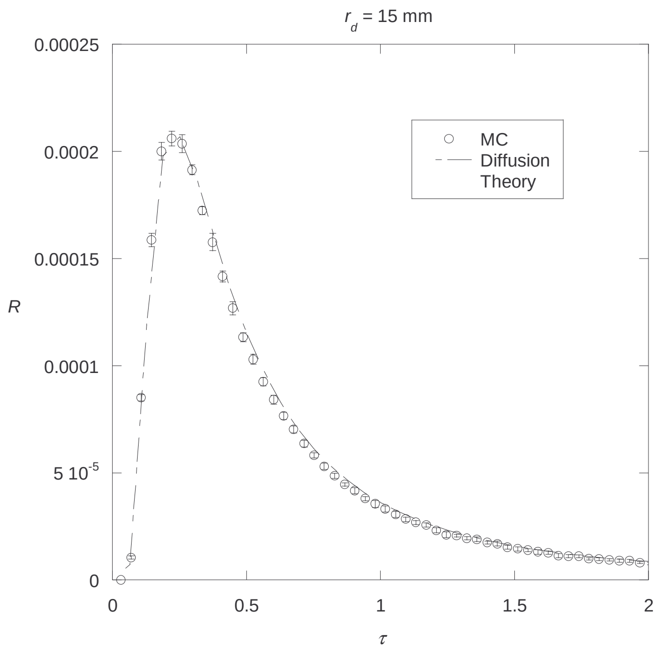


Figure B-19: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #10 ($\mu_s' = 1185$ m^{-1}, $L = 0.1529$ m) at a source-detector separation of 15 mm.

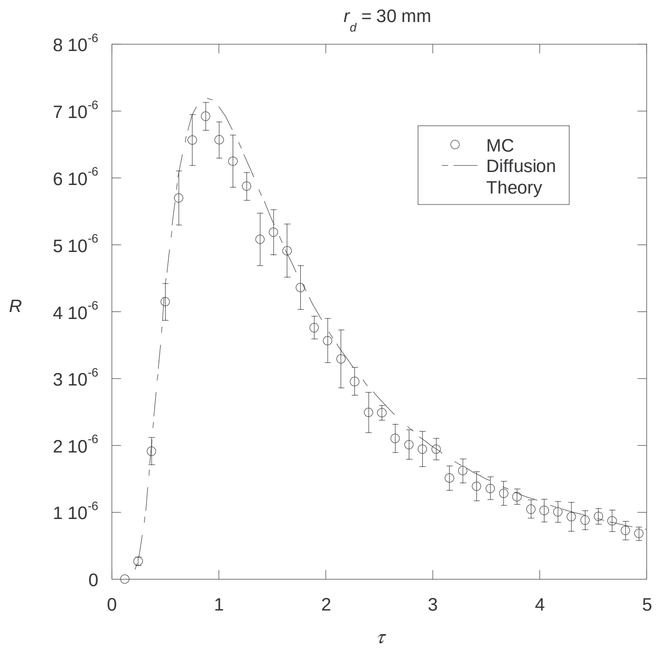


Figure B-20: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #10 ($\mu_s' = 1185$ m^{-1}, $L = 0.1529$ m) at a source-detector separation of 30 mm.

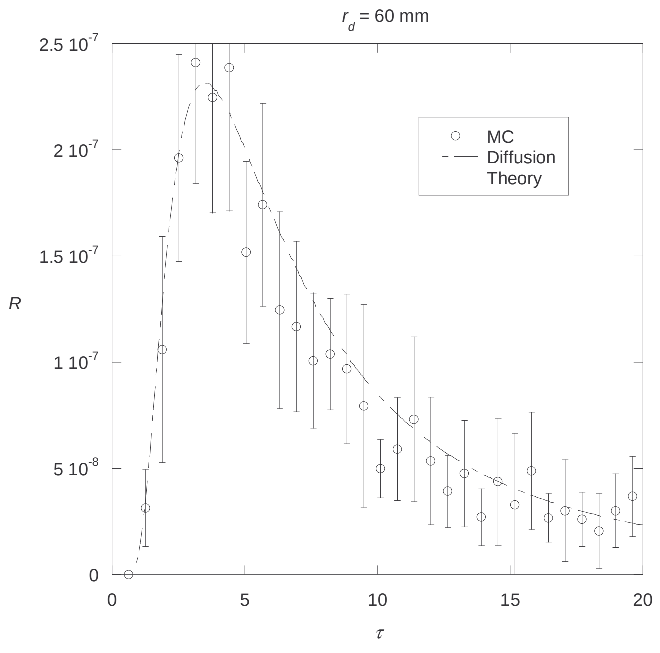


Figure B-21: Time-domain comparison between diffusion theory and Monte Carlo simulations for case #10 ($\mu_s' = 1185$ m^{-1}, $L = 0.1529$ m) at a source-detector separation of 60 mm.

B.2 FREQUENCY-DOMAIN COMPARISONS

Case2

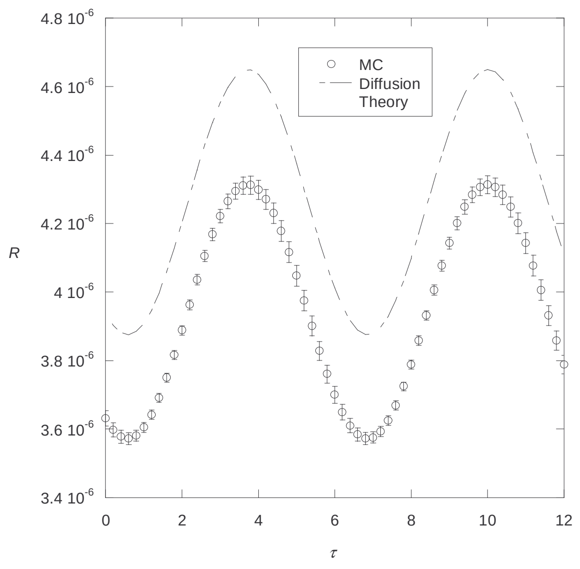
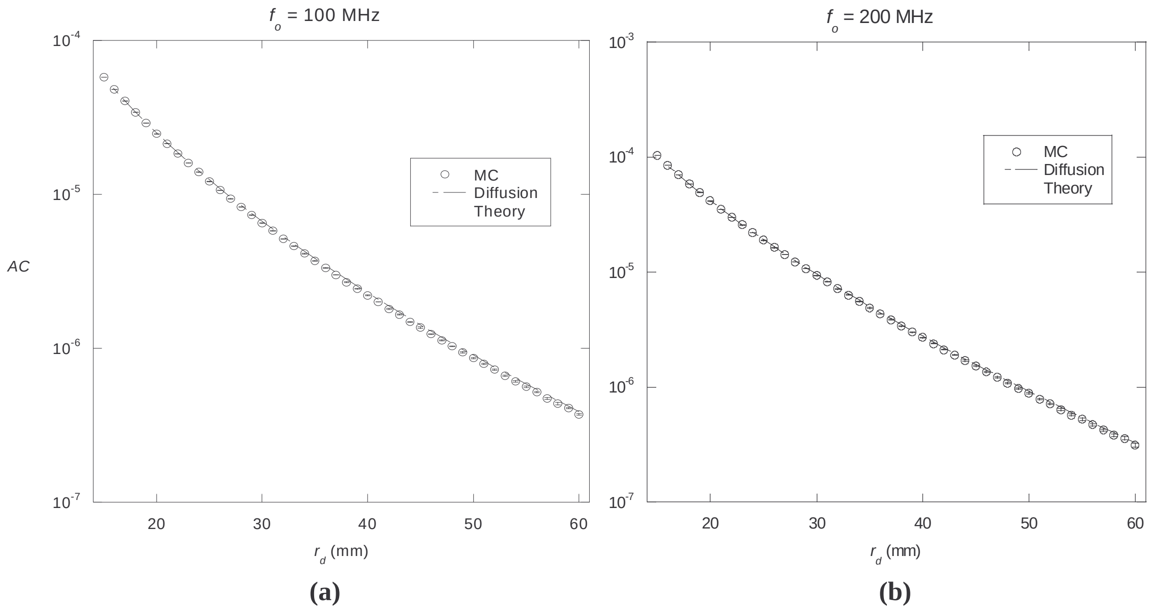
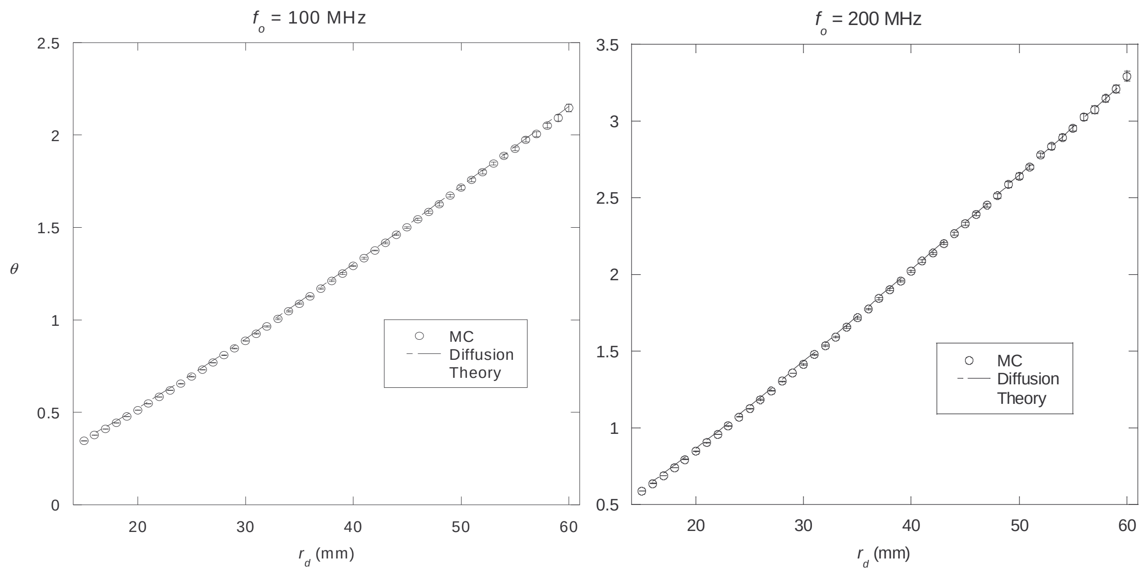


Figure B-22: Frequency-domain comparison between diffusion theory and Monte Carlo simulations for case #2 ($\mu_s' = 517$ m^{-1}, $L = 0.1376$ m) at $r_d = 60$ mm and $f_o = 100$ MHz.



(a) **(b)**

Figure B-23: Comparison between diffusion theory and Monte Carlo simulations of *AC* values for case #2 ($\mu_s' = 517$ m^{-1}, $L = 0.1376$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.



(a) **(b)**

Figure B-24: Comparisons between diffusion theory and Monte Carlo simulations of the phase shift for case #2 ($\mu_s' = 517$ m^{-1}, $L = 0.1376$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

Case 4

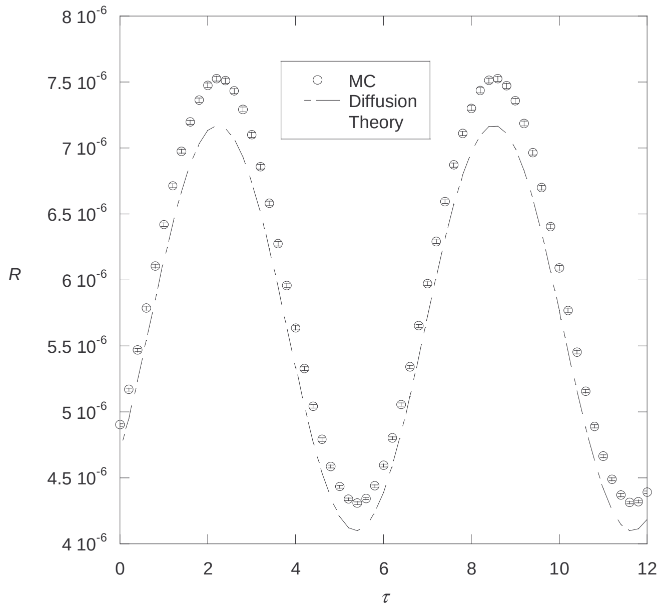


Figure B-25: Frequency-domain comparison between diffusion theory and Monte Carlo simulations for case #4 ($\mu_s' = 85$ m^{-1}, $L = 0.0655$ m) at $r_d = 60$ mm and $f_o = 100$ MHz.

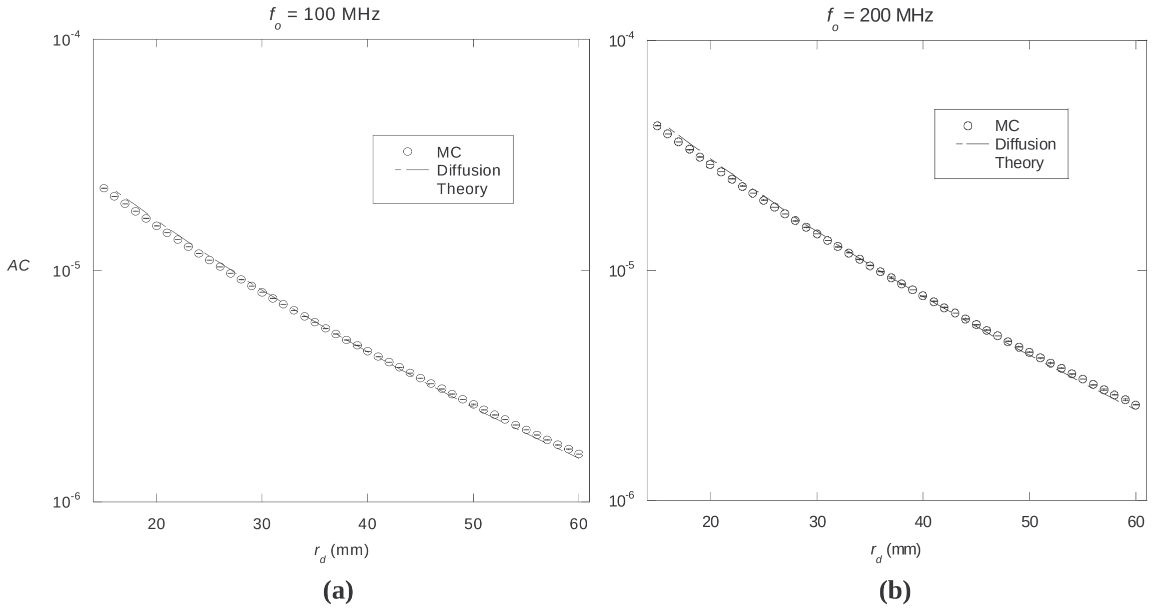


Figure B-26: Comparison between diffusion theory and Monte Carlo simulations of *AC* values for case #4 ($\mu_s' = 85$ m^{-1}, $L = 0.0655$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

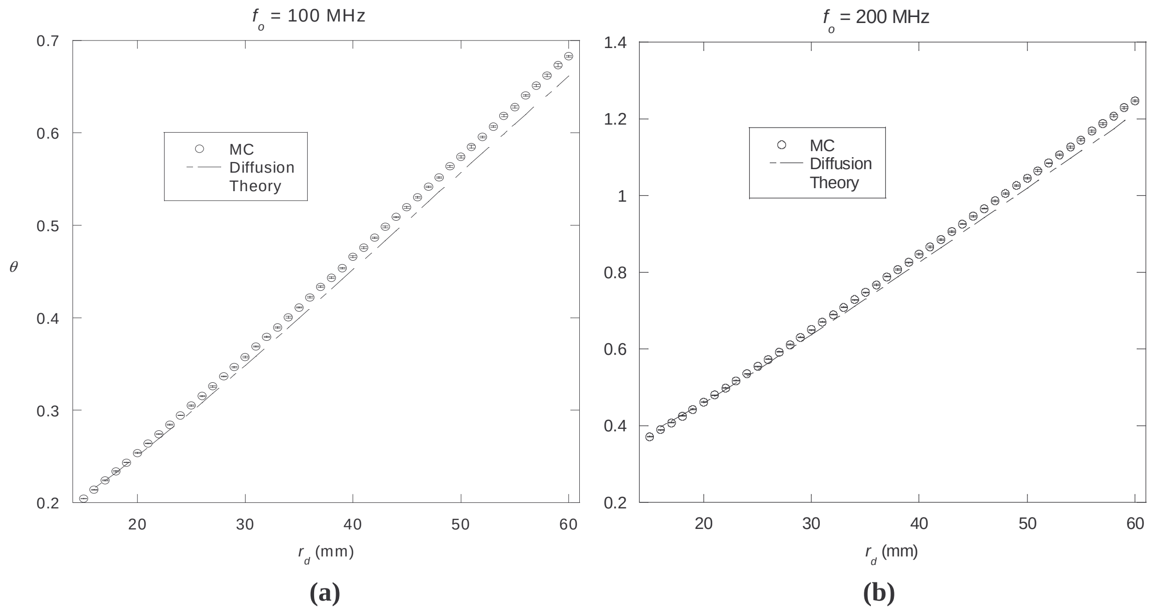


Figure B-27: Comparison of the phase shift between diffusion theory and Monte Carlo simulations for case #4 ($\mu_s' = 85$ m^{-1}, $L = 0.0655$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

Case 6

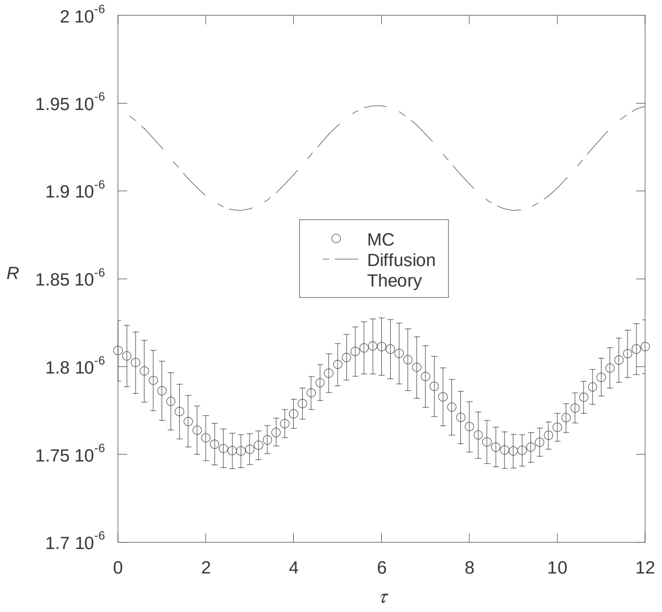
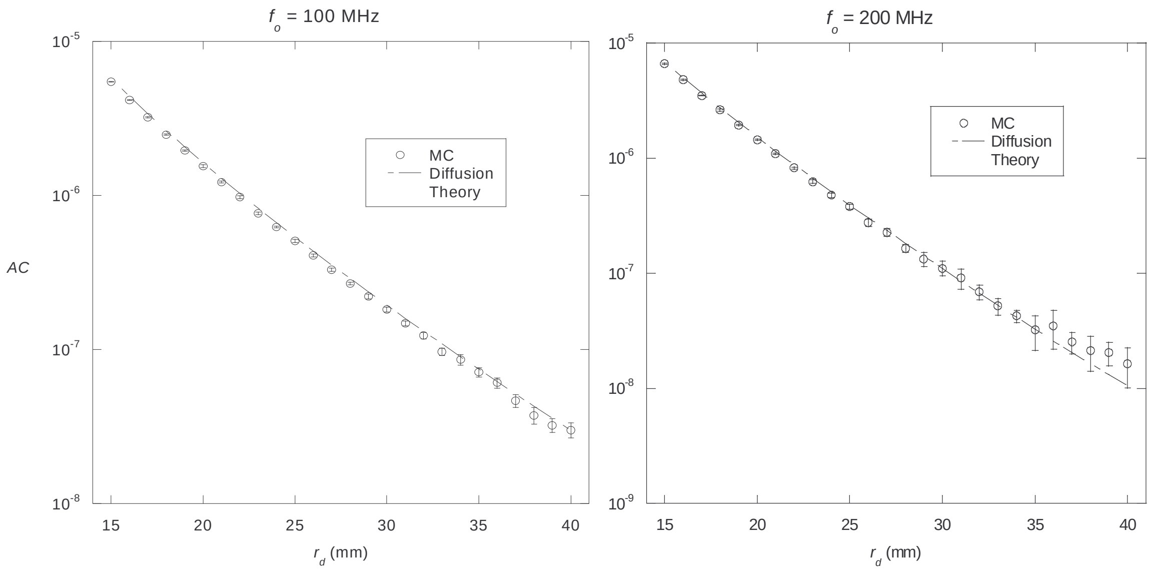


Figure B-28: Frequency-domain comparison between diffusion theory and Monte Carlo simulations for case #6 ($\mu_s' = 3866$ m^{-1}, $L = 0.1298$ m) at $r_d = 40$ mm and $f_o = 100$ MHz.



(a) **(b)**

Figure B-29: Comparison between diffusion theory and Monte Carlo simulations of *AC* values for case #6 ($\mu_s' = 3866$ m^{-1}, $L = 0.1298$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

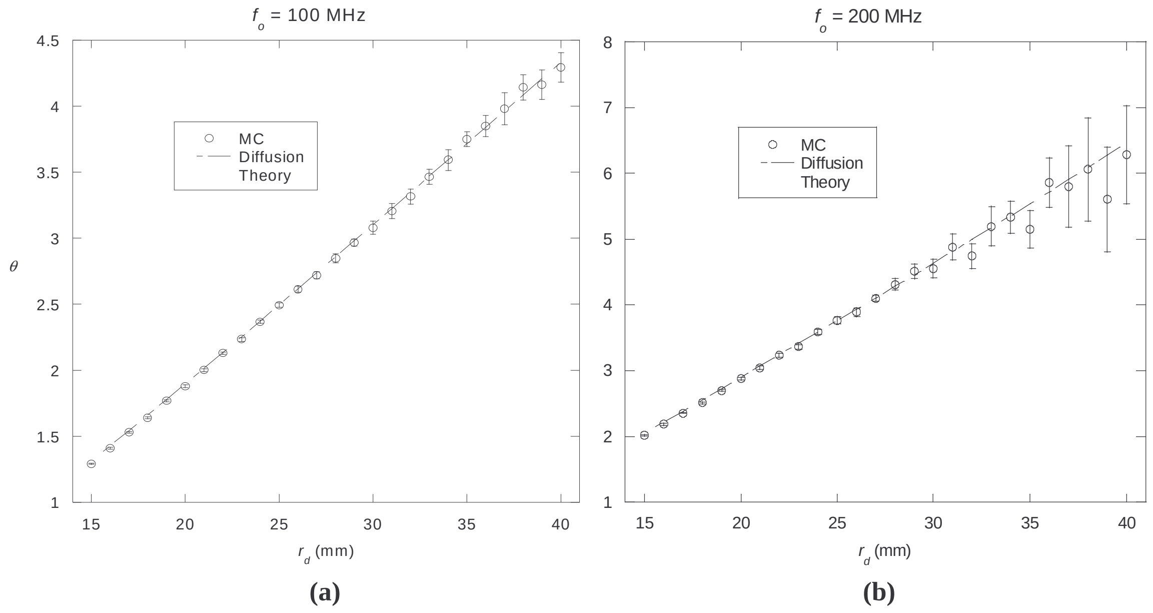


Figure B-30: Comparison of the phase shift between diffusion theory and Monte Carlo simulations for case #6 ($\mu_s' = 3866$ m^{-1}, $L = 0.1298$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

Case 7

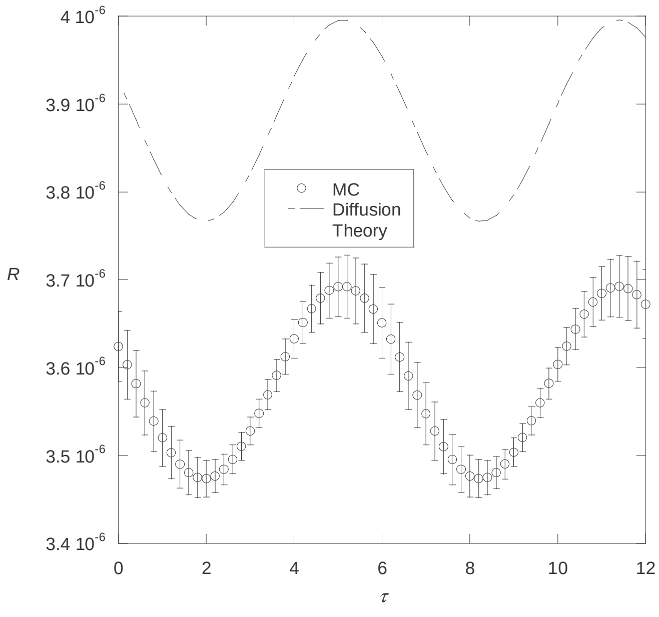


Figure B-31: Frequency-domain comparison between diffusion theory and Monte Carlo simulations for case #7 ($\mu_s' = 2731$ m^{-1}, $L = 0.1838$ m) at $r_d = 40$ mm and $f_o = 100$ MHz.

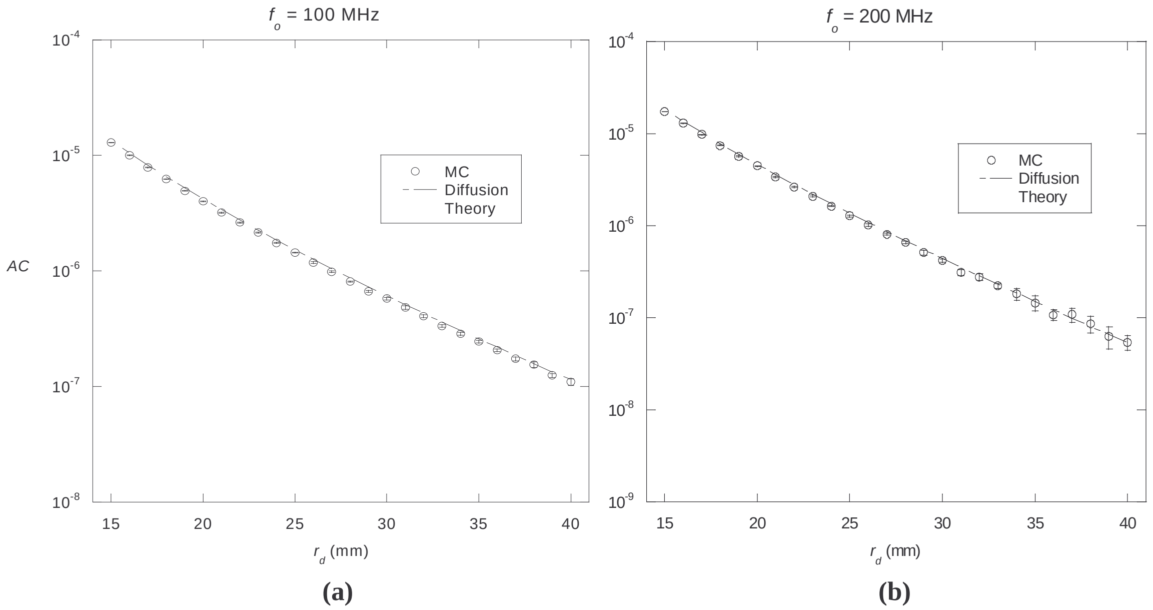


Figure B-32: Comparison between diffusion theory and Monte Carlo simulations of *AC* values for case #7 ($\mu_s' = 2731$ m^{-1}, $L = 0.1838$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

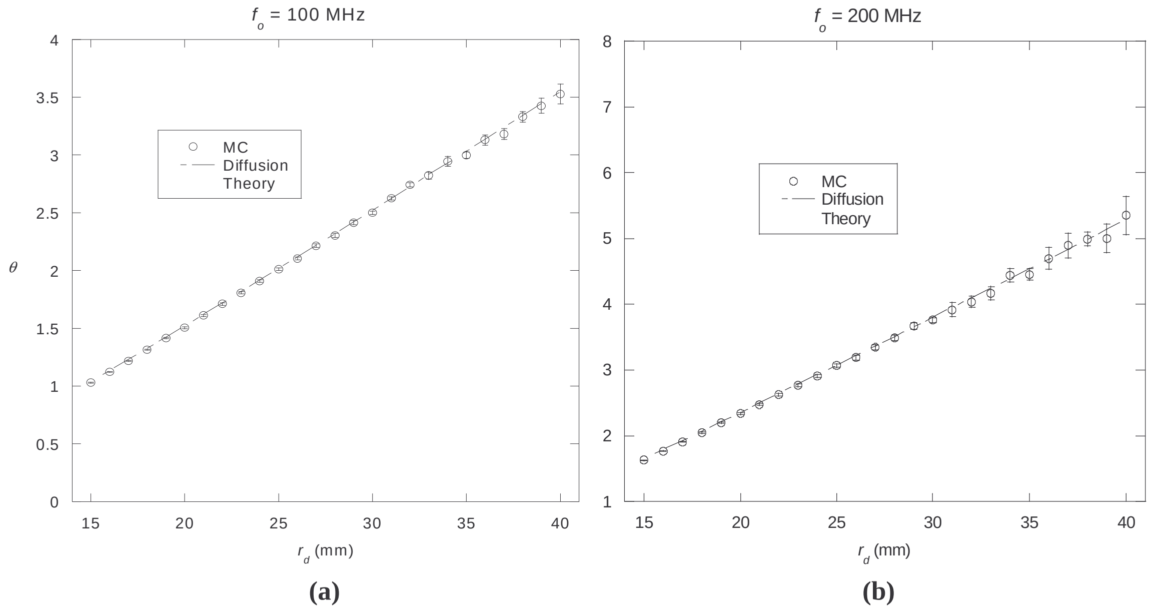


Figure B-33: Comparison of the phase shift between diffusion theory and Monte Carlo simulations for case #7 ($\mu_s' = 2731$ m^{-1}, $L = 0.1838$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

Case 8

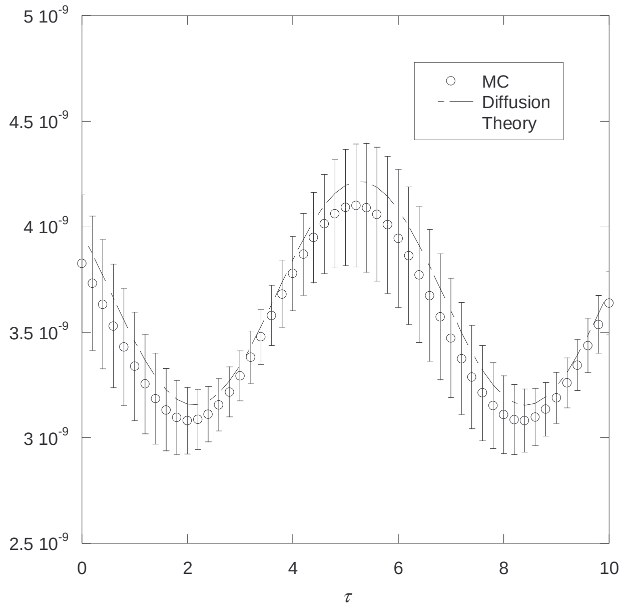
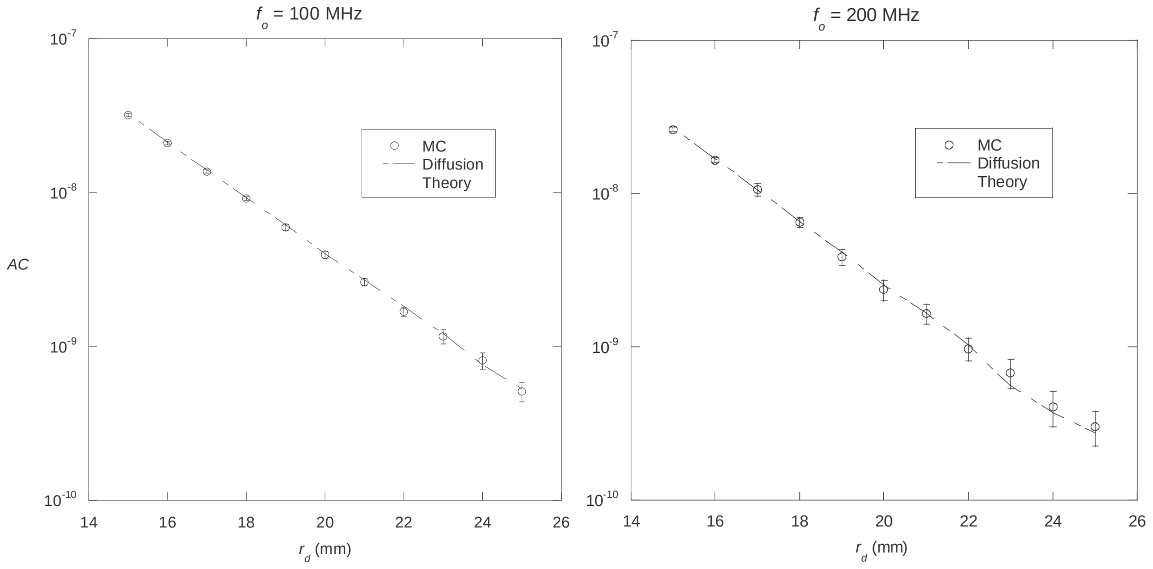
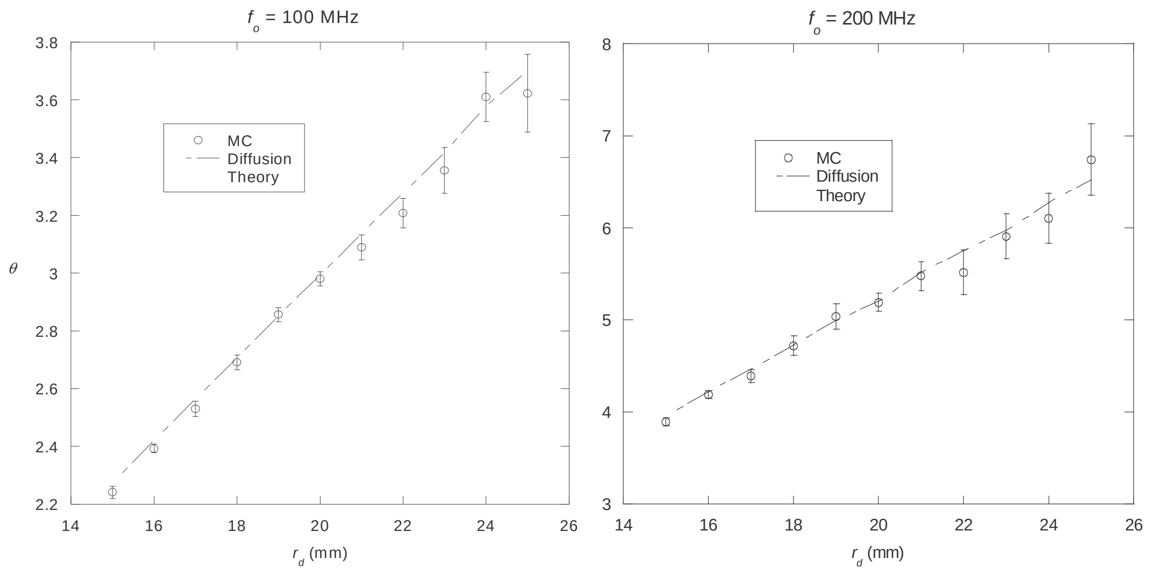


Figure B-34: Frequency-domain comparison between diffusion theory and Monte Carlo simulations for case #8 ($\mu_s' = 13358$ m^{-1}, $L = 0.0086$ m) at $r_d = 25$ mm and $f_o = 100$ MHz.



(a) **(b)**

Figure B-35: Comparison between diffusion theory and Monte Carlo simulations of *AC* values for case #8 ($\mu_s' = 13358$ m^{-1}, $L = 0.0086$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.



(a) **(b)**

Figure B-36: Comparison of the phase shift between diffusion theory and Monte Carlo simulations for case #8 ($\mu_s' = 13358$ m^{-1}, $L = 0.0086$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

<u>*Case 9*</u>

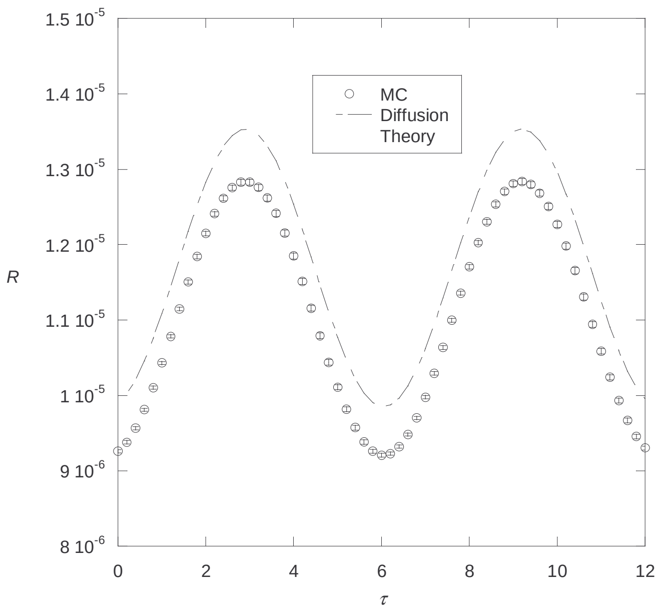
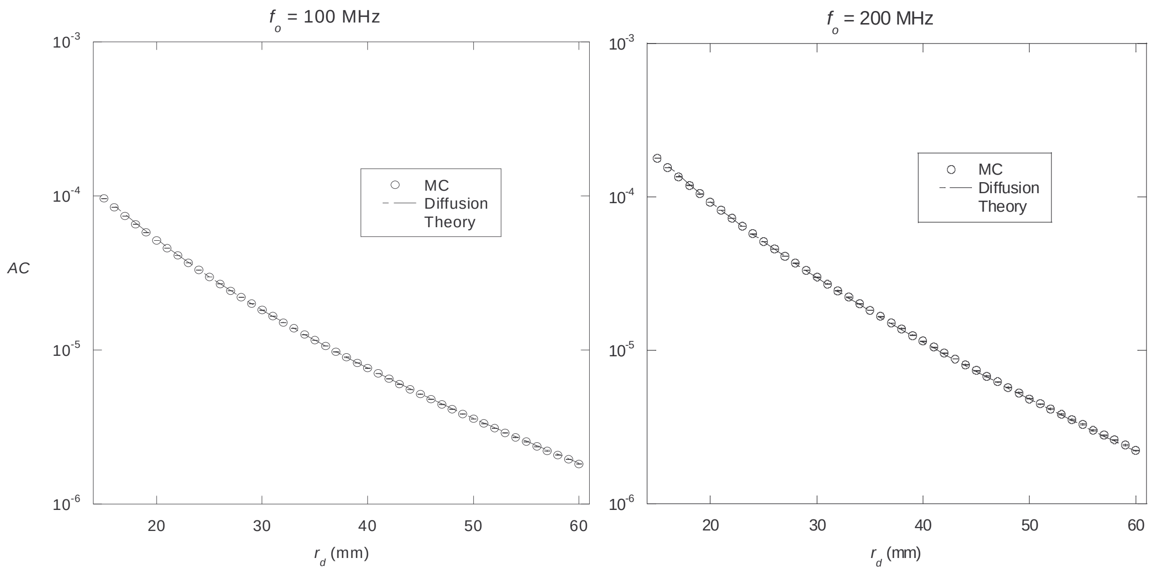
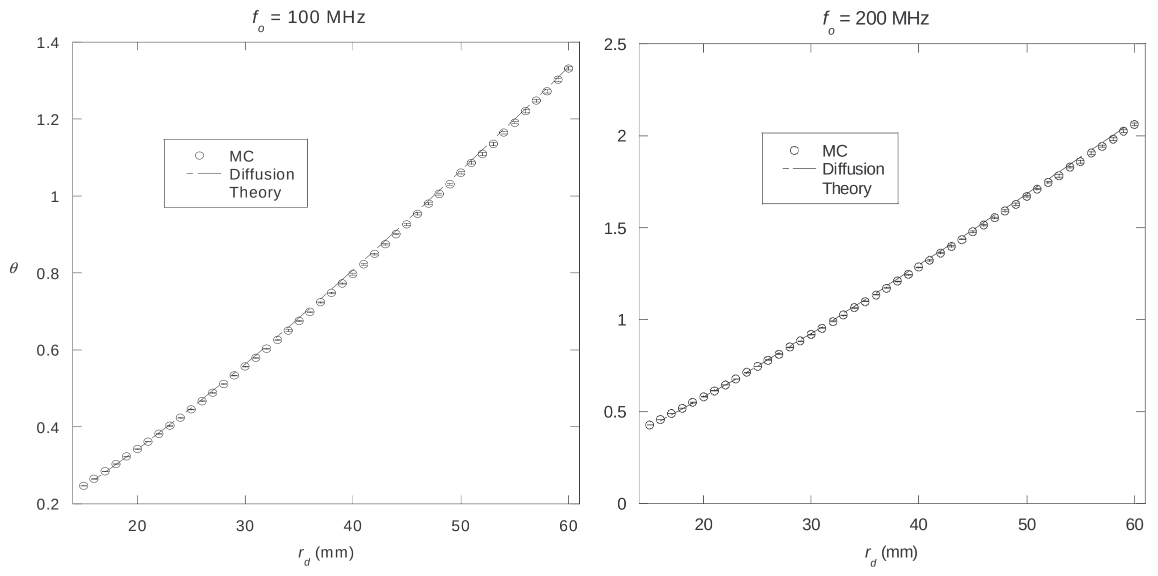


Figure B-37: Frequency-domain comparison between diffusion theory and Monte Carlo simulations for case #9 ($\mu_s' = 229$ m^{-1}, $L = 0.1803$ m) at $r_d = 60$ mm and $f_o = 100$ MHz.



(a) **(b)**

Figure B-38: Comparison between diffusion theory and Monte Carlo simulations of *AC* values for case #9 ($\mu_s' = 229$ m^{-1}, $L = 0.1803$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.



(a) **(b)**

Figure B-39: Comparison of the phase shift between diffusion theory and Monte Carlo simulations for case #9 ($\mu_s' = 229$ m^{-1}, $L = 0.1803$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

Case 10

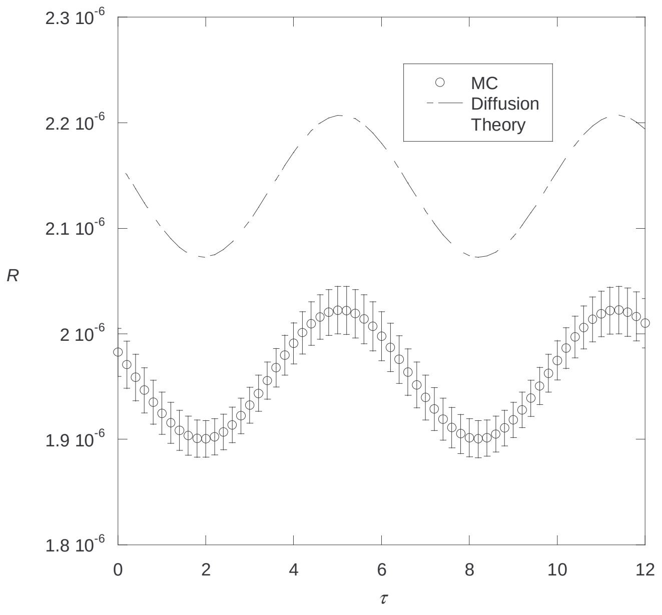
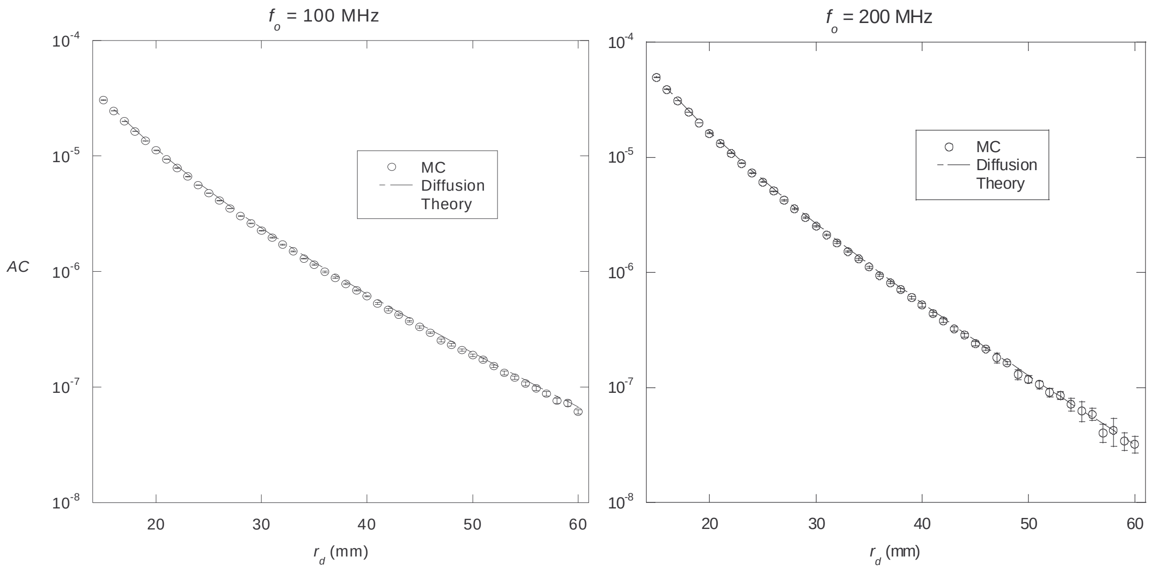
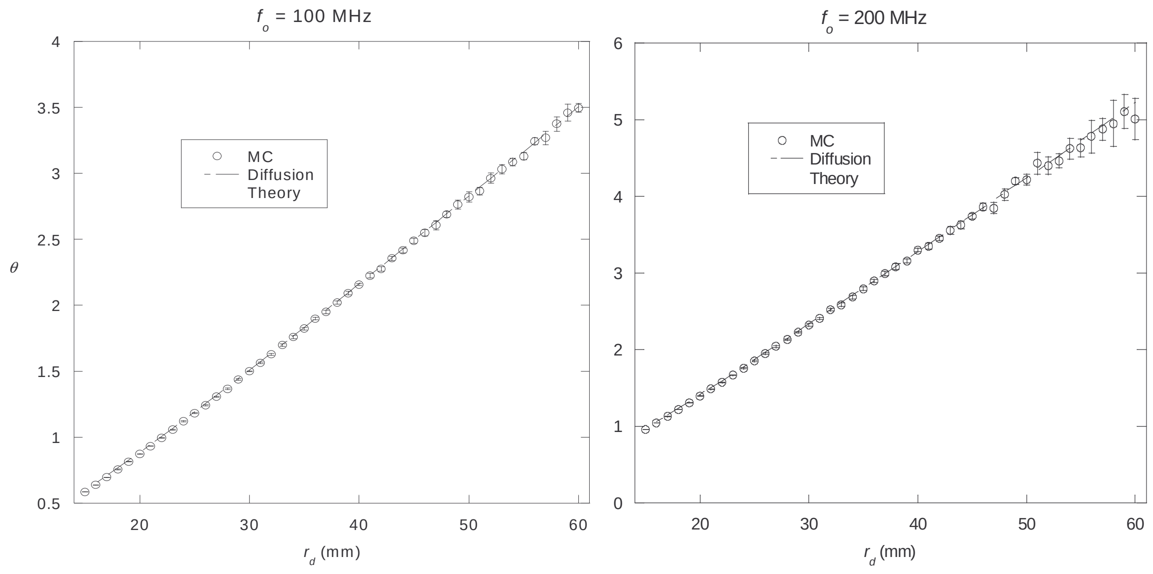


Figure B-40: Frequency-domain comparison between diffusion theory and Monte Carlo simulations for case #10 ($\mu_s' = 1185$ m^{-1}, $L = 0.1529$ m) at $r_d = 60$ mm and $f_o = 100$ MHz.



(a) **(b)**

Figure B-41: Comparison between diffusion theory and Monte Carlo simulations of *AC* values for case #10 ($\mu_s' = 1185$ m^{-1}, $L = 0.1529$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.



(a) (b)

Figure B-42: Comparison of the phase shift between diffusion theory and Monte Carlo simulations for case #10 ($\mu_s' = 1185$ m^{-1}, $L = 0.1529$ m) at (a) $f_o = 100$ MHz, (b) $f_o = 200$ MHz.

B.3 Inversion Algorithm Convergence Histories

Case 2

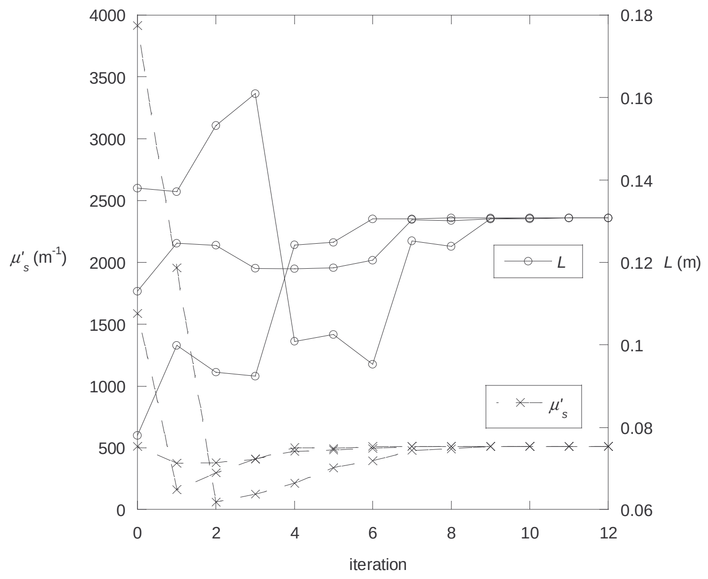


Figure B-43: Iteration history of the inverse algorithm for case #2.

Case 4

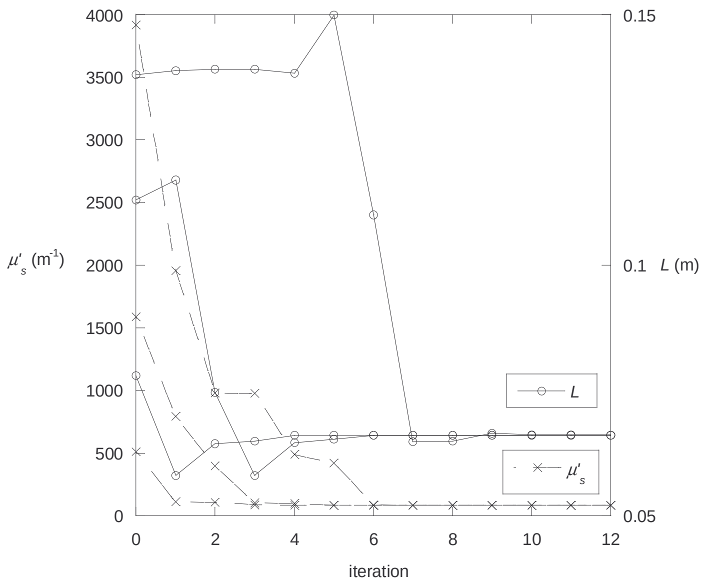


Figure B-44: Iteration history of the inverse algorithm for case #4.

Case 6

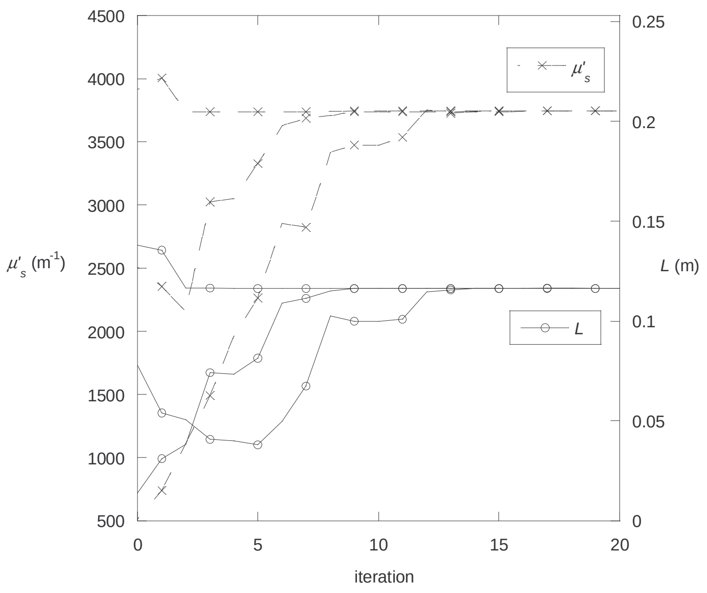


Figure B-45: Iteration history of the inverse algorithm for case #6.

Case 7

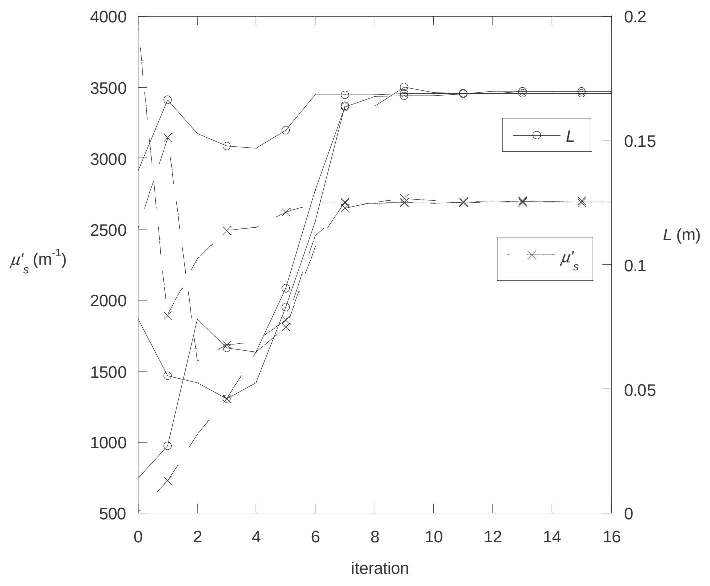


Figure B-46: Iteration history of the inverse algorithm for case #7.

Case 8

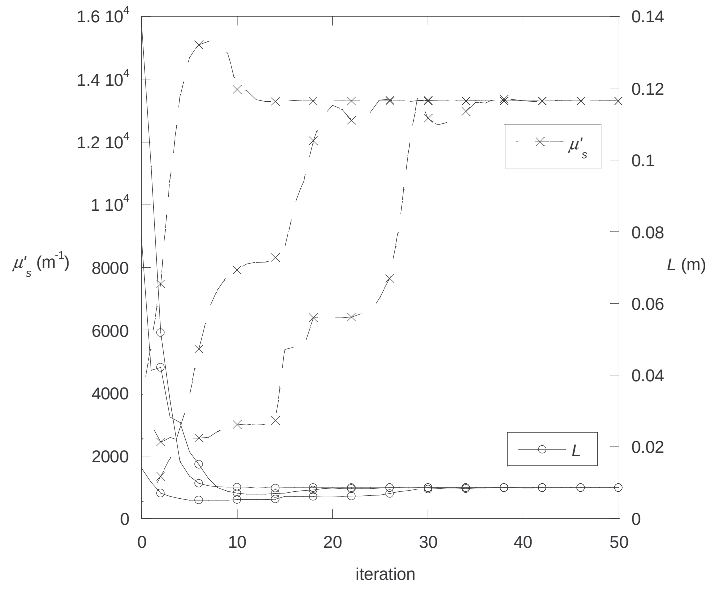


Figure B-47: Iteration history of the inverse algorithm for case #8.

Case 9

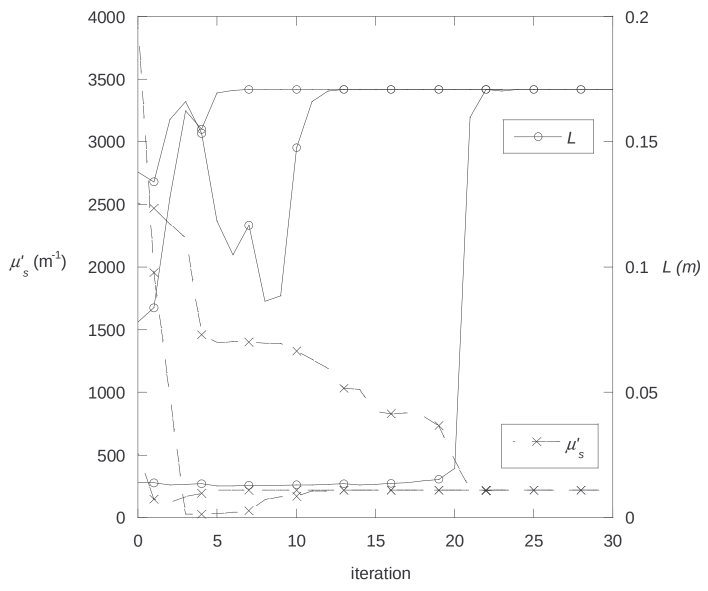


Figure B-48: Iteration history of the inverse algorithm for case #9.

Case 10

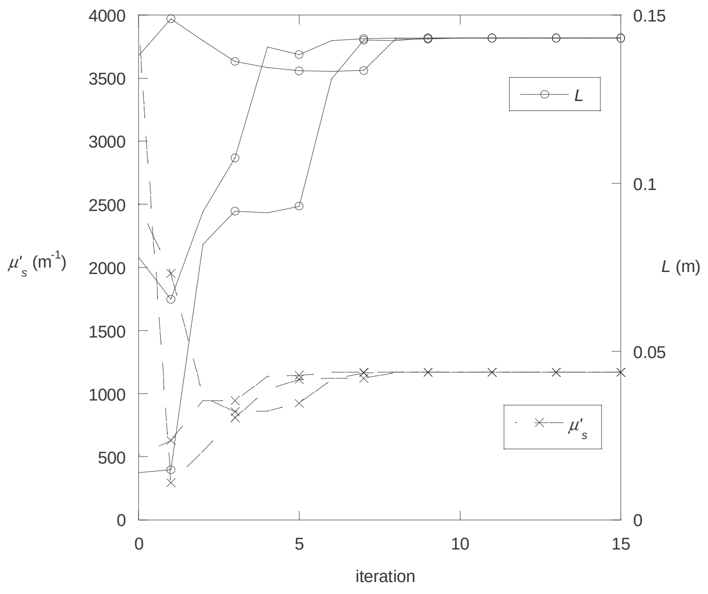


Figure B-49: Iteration history of the inverse algorithm for case #10.

APPENDIX C– DERIVATION OF MONTE CARLO CONVERSION FROM TIME-DOMAIN TO FREQUENCY DOMAIN

The Monte Carlo method is treated as a system that converts the input (the time-dependent portion of the source) into the output (the time-dependent reflectance profile). The simulations in this research were generated for a pulsed source representation. However, frequency-domain measurements are necessary for the inverse problem. The following shows the derivation of the equations necessary to convert time-domain measurements into frequency-domain measurements. Consider the Fourier transform.

$$\hat{F}(f) = \int_{-\infty}^{\infty} v(\tau) e^{i\frac{f}{f_O}\tau} d\tau = F\left\{v(\tau)\right\} \tag{C.1}$$

The transfer function for this system is defined as

$$z(f) = \frac{\hat{R}(f)}{\hat{F}(f)} \tag{C.2}$$

where $\hat{R}(f)$ is the Fourier transform of the reflectance profile and is defined below.

$$\hat{R}(f) = F\left\{R(\tau)\right\} = \int_{-\infty}^{\infty} R(\tau) e^{i\frac{f}{f_O}\tau} d\tau \tag{C.3}$$

To find $z(f)$, let $v^*(\tau) = \delta(\tau)$ be the time-dependent portion of the source and $R^*(\tau)$ be the corresponding reflectance profile. $R^*(\tau)$ is obtained from the Monte Carlo simulation with a pulse input and $\hat{R}^*(f)$ is the corresponding Fourier transform of the reflectance. However, because $R^*(\tau)$ contains data at discrete time intervals, it is convenient to calculate $\hat{R}^*(f)$ through the use of the FFT. Equation (C.1) can be used to evaluate the Fourier transform of the input $v^*(\tau)$.

$$\hat{F}^*(f) = \int_{-\infty}^{\infty} \delta(\tau) e^{i\frac{f}{f_o}\tau} d\tau = 1 \tag{C.4}$$

The transfer function simply becomes $z(f) = \hat{R}^*(f)$ and can be used for any arbitrary input $v(\tau)$ to obtain the corresponding reflectance profile. Solving Eq. (C.2) for $\hat{R}(f)$ and substituting the result of the transfer function, the Fourier transformed reflectance profile for an arbitrary source is given by

$$\hat{R}(f) = \hat{R}^*(f)\hat{F}(f) \tag{C.5}$$

Applying the inverse Fourier transform to Eq. (C.5) results in a reflectance profile corresponding to the respective input $v(\tau)$.

$$R(\tau) = \int_{-\infty}^{\infty} \hat{R}^*(f)\hat{F}(f) e^{-i\frac{f}{f_o}\tau} d\tau \tag{C.6}$$

Consider the case of a sinusoidal modulated source input $v(\tau) = 1 + A\sin\tau$.

$$\hat{F}(f) = \int_{-\infty}^{\infty} (1 + A\sin\tau) e^{i\frac{f}{f_o}\tau} d\tau \tag{C.7}$$

Evaluation of Eq. (C.7) requires a definition of the exponential portion of the integral [32].

$$\delta(x-\xi) = \int_{-\infty}^{\infty} e^{\pm i2\pi f(x-\xi)} df \tag{C.8}$$

Applying the definition of Eq. (C.8) and recalling the complex form of the sine function, $\sin\tau = \left(e^{i\tau} - e^{-i\tau}\right)\Big/2i$ Eq. (C.7) can be evaluated re-written in the following form

$$\hat{F}(f) = \int_{-\infty}^{\infty} e^{-i\frac{f}{f_o}\tau}\, d\tau + A\int_{-\infty}^{\infty} \frac{e^{i\tau} - e^{-i\tau}}{2i} e^{-i\frac{f}{f_o}\tau}\, d\tau \tag{C.9}$$

This equation can now be evaluated to yield the following.

$$\hat{F}(f) = \delta\left(\frac{f}{2\pi f_o}\right) - \frac{Ai}{2}\delta\left(\frac{f}{2\pi f_o} + \frac{1}{2\pi}\right) + \frac{Ai}{2}\delta\left(\frac{f}{2\pi f_o} - \frac{1}{2\pi}\right) \tag{C.10}$$

Substituting Eq. (C.10) into Eq. (C.6) and evaluating the integral yields

$$\tilde{R}(\tau) = \hat{R}^*(0) - \frac{Ai}{2}\left[\hat{R}^*(-f_o)e^{i\tau} - \hat{R}^*(f_o)e^{-i\tau}\right] \tag{C.11}$$

Considering only the bracketed term and putting the complex exponential terms into their sine and cosine counterparts allows for some simplification.

$$\tilde{R}_b(\tau) = \hat{R}^*(-f_o)[\cos\tau + i\sin\tau] - \hat{R}^*(f_o)\left[\cos(-\tau) + i\sin(-\tau)\right] \tag{C.12}$$

Recalling the odd and even function properties of periodic functions allows for some further simplification.

$$\begin{aligned}\cos(-\tau) &= \cos\tau \\ \sin(-\tau) &= -\sin\tau\end{aligned} \tag{C.13}$$

Using these properties, Eq. (C.12) can be re-written in the following form.

$$\tilde{R}_b(\tau) = \hat{R}^*(-f_o)[\cos\tau + i\sin\tau] - \hat{R}^*(f_o)[\cos\tau - i\sin\tau] \tag{C.14}$$

It is important to note that $\hat{R}^*$, obtained from the FFT, is a complex number and therefore contains both real and imaginary components as a function of frequency. At this point it is necessary to introduce some notation that will be used to simplify Eq. (C.14).

$$\hat{R}^*(f_o) = \mathrm{Re}(f_o) + i\,\mathrm{Im}(f_o) \tag{C.15}$$

Re() and Im() are used to simply represent the real and imaginary components respectively of the complex number at the specified frequencies. The following properties of the FFT allow for simplification of the previous equations.

$$\begin{aligned} &\mathrm{Re}(-f_o) = \mathrm{Re}(f_o) \\ &\mathrm{Im}(-f_o) = -\mathrm{Im}(f_o) \\ &\mathrm{Im}(0) = 0 \end{aligned} \tag{C.16}$$

Substituting this notation into Eq. (C.14) results in

$$\begin{aligned} \tilde{R}_b(\tau) = &\left[\mathrm{Re}(f_o) - i\,\mathrm{Im}(f_o)\right]\cos\tau + \left[\mathrm{Re}(f_o) - i\,\mathrm{Im}(f_o)\right]\sin \\ &-\left[\mathrm{Re}(f_o) + i\,\mathrm{Im}(f_o)\right]\cos\tau + \left[\mathrm{Re}(f_o) + i\,\mathrm{Im}(f_o)\right]\sin\tau \end{aligned} \tag{C.17}$$

Grouping like terms of Eq. (C.17) and putting the results back into the bracketed portion of Eq. (C.11) results in the following.

$$\tilde{R}(\tau) = \mathrm{Re}(0) - \frac{Ai}{2}\left[2i\,\mathrm{Im}(f_o)\cos\tau + 2i\,\mathrm{Re}(f_o)\sin\tau\right] \tag{C.18}$$

Distributing the terms yields

$$\tilde{R}(\tau) = \mathrm{Re}(0) - A\,\mathrm{Im}(f_o)\cos\tau + A\,\mathrm{Re}(f_o)\sin\tau \tag{C.19}$$

Although this equation is in a simplified form, some manipulation can be done to put it in a similar form to that resulting from diffusion theory, thereby making it simple to compare measurements. Eq. (C.19) can now be written in a different form.

$$\tilde{R}(\tau) = \mathrm{Re}(0) + A\sqrt{\mathrm{Re}(f_o)^2 + \mathrm{Im}(f_o)^2} \left[\frac{\mathrm{Re}(f_o)}{\sqrt{\mathrm{Re}(f_o)^2 + \mathrm{Im}(f_o)^2}} \sin\tau - \frac{\mathrm{Im}(f_o)}{\sqrt{\mathrm{Re}(f_o)^2 + \mathrm{Im}(f_o)^2}} \cos\tau \right] \tag{C.20}$$

These terms can be simplified by considering a right triangle that contains an angle θ with an opposite side of *Im(f$_o$)* and adjacent side of *Re(f$_o$)*.

$$\tilde{R}(\tau) = \mathrm{Re}(0) + A\sqrt{\mathrm{Re}(f_o)^2 + \mathrm{Im}(f_o)^2} \left[\sin\tau\cos\theta - \cos\tau\sin\theta \right] \tag{C.21}$$

Using trigonometric identities Eq. (C.21) can be written in its final form which corresponds to a sinusoidal input.

$$\tilde{R}(\tau) = \mathrm{Re}(0) + A\sqrt{\mathrm{Re}(f_o)^2 + \mathrm{Im}(f_o)^2} \sin(\tau - \theta) \tag{C.22}$$

Where θ represents the phase shift between the input and the output and is defined by the following equation.

$$\theta = \tan^{-1}\left[\frac{\mathrm{Im}(f_o)}{\mathrm{Re}(f_o)} \right] \tag{C.23}$$

APPENDIX D– SOURCE CODES USED IN THIS RESEARCH

D.1 MONTE CARLO SIMULATION SOURCE CODE INCLUDING THE FAST FOURIER TRANSFORM

Source code file:	fmc.C
Compile instructions:	use the following command in a UNIX command window in the directory where fmc.C is located g++ fmc.C –o *desirednameofexecutable* –lm -O run executable by typing "./" then name of executable file, then press "Enter."
User Inputs:	values defining the layer properties and measurement parameters
Program Output Files:	reflect.txt – reflectance profile at various detector locations FFT.txt – The fast Fourier transformed data meas.txt – calculated measurements from FFT data

Source Code:

```
/* This is a Monte Carlo simulation code. It simulates non-dimensional reflectance measurements in the
time-domain some distance from a laser source. Corresponding frequency-domain measurements are
calculated by applying a Fast Fourier Transform to the time-domain data. */

#include <iomanip>
#include <fstream>
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
using namespace std;

#define np 1.0E7              // number of photons
#define g 0.75                // asymmetry parameter
#define mua 0                 // absorption coefficient (m^-1)
#define mus 8332              // scattering coefficient (m^-1)
#define mu (mua+mus)          // attenuation coefficient (m^-1)
#define musp (mus*(1-g))      // reduced scattering coefficient (m^-1)
#define h 0.05                // layer thickness: h must be less than or equal to 0.2 m to be within the range of
                              // the curve fit for taumax
```

```
#define A 0.3                   // Amplitude of oscillation for modulated source
#define wmin 0.001                      // minimum weight of photon allowed to continue tracking the photon
#define psrw 0.10                       // probability of surviving the roulette wheel
#define taumin 10                       // minimum value of tau
#define taumax1 ceil(musp*(-7.3488*pow(h,6)-120.08*pow(h,5)+126*pow(h,4)-44.978*pow(h,3)+
                                5.7272*pow(h,2)+0.0732*h+0.00029))
#define taumax max(taumax1,double(taumin))          // minimum value of tau
#define nt int(taumax/dtau)             // the number of time bins
#define csize (2*rad+1)                 // the number of columns in the reflectance array
#define dt 20.1416015625e-12            // time resolution (sec), it is such a weird number only to simplify
                                           the frequency increments of the FFT
#define dtau (dt*omega)                 // non-dimensional time resolution
#define ro 0.001                        // laser beam radius (m)
#define dr ro                           // radial resolution (m)
#define rad 46                          // the number of detector locations
#define rsmall 0.015                    // minimum detector location (m)
#define rbig 0.060                      // maximum detector location (m)
#define omega (2*pi*0.1e9)              // angular frequency used for the non-dimensional time definition
#define siz 65536                       // size of input array to FFT algorithm
#define fmax 5                          // maximum frequency output from FFT algorithm in GHz
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr
#define nf 1.3                          // foam index of refraction
#define na 1.0                          // air index of refraction
#define ng 1.5                          // glass index of refraction
#define c0 3.0e8                        // speed of light in a vacuum (m/s)
#define c c0/nf                         // speed of light in foam (m/s)
#define pi 3.14159265358979
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1 - 1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS)

double ran2(long *idum);

int main ()
{
        //Define internal variables
        double ap,dp,tp,lpt,lpnd,w,path,x,y,z,mux,muy,muz,rin,theta,lw,tau,fct,random;
        double s,xnew,ynew,znew,phi,muxnew,muynew,muznew,alphai,alphac,alphat,ir,sc,rc,xc,yc;
        long *idum, seed;
        double reflect[nt+1][csize];
        double r[rad]={0};
        long double j,jprint;
        int jtc,i,m,check,ii,k,jj;

        //Variables declaration for the FFT algorithm
```

```
		unsigned long n,mmax,istep;
		double wtemp,wr,wpr,wpi,wi,tempr,tempi,freq,mod,AC,DC;
		unsigned long nn = siz/2;
		int isign = 1;
		double data[siz+1][rad+1];

		//Relate variables to their pointers
		idum=&seed;
		srand((unsigned)time(NULL));
		seed=-int(1e6*rand()/RAND_MAX)*2-1;

		//Define detector location array
		for (i=0; i<rad; i++)
			r[i]=rsmall+(rbig-rsmall)/(rad-1)*i;

		//Initialize the Reflectance Array: time = row, radius = column
		for (i=0; i<=nt; i++)
		{
			for (m=0; m<csize; m++)
				reflect[i][m]=0;
		}

		for (i=0; i<=siz; i++)
			for (m=0; m<=rad; m++)
				data[i][m]=0;

		//Initialize the photon tracking parameters
		ap=0;
		dp=0;
		tp=0;
		lpt=0;
		lpnd=0;
		lw=0;
		jprint=np/10;

		//Track the photon bundles

		for(j=1; j<=np; j++)
		{
			if (fmod(j,jprint)==0)
				cout<<"Number of Bundles = "<<j<<endl;

			rin=ro*sqrt(-log(ran2(idum)));
			theta=2*pi*ran2(idum);
			x=rin*cos(theta);
			y=rin*sin(theta);
			z=0;
			mux=0;
			muy=0;
			muz=1.0;
			w=1.0;
			path=0;
			bool cont=true;
	step:
			//Calculate the path length for the next step
			s=-log(ran2(idum))/mu;
```

```
//Move the Photon Bundle
xnew=x+mux*s;
ynew=y+muy*s;
znew=z+muz*s;

//If there is no boundary interaction
if(znew>0 && znew<h)
{
        //Update the position of the photon bundle
        x=xnew;
        y=ynew;
        z=znew;

        //Update the weight of the photon bundle
        ap=ap+w*(1-mus/mu);
        w=w*(mus/mu);

        //Calculate the total path length
        path=path+s;

        //Calculate the new scattering direction
        if(g<0.01)
                theta=acos(2*ran2(idum)-1);
        else
                theta=acos((1+g*g-pow(((1-g*g)/(1-g+2*g*ran2(idum))),2))/(2*g));

        phi=2*pi*ran2(idum);

        if(fabs(muz)>=0.99999)
        {
                muxnew=sin(theta)*cos(phi);
                muynew=sin(theta)*sin(phi);
                muznew=muz*cos(theta)/fabs(muz);
        }
        else
        {
                muxnew=sin(theta)*(mux*muz*cos(phi)-muy*sin(phi))
                           /sqrt(1-pow(muz,2))+mux*cos(theta);
                muynew=sin(theta)*(muy*muz*cos(phi)+mux*sin(phi))
                           /sqrt(1-pow(muz,2))+muy*cos(theta);
                muznew=-sin(theta)*cos(phi)*sqrt(1-pow(muz,2))
                           +muz*cos(theta);
        }
        mux=muxnew;
        muy=muynew;
        muz=muznew;

        goto check;
}

//if there is an interaction with the interface at z=0
else if(znew<=0)
{
        //Calculate the angle of incidence
        alphai=pi-acos(muz);
```

```
                    alphac=asin(na/nf);

                    //Calculate the internal reflectance
                    if(alphai>0.05 && alphai<alphac)
                    {
                            alphat=asin(nf*sin(alphai)/na);
                            ir=(pow(sin(alphai-alphat),2)/pow(sin(alphai+alphat),2)+
                               pow(tan(alphai-alphat),2)/pow(tan(alphai+alphat),2))/2;
                    }
                    else if(alphai<=0.05)
                            ir=pow((nf-na),2)/pow((nf+na),2);
                    else
                            ir=1;

                    //Calculate the point at which the photon crosses the interface
                    sc=s+fabs(znew)/muz;
                    xc=x+mux*sc;
                    yc=y+muy*sc;
                    rc=sqrt(pow(xc,2)+pow(yc,2));
                    jtc=int((path+sc)/(c*dt));

                    //Update the reflectance or the photon tracking parameter
                    ap=ap+w*(1-mus/mu)*(sc/s);
                    w=w*(1-(1-mus/mu)*(sc/s));

                    check=1;

                    if(jtc>nt)
                    {
                            lpt=lpt+(1-ir)*w;
                            goto after;
                    }

                    for(i=0; i<rad; i++)
                    {
                            if((r[i]-dr/2)<=rc && rc<(r[i]+dr/2))
                            {
                                    dp=dp+(1-ir)*w;
                                    reflect[jtc][2*i+1]=reflect[jtc][2*i+1]+(1-ir)*w;
                                    check=0;
                            }
                    }

                    if(check == 1)
                            lpnd=lpnd+(1-ir)*w;
after:
                    //Update the photon weight
                    ap=ap+w*ir*(1-mus/mu)*(1-sc/s);
                    w=w*ir*(1-(1-mus/mu)*(1-sc/s));

                    //Update the position and direction of the portion of the photon bundle that was
                      internally reflected
                    x=xnew;
                    y=ynew;
                    z=-znew;
                    muz=-muz;
```

```
		//Calculate the total path length the photon bundle has traveled
		path=path+s;

		//Calculate the new scattering direction
		if(g<0.01)
			theta=acos(2*ran2(idum)-1);
		else
			theta=acos((1+g*g-pow(((1-g*g)/
				(1-g+2*g*ran2(idum))),2))/(2*g));

		phi=2*pi*ran2(idum);

		if(fabs(muz)>=0.99999)
		{
			muxnew=sin(theta)*cos(phi);
			muynew=sin(theta)*sin(phi);
			muznew=muz*cos(theta)/fabs(muz);
		}
		else
		{
			muxnew=sin(theta)*(mux*muz*cos(phi)-muy*sin(phi))/
					sqrt(1-pow(muz,2))+mux*cos(theta);
			muynew=sin(theta)*(muy*muz*cos(phi)+mux*sin(phi))/
					sqrt(1-pow(muz,2))+muy*cos(theta);
			muznew=-sin(theta)*cos(phi)*sqrt(1-pow(muz,2))
					+muz*cos(theta);
		}
		mux=muxnew;
		muy=muynew;
		muz=muznew;
		goto check;
	}

	//If there is an interaction with the interface at z=h
	else if(znew>=h)
	{
		//Calculate the angle of incidence
		alphai=acos(muz);

		//Calculate the internal reflectance
		if((nf*sin(alphai)/ng)>1)
			ir=1;
		else if(alphai<0.05)
			ir=pow((nf-ng),2)/pow((nf+ng),2);
		else
		{
			alphat=asin(nf*sin(alphai)/ng);
			ir=(pow(sin(alphai-alphat),2)/pow(sin(alphai+alphat),2)+
			     pow(tan(alphai-alphat),2)/pow(tan(alphai+alphat),2))/2;
		}

		//Update the photon tracking parameter
		tp=tp+(1-ir)*w;

		//Update the photon weight
```

```
			sc=s+fabs(znew)/muz;
			ap=ap+w*ir*(1-mus/mu)*(1-sc/s);
			w=w*ir*(1-(1-mus/mu)*(1-sc/s));

			//Update the position and direction of the portion of the photon  bundle that was
			internally reflected
			x=xnew;
			y=ynew;
			z=2*h-znew;
			muz=-muz;

			//Calculate the total path length the photon bundle has traveled
			path=path+s;

			//Calculate the new scattering direction
			if(g<0.01)
				theta=acos(2*ran2(idum)-1);
			else
				theta=acos((1+g*g-pow(((1-g*g)/
					(1-g+2*g*ran2(idum))),2))/(2*g));

			phi=2*pi*ran2(idum);

			if(fabs(muz)>=0.99999)
			{
				muxnew=sin(theta)*cos(phi);
				muynew=sin(theta)*sin(phi);
				muznew=muz*cos(theta)/fabs(muz);
			}
			else
			{
				muxnew=sin(theta)*(mux*muz*cos(phi)-muy*sin(phi))/
					sqrt(1-pow(muz,2))+mux*cos(theta);
				muynew=sin(theta)*(muy*muz*cos(phi)+mux*sin(phi))/
					sqrt(1-pow(muz,2))+muy*cos(theta);
				muznew=-sin(theta)*cos(phi)*sqrt(1-pow(muz,2))
					+muz*cos(theta);
			}
			mux=muxnew;
			muy=muynew;
			muz=muznew;

			goto check;
		}
check:
		/*Decide whether to continue tracking the photon*/
		random=double(rand())/RAND_MAX;

		if((path/(c*dt))>nt)
		{
			lpt=lpt+w;
			cont=false;
		}
		else if(w<wmin && random<=psrw)
		{
			w=w+lw;
```

```
                lw=0;
                cont=true;
            }
            else if(w<wmin && random>psrw)
            {
                lw=lw+w;
                cont=false;
            }

            if(cont==true)
                goto step;


            if (fmod(j,jprint)==0)
            {
                //Output the conservation of energy results
                ofstream outdat ("ec.dat");
                outdat<<setprecision(20)<<
                    "Number of Photons ="<<j<<endl<<
                    "Detected Photons ="<<dp<<endl<<
                    "Transmitted Photons ="<<tp<<endl<<
                    "Absorbed Photons ="<<ap<<endl<<
                    "Photons with excess time of flight ="<<lpt<<endl<<
                    "Photons reflected but not detected ="<<lpnd<<endl<<
                    "Lost Weight ="<<lw<<endl<<
                    "Total ="<<dp+tp+lpt+lpnd+lw+ap<<endl<<endl;

                //Output the reflectance profiles
                fct=2*j*dr*dt*c/(h*pow(ro,2));
                for(i=0; i<=nt; i++)
                    for(ii=0; ii<rad; ii++)
                        reflect[i][2*ii+2]=reflect[i][2*ii+1]/(fct*r[ii]);

                ofstream outrefl ("reflect.txt");

                outrefl<<left<<"#_of_Photons = "<<setw(15)<<j<<endl<<
                     setw(25)<<"time_bin"<<setw(25)<<"tau";

                for(ii=0; ii<rad; ii++)

                outrefl<<setw(4)<<right<<"out_"<<setw(21)<<left<<r[ii]<<
                     setw(5)<<right<<"Refl_"<<setw(20)<<left<<r[ii]<<endl;

                for(i=0; i<=nt; i++)
                {
                    tau=(i+0.5)*dt*omega;
                    outrefl<<showpoint<<fixed<<setprecision(15)<<
                         setw(25)<<i<<setw(25)<<tau;

                    for(ii=1; ii<csize; ii++)
                        outrefl<<setw(25)<<reflect[i][ii];
                    outrefl<<endl;
                }
            }
        }
```

```
// Shift the data so the initial data point corresponds to zero time for the FFT algorithm
for(i=0; i<nt; i++)
        for(ii=0; ii<rad; ii++)
                data[2*i+3][ii+1]=(reflect[i][2*ii+2]+reflect[i+1][2*ii+2])/2;

// Initialize and format output files
ofstream outFFT ("FFT.txt");
ofstream outMeas ("Meas.txt");

outFFT<<right<<setw(25)<<"";
outMeas<<right<<setw(25)<<"";

for(ii=0; ii<rad; ii++)
{
        outFFT<<setw(12)<<right<<"r_"<<setw(13)<<left<<r[ii]<<setw(25)<<"";
        outMeas<<setw(12)<<right<<"r_"<<setw(13)<<left<<r[ii]<<setw(75)<<"";
}
outFFT<<endl;
outMeas<<endl;

outFFT<<setw(25)<<"Frequency (GHz)";
outMeas<<setw(25)<<"Frequency (GHz)";

for(ii=0; ii<rad; ii++)
{
        outFFT<<setw(25)<<"Real"<<setw(25)<<"Imaginary";
        outMeas<<setw(25)<<"DC"<<setw(25)<<"AC"<<setw(25)<<"Modulation"
          <<setw(25)<<"Phase_Shift";
}
outFFT<<endl;
outMeas<<endl;

// Begin the FFT algorithm
for(ii=1; ii<=rad; ii++)
{
        n=nn << 1;
        k=1;
        for (i=1;i<n;i+=2)
        {
                if(k>i)
                {
                        SWAP(data[k][ii],data[i][ii]);
                        SWAP(data[k+1][ii],data[i+1][ii]);
                }

                m=n >> 1;
                while (m >= 2 && k > m)
                {
                        k -= m;
                        m >>= 1;
                }
                k += m;
        }

        mmax=2;
```

```
		while(n > mmax)
		{
			istep=mmax << 1;
			theta=isign*(2*pi/mmax);
			wtemp=sin(0.5*theta);
			wpr = -2.0*wtemp*wtemp;
			wpi=sin(theta);
			wr=1.0;
			wi=0.0;
			for(m=1;m<mmax;m+=2)
			{
				for(i=m;i<=n;i+=istep)
				{
					k=i+mmax;
					tempr=wr*data[k][ii]-wi*data[k+1][ii];
					tempi=wr*data[k+1][ii]+wi*data[k][ii];
					data[k][ii]=data[i][ii]-tempr;
					data[k+1][ii]=data[i+1][ii]-tempi;
					data[i][ii] += tempr;
					data[i+1][ii] += tempi;
				}
				wr=(wtemp=wr)*wpr-wi*wpi+wr;
				wi=wi*wpr+wtemp*wpi+wi;
			}
			mmax=istep;
		}
	}

	// Output the data as a function of frequency with a maximum  frequency of fmax
	for(jj=1;jj<(fmax*siz*dt*1e9+2);jj+=2)
	{
		freq=(jj-1)/(siz*dt*1e9);
		outFFT<<setw(25)<<setprecision(15)<<freq;
		outMeas<<setw(25)<<setprecision(15)<<freq;

		for(ii=1; ii<=rad; ii++)
		{
			mod=A*pow((pow(data[jj+1][ii],2)+pow(data[jj][ii],2))/
			        (pow(data[2][ii],2)+pow(data[1][ii],2)),0.5);
			phi=atan2(data[jj+1][ii],data[jj][ii]);
			AC=A*dtau*freq*1e9*2*pi/omega*
			      pow((pow(data[jj+1][ii],2)+pow(data[jj][ii],2)),0.5);
			DC=data[1][ii]*dtau*freq*1e9*2*pi/omega;

			outFFT<<setw(25)<<data[jj][ii]*dtau<<setw(25)<<data[jj+1][ii]*dtau;
			outMeas<<setw(25)<<DC<<setw(25)<<AC<<setw(25)
			          <<mod<<setw(25)<<phi;
		}
		outFFT<<endl;
		outMeas<<endl;
	}


}
```

```
// Random Number Generator
double ran2(long *idum)
{
        int j;
        long k;
        static long idum2=123456789;
        static long iy=0;
        static long iv[NTAB];
        double temp;

        if(*idum <= 0)
        {
                if(-(*idum) < 1) *idum = 1;
                else *idum = -(*idum);
                idum2 = (*idum);
                for(j=NTAB+7;j>=0;j--)
                {
                        k=(*idum)/IQ1;
                        *idum = IA1*(*idum-k*IQ1)-k*IR1;
                        if(idum < 0) *idum += IM1;
                        if(j < NTAB) iv[j] = *idum;
                }
                iy = iv[0];
        }
        k = (*idum)/IQ1;
        *idum = IA1*(*idum-k*IQ1)-k*IR1;
        if(*idum < 0) *idum += IM1;
        k = idum2/IQ2;
        idum2 = IA2*(idum2-k*IQ2)-k*IR2;
        if(idum2 < 0) idum2 += IM2;
        j = iy/NDIV;
        iy = iv[j]-idum2;
        iv[j] = *idum;
        if(iy < 1) iy += IMM1;
        temp = AM * iy;
        if(temp > RNMX) return RNMX;
        else return temp;

}
```

D.2 Conjugate-Gradient Inversion Algorithm Code

Source code file: inv_MPS.C

Compile instructions: use the following command in a UNIX command window in the directory where inv_MPS.C is located

```
g++ inv_MPS.C –o desirednameofexecutable –lm -O
```

run executable by typing "./" then name of executable file, then press "Enter."

User Inputs:	initial guesses for the parameters, the detector range to be used in inverse problem, desired frequency and other measurement parameters (see note below).
Program Output Files:	dif.txt – measurements based on diffusion theory sens.txt – sensitivities of the measurements with respect to the parameters pars.txt – convergence history of the estimated parameters

Source Code:

```
/*This code is set to perform the inverse problem using a Modified Point Source model. It is coded such
that any arbitrary range of detector locations can be used for the measurement set at a single frequency. The
measurement file (MC.txt) should include DC, AC, and Phase shift (in that order) for the full range of
detectors defined by "rsmall" to "rbig".*/

#include <iomanip>
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
using namespace std;

#define na 1.0
#define nf 1.3
#define ng 1.5
#define nb 1e6
#define c (3e8/nf)
#define ro 0.001
#define drad int(1000*(rbig-rsmall)+1.0001)
#define rad int(1000*(rmax-rmin)+1.0001)
#define rmin 0.020          // minimum desired detector location to be used in the inverse algorithm
#define rrmin int(1000*(rmin-rsmall)+0.0001)
#define rmax 0.020          // maximum desired detector location to be used in the inverse algorithm
#define rsmall 0.015        // smallest measurement location contained within MC.txt
#define rbig 0.040          // largest measurement location contained within MC.txt
#define a ro/L
#define A 0.3
#define d (c/(3*musp*omega*pow(ro,2)))
#define N 1000001
#define MM 10
#define pi 3.141592653589793
#define conv 0.0001
#define conv1 0.00001
#define L_start 0.08        // initial guess for layer thickness (m)
#define musp_start 198      // initial guess for the reduced scattering coefficient (m-1)
#define inc_musp (1e-7*mid_musp)          // increment used for numerical derivative
#define min_musp (mid_musp-inc_musp)
#define max_musp (mid_musp+inc_musp)
#define inc_L (2e-7*mid_L)                // increment used for numerical derivative
#define min_L (mid_L-inc_L/2)
#define max_L (mid_L+inc_L/2)
#define zetao (1/(musp*L))
```

```
#define tauo (omega/(musp*c))
#define omega 2*pi*1e9*freq
#define freq 0.01               //source frequency (GHz)

double bessj0(double xxx[N],double bess[N],double siz);
double bessjj0(double xxx);
double bessi0(double xxx);
double bessk0(double xxx);
double bessi1(double xxx);
double bessk1(double xxx);

int main()
{
            ifstream infile("MC.txt");
            ofstream outmeas("dif20-20.txt", ios::app);
            ofstream outsens("sens20-20.txt", ios::app);
            ofstream outpar("pars20-20.txt", ios::app);

            /*Variables needed for the binding routine and more*/
            int nbb,i,rr,count,count2,ncc,checkS,checkT,checkU,term;
            double x,fp,fc,dx,x1,x2,x11,x22,dxx,ddxx,fpp,fcc;
            double *xb1 = new double[N];
            double *xb2 = new double[N];
            double *xbb1 = new double[N];
            double *xbb2 = new double[N];
            double *xbb3 = new double[N];
            double *xbb4 = new double[N];
            double *y = new double[N];
            double *y1 = new double[N];
            double *bessy = new double [N];
            double *bessxbb1 = new double [N];
            double *bessxbb2 = new double [N];
            double *r = new double [rad];
            double *rho = new double [rad];
            double *MC = new double [3*drad];
            double *meas = new double [3*rad];
            double *epsilon = new double [3*rad];
            double sens[3*rad][2] = {0};
            double *GS = new double [2];
            double *GSold = new double [2];
            double *dk = new double [2];
            double *dkold = new double [2];

            /*Variables needed for the root finding technique*/
            int j,k,iter,checkmusp,checkL;
            double fL,fH,xL,xH,swap,del,f,fLL,fHH,xLL,xHH,ff;
            double *beta = new double[N];
            double *lambda = new double[N];
            double xacc = 1e-6;
            double maxit = 100;

            /*Variables necessary for the diffusion model*/
            double RO,RL,hO,hL,HO,HL,NN,KO,Kzetao,Ssum1,Tsum1,Usum1,U2sum;
            double *bess = new double [N];
            double *bess1 = new double [N];
            double *bess2 = new double [N];
```

```
        double *Ssum2 = new double [N];
        double *Tsum2 = new double [N];
        double *Usum2 = new double [N];

        /*Variables necessary for the Numerical Integration*/
        double tnm,sumS,sumT,sumU1,sumU2,del1,ddel1,it,aa,bb,sold,told,u1old,u2old,musp,L;
        double mid_musp,mid_L,mid_muspold,mid_Lold,gam,step,top,bot;
        int kk,jj,n,fr;
        double *sn = new double [N];
        double *tn = new double [N];
        double *un1 = new double [N];
        double *un2 = new double [N];
        double *xx1 = new double [N];
        double *xx2 = new double [N];

        double DC[3][rad] = {0};
        double AC[3][rad] = {0};
        double Phase[3][rad] = {0};

        for (i=0; i<rad; i++)
        {
                r[i]=rmin+0.001*i;          //assumes uniform increments of 0.001 m
                rho[i]=r[i]/ro;
        }
        for (i=0; i<(3*drad); i++)
                infile >> MC[i];

        x1=0.01;
        x2=50000;
        x11=0.01;
        x22=50000;

        RO=1-(1-(-0.4399+0.7099*nf/na-0.3319*pow(nf,2)/pow(na,2)+
             0.0636*pow(nf,3)/pow(na,3)))/(pow(nf,2)/pow(na,2));
        RL=1-(1-(-0.4399+0.7099*nf/ng-0.3319*pow(nf,2)/pow(ng,2)+
            0.0636*pow(nf,3)/pow(ng,3)))/(pow(nf,2)/pow(ng,2));
        hO=(1-RO)/(2*(1+RO));
        hL=(1-RL)/(2*(1+RL));

        mid_musp = musp_start;
        mid_L = L_start;
        term=0;
        checkmusp=5;
        checkL=5;

        outpar<<setprecision(10)<<setw(15)<<mid_L<<setw(15)<<mid_musp<<endl;

        iter=1;
begin:;
        musp=min_musp-inc_musp;
        L=mid_L;
        k=-1;

        do
        {
                k=k+1;
```

```
musp=musp+inc_musp;
cout<<endl<<"Current musp = "<<musp<<endl<<endl;
HO=3*musp*hO;
HL=3*musp*hL;
nbb=0;
count=0;
dx=(x2-x1)/nb;
dxx=(x22-x11)/nb;

/*Here we begin the binding routine*/
fp=(x=x1)*L*(HL+HO)*cos(x=x1)-(pow((x=x1),2)-HO*HL*pow(L,2))*sin(x=x1);

for (i=1;i<=nb;i++)
{
        fc=(x+=dx)*L*(HL+HO)*cos(x)-(pow((x),2)-HO*HL*pow(L,2))*sin(x);

        if (fc*fp <= 0.0)
        {
                xb1[++nbb]=x-dx;
                xb2[nbb]=x;
                count=count+1;
        }
        fp=fc;
}

/*Here we begin the root finding technique for the eigenvalues using the information
from the binding routine*/
for (i=1;i<=count;i++)
{
        fL=(x=xb1[i])*L*(HL+HO)*cos(x=xb1[i])-(pow((x=xb1[i]),2)-
                HO*HL*pow(L,2))*sin(x=xb1[i]);
        fH=(x=xb2[i])*L*(HL+HO)*cos(x=xb2[i])-(pow((x=xb2[i]),2)-
                HO*HL*pow(L,2))*sin(x=xb2[i]);

        if (fL < 0.0)
        {
                xL=xb1[i];
                xH=xb2[i];
        }
        else
        {
                xL=xb2[i];
                xH=xb1[i];
                swap=fL;
                fL=fH;
                fH=swap;
        }
        dx=xH-xL;
        for (j=1;j<=maxit;j++)
        {
                beta[i]=xL+dx*fL/(fL-fH);
                f=(x=beta[i])*L*(HL+HO)*cos(x=beta[i])-(pow((x=beta[i]),2)-
                        HO*HL*pow(L,2))*sin(x=beta[i]);

                if (f < 0.0)
                {
                    del=xL-beta[i];
                    xL=beta[i];
                    fL=f;
                }
                else
                {
                    del=xH-beta[i];
                    xH=beta[i];
                    fH=f;
                }
                dx=xH-xL;

                if (fabs(del) < xacc || f == 0.0)
                    goto quit1;
            }
quit1:;

        }

        checkmusp=0;

        for (rr=0; rr<rad; rr++)
        {
            count2=0;
            ncc=0;
            cout<<"Current Radial Location = "<<r[rr]<<endl;
            y[1]=x11;
            y1[1]=rho[rr]*y[1];

            for (i=2;i<=nb;i++)
            {
                y[i]=y[i-1]+dxx;
                y1[i]=rho[rr]*y[i];
            }

            bessj0(y1,bessy,nb);
            fpp=bessy[1];

            for (i=2;i<=nb;i++)
            {
                fcc=bessy[i];

                if (fcc*fpp <= 0.0)
                {
                    xbb1[++ncc]=y[i]-dxx;
                    xbb2[ncc]=y[i];
                    count2=count2+1;
                }
                fpp=fcc;
            }

            for(i=1;i<=count2;i++)
            {
                xbb3[i]=xbb1[i]*rho[rr];
                xbb4[i]=xbb2[i]*rho[rr];
            }
```

```
                    bessj0(xbb3,bessxbb1,count2);
                    bessj0(xbb4,bessxbb2,count2);

                    /*Here we begin the root finding technique for the values of lambda using the
                    information from the binding routine*/
                    for (i=1;i<=count2;i++)
                    {
                            fLL=bessxbb1[i];
                            fHH=bessxbb2[i];

                            if (fLL < 0.0)
                            {
                                    xLL=xbb1[i];
                                    xHH=xbb2[i];
                            }
                            else
                            {
                                    xLL=xbb2[i];
                                    xHH=xbb1[i];
                                    swap=fLL;
                                    fLL=fHH;
                                    fHH=swap;
                            }
                            ddxx=xHH-xLL;
                            for (j=1;j<=maxit;j++)
                            {
                                    lambda[i]=xLL+ddxx*fLL/(fLL-fHH);
                                    ff=bessjj0(rho[rr]*lambda[i]);

                                    if (ff < 0.0)
                                    {
                                            del=xLL-lambda[i];
                                            xLL=lambda[i];
                                            fLL=ff;
                                    }
                                    else
                                    {
                                            del=xHH-lambda[i];
                                            xHH=lambda[i];
                                            fHH=ff;
                                    }
                                    ddxx=xHH-xLL;

                                    if (fabs(del) < xacc || f == 0.0)
                                            goto quit2;
                            }
quit2:;
                    }

                    kk=0;
                    checkS=0;
                    checkT=0;
                    checkU=0;

                    do      /*Summing Loop*/
```

```
                                        {
                                                kk=kk+1;
                                                Ssum1=0;
                                                Tsum1=0;
                                                Usum1=0;
                                                U2sum=0;
                                                NN=((pow(beta[kk],2)+pow(HO,2)*pow(L,2))*(1+HL*L/(pow
                                                (beta[kk],2)+pow(HL,2)*pow(L,2)))+HO*L)/(2*pow(beta[kk],2));
                                                KO=1/sqrt(NN);
                                                Kzetao=(cos(beta[kk]*zetao)+HO*L/beta[kk]
                                                        *sin(beta[kk]*zetao))/sqrt(NN);

                                                jj=0;
                                                do
                                                {
                                                        jj=jj+1;
                                                        aa=lambda[jj-1];
                                                        bb=lambda[jj];

                                                        i=1;
                                                        for(n=1;n<=MM;n++)
                                                        {
                                                                if(n==1)
                                                                {
                                                                        xx1[i]=0.5*(aa+bb);
                                                                        i=i+1;
                                                                        goto quit3;
                                                                }
                                                                for(it=1,j=1;j<n-1;j++) it *=3;
                                                                tnm=it;
                                                                del1=(bb-aa)/(3.0*tnm);
                                                                ddel1=del1+del1;
                                                                xx1[i]=aa+0.5*del1;
                                                                i=i+1;

                                                                for(j=1;j<=it;j++)
                                                                {
                                                                        xx1[i]=xx1[i-1]+ddel1;
                                                                        i=i+1;
                                                                        xx1[i]=xx1[i-1]+del1;
                                                                        i=i+1;
                                                                }
quit3:;
                                                        }

                                                        for(j=1;j<i;j++)
                                                                xx2[j]=rho[rr]*xx1[j];

                                                        bessj0(xx2,bess,i);

                                                        i=1; /*Definite Integration between two finite numbers for S */
                                                        n=1;
                                                        sold=0;

                                                        if(pow(1/d,2)-(conv1*pow((pow(aa,2)
                                                        +pow((a*beta[kk]),2)),2)) < 0)
                                        goto Send3;
                                do
                                {
                                        if(n==1)
                                        {
                                                sn[jj]=(bb-aa)*xx1[i]*d*bess[i]
                                                /(pow(xx1[i],2)+pow((a*beta[kk]),2));
                                                i=i+1;
                                                goto Send1;
                                        }
                                        for(it=1,j=1;j<n-1;j++) it *=3;
                                        tnm=it;
                                        sumS=0.0;
                                        for(j=1;j<=it;j++)
                                        {
                                                sumS += xx1[i]*d*bess[i]/(pow(xx1[i],2)
                                                        +pow((a*beta[kk]),2));
                                                i=i+1;
                                                sumS += xx1[i]*d*bess[i]/(pow(xx1[i],2)
                                                +pow((a*beta[kk]),2));
                                                i=i+1;
                                        }

                                        sold=sn[jj];
                                        sn[jj]=(sn[jj]+(bb-aa)*sumS/tnm)/3.0;
Send1:;
                                        n=n+1;

                                        if(n>MM)
                                                goto Send2;

                                }
                                while(fabs((sn[jj]-sold)/sn[jj]) > conv);
Send2:;
                                Ssum1=Ssum1+sn[jj];

                                if(jj>=count2)
                                        goto Send3;
                        }
                        while(checkS < 5);
Send3:;
                        if(aa==0)
                                checkS=checkS+1;

                        jj=0;
                        do
                        {
                                jj=jj+1;
                                aa=lambda[jj-1];
                                bb=lambda[jj];

                                i=1;
                                for(n=1;n<=MM;n++)
                                {
                                        if(n==1)
                                        {
```

```
                        xx1[i]=0.5*(aa+bb);
                        i=i+1;
                        goto quit4;
                }
                for(it=1,j=1;j<n-1;j++) it *=3;
                tnm=it;
                del1=(bb-aa)/(3.0*tnm);
                ddel1=del1+del1;
                xx1[i]=aa+0.5*del1;
                i=i+1;

                for(j=1;j<=it;j++)
                {
                        xx1[i]=xx1[i-1]+ddel1;
                        i=i+1;
                        xx1[i]=xx1[i-1]+del1;
                        i=i+1;
                }

quit4:;


        }

        for(j=1;j<i;j++)
                xx2[j]=rho[rr]*xx1[j];

        bessj0(xx2,bess,i);

        i=1;/*Definite Integration between two finite numbers for T */
        n=1;
        told=0;

        if(pow(1/d,2)<(conv1*pow((pow(aa,2)
                +pow((a*beta[kk]),2)),2)))
                goto Tend3;

        do
        {
                if(n==1)
                {
                        tn[jj]=(bb-aa)*xx1[i]*d*bess[i]*
                        (pow(xx1[i],2)+pow((a*beta[kk]),2))/
                        (pow((pow(xx1[i],2)+pow
                        ((a*beta[kk]),2)),2)+1/pow(d,2));
                        i=i+1;
                        goto Tend1;
                }
                for(it=1,j=1;j<n-1;j++) it *=3;
                tnm=it;
                sumT=0.0;
                for(j=1;j<=it;j++)
                {
                        sumT += xx1[i]*d*bess[i]*(pow(xx1[i],2)+
                        pow((a*beta[kk]),2))/(pow((pow(xx1[i],2)
                        +pow((a*beta[kk]),2)),2)+1/pow(d,2));
                        i=i+1;
```

```
                            sumT += xx1[i]*d*bess[i]*(pow(xx1[i],2)+
                            pow((a*beta[kk]),2))/(pow((pow(xx1[i],2)
                            +pow((a*beta[kk]),2)),2)+1/pow(d,2));
                            i=i+1;
                        }

                        told=tn[jj];
                        tn[jj]=(tn[jj]+(bb-aa)*sumT/tnm)/3.0;
Tend1:;
                        n=n+1;

                        if(n>MM)
                            goto Tend2;

                    }
                    while(fabs((tn[jj]-told)/tn[jj]) > conv);
Tend2:;
                    Tsum1=Tsum1+tn[jj];

                    if(jj>=count2)
                        goto Tend3;
                }
                while(checkT < 5);
Tend3:;
                if(aa==0)
                    checkT=checkT+1;

                jj=0;
                do
                {
                    jj=jj+1;
                    aa=lambda[jj-1];
                    bb=lambda[jj];

                    i=1;
                    for(n=1;n<=MM;n++)
                    {
                        if(n==1)
                        {
                            xx1[i]=0.5*(aa+bb);
                            i=i+1;
                            goto quit5;
                        }
                        for(it=1,j=1;j<n-1;j++) it *=3;
                        tnm=it;
                        del1=(bb-aa)/(3.0*tnm);
                        ddel1=del1+del1;
                        xx1[i]=aa+0.5*del1;
                        i=i+1;

                        for(j=1;j<=it;j++)
                        {
                            xx1[i]=xx1[i-1]+ddel1;
                            i=i+1;
                            xx1[i]=xx1[i-1]+del1;
                            i=i+1;
```

```
				}
quit5:;
			}

			for(j=1;j<i;j++)
				xx2[j]=rho[rr]*xx1[j];

			bessj0(xx2,bess,i);

			i=1;
			n=1;
			u1old=0;
			u2old=0;

			if(pow(1/d,2)<(conv1*pow((pow(aa,2)
			+pow((a*beta[kk]),2)),2)))
				goto Uend3;

			do/*Definite Integration between two finite numbers for U*/
			{
				if(n==1)
				{
					un1[jj]=(bb-aa)*xx1[i]*bess[i]/
					(pow((pow(xx1[i],2)+pow(
					(a*beta[kk]),2)),2)+1/pow(d,2));
					un2[jj]=(bb-aa)*xx1[i]*bess[i]/
					pow((pow(xx1[i],2)+pow(
					(a*beta[kk]),2)),2);
					i=i+1;
					goto Uend1;
				}
				for(it=1,j=1;j<n-1;j++) it *=3;
				tnm=it;
				sumU1=0.0;
				sumU2=0.0;
				for(j=1;j<=it;j++)
				{
					sumU1 += xx1[i]*bess[i]/
					(pow((pow(xx1[i],2)+
					pow((a*beta[kk]),2)),2)+1/pow(d,2));
					sumU2 += xx1[i]*bess[i]/
					pow((pow(xx1[i],2)+
					pow((a*beta[kk]),2)),2);
					i=i+1;
					sumU1 += xx1[i]*bess[i]/
					(pow((pow(xx1[i],2)+
					pow((a*beta[kk]),2)),2)+1/pow(d,2));
					sumU2 += xx1[i]*bess[i]/
					pow((pow(xx1[i],2)+
					pow((a*beta[kk]),2)),2);
					i=i+1;
				}

				u1old=un1[jj];
				un1[jj]=(un1[jj]+(bb-aa)*sumU1/tnm)/3.0;
				u2old=un2[jj];
```

```
                                                        un2[jj]=(un2[jj]+(bb-aa)*sumU2/tnm)/3.0;
Uend1:;
                                                        n=n+1;
                                                        if(n>MM)
                                                                goto Uend2;
                                                }
                                                while(fabs((un1[jj]-u1old)/un1[jj]) >
                                                        conv || fabs((un2[jj]-u2old)/un2[jj]) > conv );
Uend2:;
                                                Usum1=Usum1+un1[jj];
                                                U2sum=U2sum+un2[jj];

                                                if(jj>=count2)
                                                        goto Uend3;
                                        }
                                        while(checkU < 5);
Uend3:;

                                        if(aa==0)
                                                checkU=checkU+1;

                                        Ssum2[kk]=Ssum2[kk-1]+KO*Kzetao*d*bessk0(a*rho[rr]*beta[kk]);
                                        Tsum2[kk]=Tsum2[kk-1]+KO*Kzetao*
                                                (Tsum1+d*bessk0(a*rho[rr]*beta[kk])-Ssum1);
                                        Usum2[kk]=Usum2[kk-1]+KO*Kzetao*(Usum1+rho[rr]/
                                                (2*a*beta[kk])*bessk1(a*rho[rr]*beta[kk])-U2sum);
                                }
                                while(fabs(Ssum2[kk]-Ssum2[kk-1])!=0 || fabs(Tsum2[kk]-Tsum2[kk-1])!=0 ||
                                         fabs(Usum2[kk]-Usum2[kk-1])!=0);

                                DC[k][rr]=hO*Ssum2[kk]/(2*pow(d,2));
                                AC[k][rr]=hO*A/(2*pow(d,2))*sqrt(pow(Tsum2[kk],2)+pow(Usum2[kk],2));
                                Phase[k][rr]=tauo+atan2(Usum2[kk],Tsum2[kk]);

                                if(checkmusp>=5)
                                        Phase[k][rr]=Phase[k][rr]+2*pi;

                                if(Usum2[kk]<0 && checkmusp<5)
                                {
                                        Phase[k][rr]=Phase[k][rr]+2*pi;
                                        checkmusp=checkmusp+1;
                                }

                        }

                }
                while(k < 2);

                for (i=0; i<rad; i++)
                {
                        meas[i]=DC[1][i];
                        epsilon[i]=(MC[i+rrmin]-meas[i])/MC[i+rrmin];
                        outmeas<<setprecision(15)<<setw(25)<<DC[0][i]<<setw(25)<<DC[1][i]<<setw(25)
                                <<DC[2][i]<<setw(25)<<MC[i+rrmin]<<setw(25)<<mid_musp<<setw(25)
                                <<max_musp-min_musp<<endl;
```

```
		sens[i][0]=(DC[2][i]-DC[0][i])*mid_musp/MC[i+rrmin]/(max_musp-min_musp);

	}

	musp=mid_musp;
	L=min_L-inc_L;
	k=-1;

	do
	{
		k=k+1;
		L=L+inc_L;
		cout<<endl<<"Current L = "<<L<<endl<<endl;
		HO=3*musp*hO;
		HL=3*musp*hL;
		nbb=0;
		count=0;
		dx=(x2-x1)/nb;
		dxx=(x22-x11)/nb;

		//Here we begin the binding routine

		fp=(x=x1)*L*(HL+HO)*cos(x=x1)-(pow((x=x1),2)-HO*HL*pow(L,2))*sin(x=x1);

		for (i=1;i<=nb;i++)
		{
			fc=(x+=dx)*L*(HL+HO)*cos(x)-(pow((x),2)-HO*HL*pow(L,2))*sin(x);

			if (fc*fp <= 0.0)
			{
				xb1[++nbb]=x-dx;
				xb2[nbb]=x;
				count=count+1;
			}
			fp=fc;
		}

		//Here we begin the root finding technique for the eigenvalues using the information from
		the binding routine
		for (i=1;i<=count;i++)
		{
			fL=(x=xb1[i])*L*(HL+HO)*cos(x=xb1[i])-(pow((x=xb1[i]),2)-
			HO*HL*pow(L,2))*sin(x=xb1[i]);
			fH=(x=xb2[i])*L*(HL+HO)*cos(x=xb2[i])-(pow((x=xb2[i]),2)-
			HO*HL*pow(L,2))*sin(x=xb2[i]);

			if (fL < 0.0)
			{
				xL=xb1[i];
				xH=xb2[i];
			}
			else
			{
				xL=xb2[i];
				xH=xb1[i];
				swap=fL;
```

```
                fL=fH;
                fH=swap;
            }
            dx=xH-xL;
            for (j=1;j<=maxit;j++)
            {
                beta[i]=xL+dx*fL/(fL-fH);
                f=(x=beta[i])*L*(HL+HO)*cos(x=beta[i])-(pow((x=beta[i]),2)-HO*HL
                *pow(L,2))*sin(x=beta[i]);

                if (f < 0.0)
                {
                    del=xL-beta[i];
                    xL=beta[i];
                    fL=f;
                }
                else
                {
                    del=xH-beta[i];
                    xH=beta[i];
                    fH=f;
                }
                dx=xH-xL;

                if (fabs(del) < xacc || f == 0.0)
                    goto qquit1;
            }
qquit1:;

        }

        checkL=0;

        for (rr=0; rr<rad; rr++)
        {
            count2=0;
            ncc=0;
            cout<<"Current Radial Location = "<<r[rr]<<endl;
            y[1]=x11;
            y1[1]=rho[rr]*y[1];

            for (i=2;i<=nb;i++)
            {
                y[i]=y[i-1]+dxx;
                y1[i]=rho[rr]*y[i];
            }

            bessj0(y1,bessy,nb);
            fpp=bessy[1];

            for (i=2;i<=nb;i++)
            {
                fcc=bessy[i];

                if (fcc*fpp <= 0.0)
                {
```

```
                xbb1[++ncc]=y[i]-dxx;
                xbb2[ncc]=y[i];
                count2=count2+1;
        }
        fpp=fcc;
}

for(i=1;i<=count2;i++)
{
        xbb3[i]=xbb1[i]*rho[rr];
        xbb4[i]=xbb2[i]*rho[rr];
}

bessj0(xbb3,bessxbb1,count2);
bessj0(xbb4,bessxbb2,count2);

//Here we begin the root finding technique for the values of lambda using the
information from the binding routine
for (i=1;i<=count2;i++)
{
        fLL=bessxbb1[i];
        fHH=bessxbb2[i];

        if (fLL < 0.0)
        {
                xLL=xbb1[i];
                xHH=xbb2[i];
        }
        else
        {
                xLL=xbb2[i];
                xHH=xbb1[i];
                swap=fLL;
                fLL=fHH;
                fHH=swap;
        }
        ddxx=xHH-xLL;
        for (j=1;j<=maxit;j++)
        {
                lambda[i]=xLL+ddxx*fLL/(fLL-fHH);
                ff=bessjj0(rho[rr]*lambda[i]);

                if (ff < 0.0)
                {
                        del=xLL-lambda[i];
                        xLL=lambda[i];
                        fLL=ff;
                }
                else
                {
                        del=xHH-lambda[i];
                        xHH=lambda[i];
                        fHH=ff;
                }
                ddxx=xHH-xLL;
```

```
                        if (fabs(del) < xacc || f == 0.0)
                            goto qquit2;
                    }
qquit2:;

                }

                kk=0;
                checkS=0;
                checkT=0;
                checkU=0;

                do      //Summing Loop
                {
                    kk=kk+1;
                    Ssum1=0;
                    Tsum1=0;
                    Usum1=0;
                    U2sum=0;
                    NN=((pow(beta[kk],2)+pow(HO,2)*pow(L,2))*(1+HL*L/(
                    pow(beta[kk],2)+pow(HL,2)*pow(L,2)))+HO*L)/(2*pow(beta[kk],2));
                    KO=1/sqrt(NN);
                    Kzetao=(cos(beta[kk]*zetao)+HO*L/
                        beta[kk]*sin(beta[kk]*zetao))/sqrt(NN);

                    jj=0;
                    do
                    {
                        jj=jj+1;
                        aa=lambda[jj-1];
                        bb=lambda[jj];

                        i=1;
                        for(n=1;n<=MM;n++)
                        {
                            if(n==1)
                            {
                                xx1[i]=0.5*(aa+bb);
                                i=i+1;
                                goto qquit3;
                            }
                            for(it=1,j=1;j<n-1;j++) it *=3;
                            tnm=it;
                            del1=(bb-aa)/(3.0*tnm);
                            ddel1=del1+del1;
                            xx1[i]=aa+0.5*del1;
                            i=i+1;

                            for(j=1;j<=it;j++)
                            {
                                xx1[i]=xx1[i-1]+ddel1;
                                i=i+1;
                                xx1[i]=xx1[i-1]+del1;
                                i=i+1;
                            }
qquit3:;
```

```
                        }

                        for(j=1;j<i;j++)
                                xx2[j]=rho[rr]*xx1[j];

                        bessj0(xx2,bess,i);


                        i=1;//Definite Integration between two finite numbers for S
                        n=1;
                        sold=0;

                        if(pow(1/d,2)-(conv1*pow((pow
                        (aa,2)+pow((a*beta[kk]),2)),2)) < 0)
                                goto SSend3;

                        do
                        {
                                if(n==1)
                                {
                                        sn[jj]=(bb-aa)*xx1[i]*d*bess[i]/
                                        (pow(xx1[i],2)+pow((a*beta[kk]),2));
                                        i=i+1;
                                        goto SSend1;
                                }
                                for(it=1,j=1;j<n-1;j++) it *=3;
                                tnm=it;
                                sumS=0.0;
                                for(j=1;j<=it;j++)
                                {
                                        sumS += xx1[i]*d*bess[i]/
                                        (pow(xx1[i],2)+pow((a*beta[kk]),2));
                                        i=i+1;
                                        sumS += xx1[i]*d*bess[i]/
                                        (pow(xx1[i],2)+pow((a*beta[kk]),2));
                                        i=i+1;
                                }

                                sold=sn[jj];
                                sn[jj]=(sn[jj]+(bb-aa)*sumS/tnm)/3.0;
SSend1:;
                                n=n+1;

                                if(n>MM)
                                        goto SSend2;

                        }
                        while(fabs((sn[jj]-sold)/sn[jj]) > conv);
SSend2:;
                        Ssum1=Ssum1+sn[jj];

                        if(jj>=count2)
                                goto SSend3;
                }
                while(checkS < 5);
SSend3:;
```

```
                    if(aa==0)
                            checkS=checkS+1;

                    jj=0;
                    do
                    {
                            jj=jj+1;
                            aa=lambda[jj-1];
                            bb=lambda[jj];

                            i=1;
                            for(n=1;n<=MM;n++)
                            {
                                    if(n==1)
                                    {
                                            xx1[i]=0.5*(aa+bb);
                                            i=i+1;
                                            goto qquit4;
                                    }
                                    for(it=1,j=1;j<n-1;j++) it *=3;
                                    tnm=it;
                                    del1=(bb-aa)/(3.0*tnm);
                                    ddel1=del1+del1;
                                    xx1[i]=aa+0.5*del1;
                                    i=i+1;

                                    for(j=1;j<=it;j++)
                                    {
                                            xx1[i]=xx1[i-1]+ddel1;
                                            i=i+1;
                                            xx1[i]=xx1[i-1]+del1;
                                            i=i+1;
                                    }
qquit4:;
                            }

                            for(j=1;j<i;j++)
                                    xx2[j]=rho[rr]*xx1[j];

                            bessj0(xx2,bess,i);

                            i=1;//Definite Integration between two finite numbers for T
                            n=1;
                            told=0;
                            if(pow(1/d,2)<(conv1*pow((pow(aa,2)+pow(
                                    (a*beta[kk]),2)),2)))
                                    goto TTend3;

                            do
                            {
                                    if(n==1)
                                    {
                                            tn[jj]=(bb-aa)*xx1[i]*d*bess[i]*(pow(xx1[i]
                                            ,2)+pow((a*beta[kk]),2))/(pow((pow(xx1[i]
                                            ,2)+pow((a*beta[kk]),2)),2)+1/pow(d,2));
                                            i=i+1;
```

```
                                                goto TTend1;
                                        }
                                        for(it=1,j=1;j<n-1;j++) it *=3;
                                        tnm=it;
                                        sumT=0.0;
                                        for(j=1;j<=it;j++)
                                        {
                                                sumT += xx1[i]*d*bess[i]*(pow(xx1[i],2)+
                                                pow((a*beta[kk]),2))/(pow((pow(xx1[i],2)
                                                +pow((a*beta[kk]),2)),2)+1/pow(d,2));
                                                i=i+1;
                                                sumT += xx1[i]*d*bess[i]*(pow(xx1[i],2)+
                                                pow((a*beta[kk]),2))/(pow((pow(xx1[i],2)
                                                +pow((a*beta[kk]),2)),2)+1/pow(d,2));
                                                i=i+1;
                                        }

                                        told=tn[jj];
                                        tn[jj]=(tn[jj]+(bb-aa)*sumT/tnm)/3.0;
TTend1:;
                                        n=n+1;

                                        if(n>MM)
                                                goto TTend2;

                                }
                                while(fabs((tn[jj]-told)/tn[jj]) > conv);
TTend2:;
                                Tsum1=Tsum1+tn[jj];

                                if(jj>=count2)
                                        goto TTend3;
                        }
                        while(checkT < 5);
TTend3:;
                        if(aa==0)
                                checkT=checkT+1;

                        jj=0;
                        do
                        {
                                jj=jj+1;
                                aa=lambda[jj-1];
                                bb=lambda[jj];

                                i=1;
                                for(n=1;n<=MM;n++)
                                {
                                        if(n==1)
                                        {
                                                xx1[i]=0.5*(aa+bb);
                                                i=i+1;
                                                goto qquit5;
                                        }
                                        for(it=1,j=1;j<n-1;j++) it *=3;
                                        tnm=it;
                                        del1=(bb-aa)/(3.0*tnm);
                                        ddel1=del1+del1;
                                        xx1[i]=aa+0.5*del1;
                                        i=i+1;

                                        for(j=1;j<=it;j++)
                                        {
                                                xx1[i]=xx1[i-1]+ddel1;
                                                i=i+1;
                                                xx1[i]=xx1[i-1]+del1;
                                                i=i+1;
                                        }
qquit5:;
                                }

                                for(j=1;j<i;j++)
                                        xx2[j]=rho[rr]*xx1[j];

                                bessj0(xx2,bess,i);

                                i=1;
                                n=1;
                                u1old=0;
                                u2old=0;
                                if(pow(1/d,2)<(conv1*pow((pow(aa,2)
                                        +pow((a*beta[kk]),2)),2)))
                                        goto UUend3;

                                do//Definite Integration between two finite numbers for U
                                {
                                        if(n==1)
                                        {
                                                un1[jj]=(bb-aa)*xx1[i]*bess[i]/(pow((pow(
                                                xx1[i],2)+pow((a*beta[kk]),2)),2)+1/pow(d,
                                                2));
                                                un2[jj]=(bb-aa)*xx1[i]*bess[i]/pow((pow(
                                                xx1[i],2)+pow((a*beta[kk]),2)),2);
                                                i=i+1;
                                                goto UUend1;
                                        }
                                        for(it=1,j=1;j<n-1;j++) it *=3;
                                        tnm=it;
                                        sumU1=0.0;
                                        sumU2=0.0;
                                        for(j=1;j<=it;j++)
                                        {
                                                sumU1 += xx1[i]*bess[i]/(pow((pow(xx1[i]
                                                ,2)+pow((a*beta[kk]),2)),2)+1/pow(d,2));
                                                sumU2 += xx1[i]*bess[i]/pow((pow(xx1[i]
                                                ,2)+pow((a*beta[kk]),2)),2);
                                                i=i+1;
                                                sumU1 += xx1[i]*bess[i]/(pow((pow(xx1[i]
                                                ,2)+pow((a*beta[kk]),2)),2)+1/pow(d,2));
                                                sumU2 += xx1[i]*bess[i]/pow((pow(xx1[i]
                                                ,2)+pow((a*beta[kk]),2)),2);
                                                i=i+1;
                                                }

                                                u1old=un1[jj];
                                                un1[jj]=(un1[jj]+(bb-aa)*sumU1/tnm)/3.0;
                                                u2old=un2[jj];
                                                un2[jj]=(un2[jj]+(bb-aa)*sumU2/tnm)/3.0;
UUend1:;
                                                n=n+1;
                                                if(n>MM)
                                                        goto UUend2;
                                        }
                                        while(fabs((un1[jj]-u1old)/un1[jj]) > conv || fabs((un2[jj]-
                                                u2old)/un2[jj]) > conv );
UUend2:;
                                        Usum1=Usum1+un1[jj];
                                        U2sum=U2sum+un2[jj];

                                        if(jj>=count2)
                                                goto UUend3;
                                }
                                while(checkU < 5);
UUend3:;

                                if(aa==0)
                                        checkU=checkU+1;

                                Ssum2[kk]=Ssum2[kk-1]+KO*Kzetao*d*bessk0(a*rho[rr]*beta[kk]);
                                Tsum2[kk]=Tsum2[kk-1]+KO*Kzetao*(Tsum1+
                                        d*bessk0(a*rho[rr]*beta[kk])-Ssum1);
                                Usum2[kk]=Usum2[kk-1]+KO*Kzetao*(Usum1+rho[rr]/
                                        (2*a*beta[kk])*bessk1(a*rho[rr]*beta[kk])-U2sum);
                        }
                        while(fabs(Ssum2[kk]-Ssum2[kk-1])!=0 || fabs(Tsum2[kk]-Tsum2[kk-1])!=0 ||
                                fabs(Usum2[kk]-Usum2[kk-1])!=0);

                        DC[k][rr]=hO*Ssum2[kk]/(2*pow(d,2));
                        AC[k][rr]=hO*A/(2*pow(d,2))*sqrt(pow(Tsum2[kk],2)+
                                pow(Usum2[kk],2));
                        Phase[k][rr]=tauo+atan2(Usum2[kk],Tsum2[kk]);

                        if(checkL>=5)
                                Phase[k][rr]=Phase[k][rr]+2*pi;

                        if(Usum2[kk]<0 && checkL<5)
                        {
                                Phase[k][rr]=Phase[k][rr]+2*pi;
                                checkL=checkL+1;
                        }

                }

        }
        while(k < 1);


        for (i=0; i<rad; i++)
        sens[i][1]=(DC[1][i]-DC[0][i])*mid_L/MC[i+rrmin]/(max_L-min_L);


for (i=0; i<(3*rad); i++)
{
        for (j=0; j<2; j++)
                outsens<<setprecision(15)<<setw(25)<<sens[i][j];
        outsens<<endl;
}

GS[0]=0;
GS[1]=0;

for (i=0; i<(3*rad); i++)
{
        GS[0]=GS[0]-2*epsilon[i]*sens[i][0];
        GS[1]=GS[1]-2*epsilon[i]*sens[i][1];
}

if (iter==1)
        gam=0;

else
        gam=(GS[0]*(GS[0]-GSold[0])+GS[1]*(GS[1]-GSold[1]))/
                (pow(GSold[0],2)+pow(GSold[1],2));

dk[0]=GS[0]+gam*dkold[0];
dk[1]=GS[1]+gam*dkold[1];

top=0;
bot=0;

for (i=0; i<(3*rad); i++)
{
        top=top+(-(sens[i][0]*dk[0]+sens[i][1]*dk[1])*epsilon[i]);
        bot=bot+pow((sens[i][0]*dk[0]+sens[i][1]*dk[1]),2);
}
step=top/bot;
mid_muspold=mid_musp;
mid_Lold=mid_L;
mid_musp=mid_musp-step*dk[0]*mid_musp;
mid_L=mid_L-step*dk[1]*mid_L;

if(fabs(mid_musp-mid_muspold)/mid_musp<1e-10 && fabs(mid_L-mid_Lold)/mid_L<1e-10)
        term=term+1;

if(term>5)
        goto end;

outpar<<setprecision(10)<<setw(15)<<mid_L<<setw(15)<<mid_musp<<endl;

dkold[0]=dk[0];
dkold[1]=dk[1];
GSold[0]=GS[0];
GSold[1]=GS[1];
```

```
        iter=iter+1;

        goto begin;
end:;
        cout<<"Done"<<endl;
}


double bessj0(double xxx[N],double bess[N],double siz)
{
        double ax,z,x2x,y,ans1,ans2;

        for(int i=1;i<=siz;i++)
        {
                if((ax=fabs(xxx[i])) < 8.0)
                {
                        y=pow(xxx[i],2);
                        ans1=57568490574.0+y*(-13362590354.0+y*(651619640.7+
                                y*(-11214424.18+y*(77392.33017+y*(-184.9052456)))));
                        ans2=57568490411.0+y*(1029532985.0+y*(9494680.718+y*(59272.64853+
                                y*(267.8532712+y*1.0))));
                        bess[i]=ans1/ans2;
                }
                else
                {
                        z=8.0/ax;
                        y=pow(z,2);
                        x2x=ax-0.785398164;
                        ans1=1.0+y*(-0.1098628627e-2+y*(0.2734510407e-4+
                                y*(-0.2073370639e-5+y*0.2093887211e-6)));
                        ans2=-0.1562499995e-1+y*(0.1430488765e-3+y*(-0.691114765e-5+
                                y*(0.7621095161e-6-y*0.934935152e-7)));
                        bess[i]=sqrt(0.636619772/ax)*(cos(x2x)*ans1-z*sin(x2x)*ans2);
                }
        }

        return bess[N];
}

double bessjj0(double xxx)
{
        double ax,z,x2x,y,ans1,ans2,bess;

        if((ax=fabs(xxx)) < 8.0)
        {
                y=pow(xxx,2);
                ans1=57568490574.0+y*(-13362590354.0+y*(651619640.7+
                        y*(-11214424.18+y*(77392.33017+y*(-184.9052456)))));
                ans2=57568490411.0+y*(1029532985.0+y*(9494680.718+y*(59272.64853+
                        y*(267.8532712+y*1.0))));
                bess=ans1/ans2;
        }
        else
        {
                z=8.0/ax;
                y=pow(z,2);
                x2x=ax-0.785398164;
                ans1=1.0+y*(-0.1098628627e-2+y*(0.2734510407e-4+
                        y*(-0.2073370639e-5+y*0.2093887211e-6)));
                ans2=-0.1562499995e-1+y*(0.1430488765e-3+y*(-0.691114765e-5+
                        y*(0.7621095161e-6-y*0.934935152e-7)));
                bess=sqrt(0.636619772/ax)*(cos(x2x)*ans1-z*sin(x2x)*ans2);
        }


        return bess;
}
double bessi0(double xxx)
{
        double ax,ans,y;

        if((ax=fabs(xxx)) < 3.75)
        {
                y=xxx/3.75;
                y*=y;
                ans=1.0+y*(3.5156229+y*(3.0899424+y*(1.2067492+y*(0.2659732+
                        y*(0.360768e-1+y*0.45813e-2)))));
        }
        else
        {
                y=3.75/ax;
                ans=(exp(ax)/sqrt(ax))*(0.39894228+y*(0.1328592e-1+y*(0.225319e-2+y*
                        (-0.157565e-2+y*(0.916281e-2+y*(-0.2057706e-1+
                        y*(0.2635537e-1+y*(-0.1647633e-1+y*0.392377e-2))))))));
        }

        return ans;
}

double bessk0(double xxx)
{
        double bessi0(double xxx);
        double y,ans;

        if (xxx<= 2.0)
        {
                y=xxx*xxx/4.0;
                ans=(-log(xxx/2.0)*bessi0(xxx))+(-0.57721566+y*(0.42278420+y*(0.23069756+
                        y*(0.3488590e-1+y*(0.262698e-2+y*(0.10750e-3+y*0.74e-5))))));
        }
        else
        {
                y=2.0/xxx;
                ans=(exp(-xxx)/sqrt(xxx))*(1.25331414+y*(-0.7832358e-1+y*(0.2189568e-1+
                        y*(-0.1062446e-1+y*(0.587872e-2+y*(-0.251540e-2+y*0.53208e-3))))));
        }

        return ans;
}

double bessi1(double xxx)
{
	double ax,ans,y;

	if((ax=fabs(xxx)) < 3.75)
	{
		y=xxx/3.75;
		y*=y;
		ans=ax*(0.5+y*(0.87890594+y*(0.51498869+y*(0.15084934+
			y*(0.2658733e-1+y*(0.301532e-2+y*0.32411e-3))))));
	}
	else
	{
		y=3.75/ax;
		ans=0.2282967e-1+y*(-0.2895312e-1+y*(0.1787654e-1-y*0.420059e-2));
		ans=0.39894228+y*(-0.3988024e-1+y*(-0.362018e-2+y*(0.163801e-2+
			y*(-0.1031555e-1+y*ans))));
		ans *= (exp(ax)/sqrt(ax));
	}

	return xxx < 0.0 ? -ans : ans;
}

double bessk1(double xxx)
{
	double bessi1(double xxx);
	double y,ans;

	if (xxx<= 2.0)
	{
		y=xxx*xxx/4.0;
		ans=(log(xxx/2.0)*bessi1(xxx))+(1.0/xxx)*(1.0+y*(0.15443144+y*(-0.67278579+
			y*(-0.18156897+y*(-0.1919402e-1+y*(-0.110404e-2+y*(-0.4686e-4)))))));
	}
	else
	{
		y=2.0/xxx;
		ans=(exp(-xxx)/sqrt(xxx))*(1.25331414+y*(0.23498619+y*(-0.3655620e-1+
			y*(0.1504268e-1+y*(-0.780353e-2+y*(0.325614e-2+y*(-0.68245e-3)))))));
	}

	return ans;
}
```