2006

Study of reconstruction algorithms for diffracted observations of Saturn's rings

Katayoun Goudarzi
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Goudarzi, Katayoun, "Study of reconstruction algorithms for diffracted observations of Saturn's rings" (2006). *Master's Theses*. 3001.
DOI: <https://doi.org/10.31979/etd.8ebk-gqbf>
https://scholarworks.sjsu.edu/etd_theses/3001

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

STUDY OF RECONSTRUCTION ALGORITHMS FOR
DIFFRACTED OBSERVATIONS OF SATURN'S RINGS

A Thesis

Presented to

The Faculty of the Department of Electrical Engineering

San Jose State University

In Partial Fulfillment

of the Requirements of the Degree

Master of Sciences

by

Katayoun Goudarzi

December 2006

UMI Number: 1441095

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform 1441095

Copyright 2007 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

© 2006

Katayoun Goudarzi

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF ELECTRICAL ENGINEERING

A handwritten signature in black ink, appearing to read "Essam Marouf", written over a horizontal line.

Dr. Essam Marouf

A handwritten signature in black ink, appearing to read "Robert Morelos-Zaragoza", written over a horizontal line.

Dr. Robert Morelos-Zaragoza

A handwritten signature in black ink, appearing to read "Avtar Singh", written over a horizontal line.

Dr. Avtar Singh

APPROVED FOR THE UNIVERSITY

A handwritten signature in black ink, appearing to read "Rhea Williamson", written over a horizontal line.

ABSTRACT

STUDY OF RECONSTRUCTION ALGORITHMS FOR DIFFRACTED OBSERVATIONS OF SATURN'S RINGS

by Katayoun Goudarzi

Work presented in this thesis describes the need and investigates and implements algorithms for processing data captured by a spacecraft occulting Saturn's rings to reveal detailed ring structure. Measurements of structure that vary on a scale smaller or comparable to the Fresnel scale are diffraction-limited. Following algorithms originally introduced by Marouf *et al.* [1], the thesis presents MATLAB-based computational simulations of diffraction-limited "observations" of idealistic ring profiles and their reconstructions to remove diffraction effects. Relevant theoretical results are re-derived from first principles. The thesis investigates the impact of processing artifacts on reconstructed data, such as impacts of finite data segment and varying Fresnel scale. Two computational approaches to dynamically update the Fresnel scale used in the reconstruction process to achieve a given resolution are developed and tested using actual Cassini spacecraft ring occultation geometry. Also, two methods to explore the possibility of inference of edge thickness are proposed and tested.

Acknowledgements

I would like to express my gratitude towards my advising professor, Dr. Essam Marouf; without his guidance this work would have been impossible.

I would also like to thank my loving husband for his support. His sense of motivation and encouragement made the completion of this thesis possible.

Table of Contents

1	Introduction.....	1
1.1	Overview.....	1
1.2	Radio Occultation	2
1.3	Basic Principles.....	5
1.4	Examples.....	9
2	Fundamentals.....	13
2.1	Fresnel – Huygens Principle	13
2.2	Fresnel Approximation	15
2.3	One-Dimensional Approximation of the Fresnel – Huygens Integral.....	16
2.4	Forward-Fresnel Transform.....	19
2.5	Inverse-Fresnel Transform.....	22
2.6	Impulse Response	23
2.7	Filter Analysis.....	25
2.8	Implementation of the Forward and Inverse-Fresnel Transforms	37
2.9	Verifying Reconstruct.m and SimulateThat.m	39
2.9.1	Impulse Response	40
2.9.2	Gray Edge Response	46
3	Linearly Tapered Edges	54
3.1	Optical Depth.....	54
3.2	Linear Approximation of Optical Depth at Ring Edges	55
3.3	Effect of Ring Thickness on Measured and Reconstructed Profiles.....	59
3.4	Transition Width of a Reconstructed Abrupt Edge Profile.....	65
3.5	Measuring the Ring Thickness.....	67
3.5.1	Direct Width Measurement.....	68
3.5.2	Slope Method.....	72
4	Occultation Geometry and Calculation of Fresnel Scale	77
4.1	Geometry.....	77
4.2	One-Dimensional Approximation of the Measured Transmittance.....	78
4.2.1	Stationary Phase Approximation	82
4.2.2	Quadratic Approximation of Φ	84
4.3	Geometry File Reader	91
5	Effect of Fresnel Scale Uncertainty on Reconstructed Data.....	94
5.1	Effect of Fresnel Scale Mismatch on Reconstructed Data.....	94
5.2	Effect of Fresnel Scale Uncertainty on Radial Resolution	97
5.3	Filter Update Algorithms	106
5.3.1	UpdateF.m.....	107
5.3.2	BlockUpdateF.m	109
5.3.3	TestUpdateAlgorithms.m.....	110
5.4	Effect of Fresnel Scale Mismatch on Impulse Response.....	112
5.5	Effect of Fresnel Scale Mismatch on Edges	121
5.5.1	Phase Distortion.....	122

5.5.2	Magnitude Distortion.....	127
5.6	Applying the Reconstruction Algorithm to Three Gaps with Different Fresnel Scales.....	133
5.6.1	Gaps with the same Widths.....	134
5.6.2	Gaps with Different Widths.....	144
6	Summary and Conclusions	149
	References.....	155
	Appendices: MATLAB Codes for Each Chapter	156
1	MATLAB Codes for Chapter 1	157
1.1	GapandEdgeSimulation.m	157
1.2	GrayGapSimulation.m	161
2	MATLAB Codes for Chapter 2	165
2.1	Fresnel.m.....	165
2.2	CornuSpiral.m.....	165
2.3	InverseFresnelFilter.m	165
2.4	SimulateThat.m.....	167
2.5	Reconstruct.m	168
2.6	Peter Volegov's fcs() function.....	170
2.7	Commands used for compiling the fcs mex file in MATLAB	170
2.8	ImpulseResponse.m	171
2.9	VerifySimulateThat.m	175
2.10	GrayEdgeSimulation.m.....	177
2.11	FilteringArtifact.m	181
3	MATLAB Codes for Chapter 3	184
3.1	LinearlyTaperedEdgeSimulation.m.....	184
3.2	LinearlyTaperedEdgeSimulationComparison.m	188
4	MATLAB Codes for Chapter 4	193
4.1	Code for verifying the Fresnel scale formula.....	193
4.2	Example of contents of a geometry file	197
4.3	GeometryFileReader.m.....	198
5	MATLAB Codes for Chapter 5	201
5.1	VaryingResolution.m.....	201
5.2	NormalizedResolutionAnalysis4VaryingF.m.....	202
5.3	Fresnel scale update routines.m	203
5.3.1	FindFBlock.m (Uses binary search technique).....	203
5.3.2	UpdateF.m (Uses FindFBlock.m).....	203
5.3.3	BlockUpdateF.m	205
5.4	TestUpdateAlgorithms.m.....	206
5.5	ReconstructF.m	207
5.6	ImpulseResponseVaryingF.m.....	209
5.7	GrayEdgeSimulationVaryingF.m	213
5.8	ThreeGrayGapSimulationVaryingF.m.....	217

List of Figures

Figure 1.1 Illustration of object and image plane configuration.....	5
Figure 1.2 Image plane transmittance magnitude of a gap and an edge in a gray screen with optical depth $\tau = 8$	9
Figure 1.3 Reconstructed object plane transmittance magnitude of a gap and an edge in a gray screen with optical depth $\tau = 8$	10
Figure 1.4 Image plane transmittance magnitude of a gap in a gray screen with optical depth $\tau = 8$	11
Figure 1.5 Reconstructed object plane transmittance magnitude of a gap in a gray screen with optical depth $\tau = 8$	12
Figure 2.1 Illustration of Fresnel-Huygens principle.....	14
Figure 2.2 Illustration of the object and image plane configuration.....	16
Figure 2.3 Cornu spiral	19
Figure 2.4 Idealized normal incidence geometry.....	20
Figure 2.5 Magnitude response of the inverse-Fresnel filter windowed by a rectangular window.....	29
Figure 2.6 A closer look at Figure (2.5).....	30
Figure 2.7 A closer look at Figure (2.6).....	31
Figure 2.8 A closer look at Figure (2.7).....	32
Figure 2.9 A closer look at Figure (2.8).....	32
Figure 2.10 Magnitude of frequency response of the inverse-Fresnel filter windowed by a Kaiser window of $\beta = 2.5$	33
Figure 2.11 A closer look at Figure (2.10).....	34
Figure 2.12 A closer look at Figure (2.11).....	36
Figure 2.13 Magnitude of frequency response of the inverse-Fresnel filter windowed by a Kaiser window of $\beta = 4$	36
Figure 2.14 Illustration of inverse-Fresnel filtering of the image plane transmittance....	37
Figure 2.15 Phase of the theoretical versus the reconstructed impulse response, normalized resolution of 0.1.	41
Figure 2.16 Magnitude of the theoretical versus the reconstructed impulse response in dB, normalized resolution of 0.1.....	41
Figure 2.17 A closer look at Figure (2.16).....	42
Figure 2.18 A closer look at Figure (2.17).....	42
Figure 2.19 Phase of the theoretical versus the reconstructed impulse response, normalized resolution of 0.2.	43
Figure 2.20 Magnitude of the theoretical versus the reconstructed impulse response in dB, normalized resolution of 0.2.....	43
Figure 2.21 A closer look at Figure (2.20).....	44
Figure 2.22 Magnitude of the theoretical impulse response in dB for a rectangular window versus the reconstructed impulse response for a Kaiser window of $\beta = 4$..	45
Figure 2.23 A closer look at Figure (2.22).....	45

Figure 2.24 Comparison of the expected and the simulated image plane transmittance phase of a gray edge, $\tau = 8$.	47
Figure 2.25 Comparison of the expected and the simulated image plane transmittance magnitude of a gray edge, $\tau = 8$.	48
Figure 2.26 Comparison of the expected and the simulated image plane transmittance magnitude of a gray edge in dB, $\tau = 8$.	48
Figure 2.27 The difference of the image plane transmittance magnitudes of a gray edge obtained by theory and simulation, $\tau = 8$.	49
Figure 2.28 Comparison of the object, image and reconstructed object plane transmittance magnitudes of a gray edge in dB.	49
Figure 2.29 Comparison of the Object, image and reconstructed object plane transmittance magnitudes of a gray edge.	50
Figure 2.30 Comparison of the object, image, and reconstructed object plane transmittance phases of a gray edge.	50
Figure 2.31 Phase of the reconstructed object plane transmittance of a gray edge	51
Figure 2.32 Illustration of theoretical versus simulated image plane transmittance of a gray edge.	52
Figure 2.33 Illustration of reconstruction artifact for a gray edge.	52
Figure 2.34 A closer look at Figure (2.33).	53
Figure 3.1 Ray incident on ring plane at an angle B .	54
Figure 3.2 Illustration of optical depth at ring edges	55
Figure 3.3 Profile of a sharp edge	58
Figure 3.4 Profile of a linearly tapered edge with normalized $L = 1$.	58
Figure 3.5 Illustration of measured image plane transmittance magnitudes and their reconstructed profiles for linearly tapered edges with normalized $L = 0, 1, 2, 3$.	60
Figure 3.6 Measured image plane transmittance magnitudes in dB for linearly tapered edges of $L = 0, 1, 2, 3$.	61
Figure 3.7 Reconstructed object plane transmittance magnitudes in dB for linearly tapered edges of $L = 0, 1, 2, 3$.	61
Figure 3.8 Measured image plane transmittance magnitudes for linearly tapered edges of $L = 0, 1, 2, 3$.	62
Figure 3.9 Reconstructed object plane transmittance magnitudes for linearly tapered edges of $L = 0, 1, 2, 3$.	62
Figure 3.10 Phase of the image plane transmittance for linearly tapered edges of $L = 0, 1, 2, 3$.	63
Figure 3.11 Reconstructed phase of the object plane transmittance for linearly tapered edges of $L = 0, 1, 2, 3$.	63
Figure 3.12 A closer look at Figure (3.6).	64
Figure 3.13 Comparison of the reconstructed profile of an abrupt edge versus the original edge profile.	65
Figure 3.14 A closer look at Figure (3.13).	66
Figure 3.15 An example for using the "direct width measurement" method for measuring ring thickness.	69
Figure 3.16 A closer look at Figure (3.17).	70

Figure 3.17 An example for using the "direct width measurement" method for measuring ring thickness.....	70
Figure 3.18 A closer look at Figure (3.19).....	71
Figure 3.19 An example for using the "direct width measurement" method for measuring ring thickness.....	71
Figure 3.20 Profile of a linearly tapered edge with a normalized projected ring thickness of 0.5.	73
Figure 3.21 An example for using the "slope method" for measuring ring thickness. ...	73
Figure 3.22 Illustration of how the slopes of the reconstructed profiles and the actual profiles are almost the same in some regions and differ in other regions.....	74
Figure 3.23 An example for using the "slope method" for measuring ring thickness. ...	75
Figure 3.24 An example for using the "slope method" for measuring ring thickness. ...	75
Figure 3.25 An example for using the "slope method" for measuring ring thickness. ...	76
Figure 3.26 An example for using the "slope method" for measuring ring thickness. ...	76
Figure 4.1 Radio occultation geometry in a Saturnocentric coordinate system [1].....	78
Figure 4.2 Fresnel scale calculated from Cassini Rev7, May 2-3, 2005 (DOY 122-123) geometry file	92
Figure 4.3 Velocity of Cassini spacecraft calculated from geometry file parameters (Rev 7).	93
Figure 5.1 System illustrating the relationships between the ring profile $T(x)$, measured diffraction-limited ring profile $\hat{T}(x)$, and the reconstructed ring profile $\tilde{T}(x)$	95
Figure 5.2 Resolution change versus filter length in the presence of an error in the knowledge of the characteristic Fresnel scale.....	102
Figure 5.3 Maximum achievable normalized resolution (with respect to F) in the presence of error in the characteristic Fresnel scale.	104
Figure 5.4 Normalized filter width required for obtaining the maximum normalized resolution in the presence of an error in the characteristic Fresnel scale.....	105
Figure 5.5 Number of filter sample points required for obtaining the maximum achievable normalized resolution in the presence of an error in the characteristic Fresnel scale.....	106
Figure 5.6 Sample block of data measured by Cassini	108
Figure 5.7 Block in Figure (5.6) reduced in length.....	108
Figure 5.8 Block in Figure (5.7) expanded in length.....	109
Figure 5.9 Block in Figure (5.8) reduced in length.....	109
Figure 5.10 Reconstructed magnitude(dB) of an impulse response, using a filter of 100 km wide, with and without error in F.	113
Figure 5.11 Reconstructed magnitude of an impulse response, using a filter of 100 km wide, with and without error in F.....	113
Figure 5.12 Reconstructed phase (degrees) of an impulse response, using a filter of 100 km wide, with and without error presence in F.	114
Figure 5.13 A closer look at Figure (5.12).....	114

Figure 5.14 Reconstructed magnitude(dB) of an impulse response, using a filter of 100 km wide, with and without error presence in F.	116
Figure 5.15 Reconstructed magnitude of an impulse response, using a filter of 100 km wide, with and without error presence in F.....	116
Figure 5.16 Reconstructed phase (degrees) of an impulse response, using a filter of 100 km wide, with and without error presence in F.	117
Figure 5.17 Closer look at Figure (5.16).....	117
Figure 5.18 Reconstructed phase (degrees) of an impulse response, using a filter of 100 km wide, with and without error presence in F	119
Figure 5.19 A closer look at Figure (5.18).....	119
Figure 5.20 A closer look at Figure (5.19).....	120
Figure 5.21 Reconstructed phase (degrees) of an impulse response, using a filter of 100 km wide, with no error in F	120
Figure 5.22 A closer look at Figure (5.21).....	121
Figure 5.23 Comparison of reconstructed phase versus simulated diffraction-limited phase and the actual phase of an edge, for $\frac{\delta F}{F} = 0.2, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	123
Figure 5.24 Comparison of reconstructed phase versus simulated diffraction-limited phase and the actual phase of an edge, for $\frac{\delta F}{F} = 0.01, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	124
Figure 5.25 Comparison of reconstructed phase versus simulated diffraction-limited phase and the actual phase of an edge, for $\frac{\delta F}{F} = 0.001, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	124
Figure 5.26 A closer look at Figure (5.25).....	125
Figure 5.27 Phase of an edge reconstructed from simulated diffraction-limited data for $\frac{\delta F}{F} = 10^{-7}, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	125
Figure 5.28 Phase of an edge reconstructed from theoretical diffraction-limited data for $\frac{\delta F}{F} = 10^{-7}, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	126
Figure 5.29 Phase of an edge reconstructed from theoretical diffraction-limited data for $\frac{\delta F}{F} = 0, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	126
Figure 5.30 Phase of an edge reconstructed from simulated diffraction-limited data for $\frac{\delta F}{F} = 0, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	127
Figure 5.31 Comparison of reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude of an edge for $\frac{\delta F}{F} = 0.2, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	128

Figure 5.32 Comparison of reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude of an edge for $\frac{\delta F}{F} = 0.01, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	128
Figure 5.33 A closer look at Figure (5.32).....	129
Figure 5.34 Comparison of reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude of an edge for $\frac{\delta F}{F} = 0.001, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	129
Figure 5.35 A closer look at Figure (5.34).....	130
Figure 5.36 Comparison of reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude of an edge for $\frac{\delta F}{F} = 0.2, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	130
Figure 5.37 Comparison of reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude of an edge for $\frac{\delta F}{F} = 0.01, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	131
Figure 5.38 A closer look at Figure (5.37), highlighting the overshoot in the free-space region.	131
Figure 5.39 A closer look at Figure (5.37), highlighting the undershoot in the gray ring region.	132
Figure 5.40 Comparison of reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude of an edge for $\frac{\delta F}{F} = 0.001, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$	132
Figure 5.41 A closer look at Figure (5.40).....	133
Figure 5.42 Comparison of the reconstructed phase versus diffraction-limited and actual phase for three gaps with the same width of 100km, but different Fresnel scales. A reconstruction algorithm that updates the Fresnel scale has been used..	135
Figure 5.43 A closer look at Figure (5.42).....	135
Figure 5.44 Comparison of reconstructed magnitude versus diffraction-limited and actual magnitude for three gaps with the same width of 100km, but different Fresnel scales. A reconstruction algorithm that updates the Fresnel scale has been used.....	136
Figure 5.45 A closer look at the reconstructed profile of the first gap in Figure (5.44).	136
Figure 5.46 A closer look at the reconstructed profile of the second gap in Figure (5.44).	137
Figure 5.47 A closer look at the reconstructed profile of the third gap in Figure (5.44)	137
Figure 5.48 Reconstructed magnitude for three gaps with the same width of 100km, but different Fresnel scales. A reconstruction algorithm that updates the Fresnel scale has been used.	138

Figure 5.49 A closer look at the reconstructed profile of the first gap in Figure (5.48).	138
Figure 5.50 A closer look at the reconstructed profile of the second gap in Figure (5.48).	139
Figure 5.51 A closer look at the reconstructed profile of the third gap in Figure (5.48).	139
Figure 5.52 Comparison of reconstructed phase versus diffraction-limited and actual phase for three gaps with the same width of 100 km, but different Fresnel scales. A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.	140
Figure 5.53 A closer look at Figure (5.52).	140
Figure 5.54 A close look at the reconstructed magnitude profile of the first gap of Figure (5.44). A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.	141
Figure 5.55 A close look at the reconstructed magnitude profile of the second gap of Figure (5.44). A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.	141
Figure 5.56 A close look at the reconstructed magnitude profile of the third gap of Figure (5.44). A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.	142
Figure 5.57 A close look at the reconstructed magnitude profile of the first gap of Figure (5.44). A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.	142
Figure 5.58 A close look at the reconstructed magnitude profile of the second gap of Figure (5.44). A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.	143
Figure 5.59 A close look at the reconstructed magnitude profile of the third gap of Figure (5.44). A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.	143
Figure 5.60 Comparison of reconstructed phase versus diffraction-limited and actual phase for three gaps with widths of 100(10F), 10(F), and 5(F/2) km respectively, but different Fresnel scales. A reconstruction algorithm that updates the Fresnel scale has been used.	144
Figure 5.61 A closer look at the first gap (located at 2000-2100km) in Figure (5.60).	145
Figure 5.62 A closer look at the second gap (located at 4000-4010km) in Figure (5.60).	145
Figure 5.63 A closer look at the third gap (located at 6000-6005km) in Figure (5.60).	146
Figure 5.64 Comparison of reconstructed magnitude versus diffraction-limited and actual magnitude for three gaps with widths of 100(10F), 10(F), and 5(F/2) km respectively, but different Fresnel scales. A reconstruction algorithm that updates the Fresnel scale has been used.	146
Figure 5.65 A closer look at the first gap (located at 2000-2100km) in Figure (5.64).	147
Figure 5.66 A closer look at the second gap (located at 4000-4010km) in Figure (5.64).	147
Figure 5.67 A closer look at the third gap (located at 6000-6005 km) in Figure (5.64).	148

1 Introduction

Humans have long been curious to explore and understand the universe beyond Earth. The mission to fulfill their curiosity began by the making of telescopes, and, with the recent advancements in technology, has evolved into sending sophisticated spacecrafts into outer space to collect data to analyze and help understand a small fraction of what has thus far been unknown. Pioneer 10 and 11 launched over 30 years ago, Voyager I and II launched almost thirty years ago, and Cassini which launched almost 10 years ago (and is presently orbiting Saturn) are examples of such attempts.

These spacecrafts have revealed many unknowns about the planets of our solar system. Both the Voyager spacecrafts have now reached the extremes of our solar system and are both still communicating with Earth. Cassini's mission is to study Saturn, its rings, satellites, and environment. Cassini entered into orbit around Saturn in July 2004, and, its nominal mission extends to June 2008. The work presented in this thesis is motivated by the Voyager and Cassini missions and their visit to Saturn, and Saturn's remarkable ring system.

1.1 Overview

This chapter briefly discusses radio occultation and how this technique is used to profile Saturn's ring structure. Also the need to process the data collected by the spacecraft to remove diffraction effects in order to extract ring features is discussed followed by supporting examples. The main body of this thesis starts in Chapter 2. Following approaches introduced in [1], Chapter 2 shows that the data collected by the

spacecraft is diffraction-limited and discusses in detail the Fresnel-Huygens principle and derives the forward and inverse-Fresnel transforms. Forward-Fresnel transform models the existence of diffraction in the measured data and inverse-Fresnel transform is used to eliminate diffraction. Both of these transforms are implemented in MATLAB and tested for correctness. An extensive study of the inverse-Fresnel filter (reconstruction filter), which needs to be applied to the measured data, via the inverse-Fresnel transform, for minimizing diffraction is carried out. Chapter 2 also introduces achievable radial resolution, discusses the practical constraints imposed on the inverse-Fresnel filtering algorithm, and studies the artifacts that these limitations introduce into the reconstructed profile. Chapter 3 discusses two different methods for measuring the thickness of rings. Chapter 4 introduces the occultation geometry and re-derives the corresponding Fresnel scale of diffraction, originally reported in [1]. Chapter 5 discusses how uncertainties in the occultation geometry could lead to wrong reconstructed ring profiles. It proposes methods for suppressing the effects of the uncertainties on the reconstructed profiles followed by supporting examples. The main results and conclusions are summarized in Chapter 6.

1.2 Radio Occultation

Radio Occultation is a measurement technique used to reveal the radial distribution of planetary ring material, that is, ring structure [1]. A spacecraft flying behind the rings transmits a coherent sinusoidal radio signal, which travels through the rings, and, its perturbed parameters are then measured on the Earth. Ring particles extinguish and scatter the transmitted signal, hence, attenuating the sinusoid amplitude

and changing its phase. A planetary ring acts as an enormously extended thin diffraction screen of some radially varying complex transmittance [1]. The phenomenon of rings perturbing the transmitted signal and therefore being able to study the ring's structure and features by measuring the perturbed received complex signal is known as radio occultation.

Even though the intensity and phase of the received complex signal reveal many features and characteristics of the rings, if the variation in the structure of the ring occurs over a scale comparable to or smaller than the Fresnel scale of diffraction, the observation will be diffraction limited. This means that many features such as sharp edges, narrow gaps, narrow ringlets, density and bending waves, and other unknown phenomena yet to be discovered, may not be revealed. Using a reconstruction algorithm known as "inverse-Fresnel filtering" reduces the effect of diffraction and reveals more features of the rings that are hidden within the measured diffracted signal.

As discussed in more detail in [1], many factors affect the reconstruction process of inverse-Fresnel filtering. Some of these factors are: uncertainties in the geometry (variant Fresnel scale caused by movement of the spacecraft), system phase stability, system thermal noise, finite width of the data window, and analytical approximations used. These factors limit the resolution and also introduce artifacts in the signal intensity and phase reconstructed by the algorithm. Hence, it is of significant importance to study the nature of at least some of these artifacts and to investigate methods to improve resolution limitations.

As stated above, a coherent sinusoidal radio signal, which is transmitted by the spacecraft, travels through the rings and the perturbations are then measured on the Earth. For analytical purposes, it is more convenient to assume that the signal is transmitted from the Earth, perturbed by the rings acting as a one dimensional diffraction screen, and then measured by the spacecraft. By reciprocity, this configuration should yield the same results.

Following the formulation in [1] and [2], in the plane of a 1-D diffraction screen, the transmittance at point x_o is defined as,

$$T(x_o) = \frac{E(x_o)}{E_o} \quad (1.1)$$

$E(x_o)$: Electric field at point x_o in the presence of the ring (diffraction screen)

E_o : Electric field at point x_o in the absence of the ring (diffraction screen)

x_o : Distance index in the object plane

The plane in which the ring (diffraction screen) exists is referred to as the “object plane.” The spacecraft observes or measures the diffracted signal as a function of distance x_i in a plane parallel to the object plane at some distance away. This plane is referred to as the “image plane.”

Figure (1.1) illustrates the configuration between the object and image planes [2]. For simplicity, normal incidence is assumed here. More general oblique incidence geometry is considered later.

Figure 1.1 Illustration of object and image plane configuration

1.3 Basic Principles

Starting from the Fresnel-Huygens principle and using some approximations, it can be shown that [1]

$$\hat{T}(x_i) = \frac{1-j}{2F} \int_{-\infty}^{\infty} T(x_o) e^{j\frac{\pi}{2} \left[\frac{x_i - x_o}{F} \right]^2} dx_o \quad (1.2)$$

$\hat{T}(x)$: Transmittance measured by the spacecraft in the image plane

$T(x)$: Transmittance of the object plane (diffraction screen)

F : Fresnel scale which is dependent on the wavelength of the transmitted signal and the geometry of the observation

x_i : Distance in the image plane

The convolution integral defined in Eq. (1.2) is known as the “forward-Fresnel transform.” A more detailed derivation, including the approximations used, is given in [2].

$\hat{T}(x)$ is the measurement taken by the spacecraft, which does to a certain extent reveal some features in the rings. However, as mentioned earlier, this measurement is diffraction-limited. The desired ring transmittance is $T(x)$. Therefore, algorithms must be devised to recover $T(x)$ from the convolution integral in Eq. (1.2).

If

$$h_f(x) = \frac{1-j}{2F} e^{j\frac{\pi}{2}[\frac{x}{F}]^2} \quad (1.3)$$

then

$$\hat{T}(x) = T(x) * h_f(x) \quad (1.4)$$

where ‘*’ denotes convolution.

It can easily be shown that,

$$T(x) = \hat{T}(x) * h_I(x) \quad (1.5)$$

where

$$h_I(x) = \frac{1+j}{2F} e^{-j\frac{\pi}{2}[\frac{x}{F}]^2} \quad (1.6)$$

The convolution integral defined in Eq. (1.7) is known as the “inverse-Fresnel transform;” $h_I(x)$ is known as the “inverse-Fresnel filter.” The inverse-Fresnel transform,

$$T(x_o) = \frac{1+j}{2F} \int_{-\infty}^{\infty} \hat{T}(x_i) e^{-j\frac{\pi}{2}[\frac{x_o-x_i}{F}]^2} dx_i \quad (1.7)$$

may then be used to reconstruct the ring transmittance $T(x)$ from the diffraction-limited $\hat{T}(x)$.

However, this reconstruction algorithm can not practically be used. The first issue is the limit of the integral, which is infinity, but in reality only limited data can be processed. Assuming that a data segment of length W is processed, Eq. (1.7) can be rewritten as

$$\tilde{T}(x_o) = \frac{1+j}{2F} \int_{x_o-W/2}^{x_o+W/2} \hat{T}(x_i) e^{-j\frac{\pi}{2}[\frac{x_o-x_i}{F}]^2} dx_i \quad (1.8)$$

If

$$T(x_o) = \delta(x_o - x^*)$$

which represents a very narrow feature in the ring plane, it can be shown using Eq. (1.2) and then Eq. (1.8) that the reconstructed object plane transmittance is [2]

$$\tilde{T}(x_o) = \frac{W}{2F^2} \text{sinc}\left[\frac{W}{2F^2}(x_o - x^*)\right] e^{j\frac{\pi}{2}[\frac{x_o-x^*}{F}]^2} \quad (1.9)$$

As observed in Eq. (1.9), the reconstructed object plane transmittance is no longer a delta function; but, it is a sinc function that has a main-lobe of certain width that peaks at the correct location, with side lobes (known as leakage). The width of the main-lobe defines the resolution; the narrower the main-lobe is, the finer the resolution will be. To reduce leakage, $h_l(x)$ can be windowed (multiplied by a tapered function) so that Eq. (1.8) would become

$$\tilde{T}(x_o) = \frac{1+j}{2F} \int_{x_o-W/2}^{x_o+W/2} \hat{T}(x_i) w(x_o - x_i) e^{-j\frac{\pi}{2}[\frac{x_o-x_i}{F}]^2} dx_i \quad (1.10)$$

Nonetheless, one must keep in mind that lowering the side lobe levels via windowing will worsen the resolution [3]. Therefore, the tradeoff between leakage and resolution should always be taken into consideration.

As briefly discussed above, the width of the main lobe determines the achievable radial resolution. It can be shown that the 3dB width of the main lobe, which is taken as a measure of resolution, is [1]

$$\Delta R_w = \frac{2F^2}{W} \quad (1.11)$$

As observed by Eq. (1.11), a longer data segment must be used in order to achieve finer resolution. This is a challenge for very large data blocks. Therefore, for efficient implementation, Fourier transforms are used to compute the convolutional integrals. Other factors also play a role in Eq. (1.8), such as, uncertainties in the geometry (variant Fresnel scale caused by movement of the spacecraft), system phase stability, system thermal noise, and phase approximations used. A more general form of Eq. (1.8) would therefore be [1, 2]

$$\tilde{T}(x_o) = \frac{1+j}{2(F+\delta F)} \int_{x_o-W/2}^{x_o+W/2} [\hat{T}(x_i) + \hat{n}(x_i)] e^{j\phi_r(x_i)} e^{-j\frac{\pi}{2}[\frac{x_o-x_i}{F+\delta F}]^2} dx_i \quad (1.12)$$

$\hat{n}(x_i)$: Additive thermal noise

$\phi_r(x_i)$: A random reference phase modeling the phase instability of the system

δF : Error caused in F because of the change in the observation geometry

Another point of consideration is that $\frac{\pi}{2}[(x_o - x_i)/F]^2$ is a quadratic

approximation of the actual observation phase, and this also sets a limit on the resolution.

To improve the resolution, higher order approximations of the actual phase should be used [1].

1.4 Examples

By examining the following figures, which have been generated by implementing the forward-Fresnel transform in Eq. (1.2) and the reconstruction algorithm in Eq. (1.10) in MATLAB, assuming a Kaiser window of parameter $\beta=3.5$ [3], it becomes clear how “inverse-Fresnel filtering” can reduce diffraction effects and improve the achievable resolution.

Figure 1.2 Image plane transmittance magnitude of a gap and an edge in a gray screen with optical depth $\tau = 8$ (Voyager-like observation of the Keeler gap and the edge of ring A). This is the Ring profile measured by the spacecraft. (See Appendix 1.1, GapandEdgeSimulation.m)

Figure 1.2 shows $|\hat{T}(x_i)|$ versus x_i for a gap and an edge. The effect of diffraction is very much visible. Here, $F=15$ km, the width of the gap is $2.4F$, and, the distance of the edge from the center of the gap is $18.5F$ (Voyager-like observation of the Keeler gap and the edge of Ring A).

Figure 1.3 Reconstructed object plane transmittance magnitude of a gap and an edge in a gray screen with optical depth $\tau = 8$ (Keeler gap and the edge of ring A). A Kaiser window of $\beta = 3.5$ is used for reconstruction. (See Appendix 1.1, GapandEdgeSimulation.m)

Figure (1.3) shows the reconstructed $|\tilde{T}(x_o)|$ versus x_o . It is seen that the gap and the sharp edge are much more visible when the effect of diffraction has been reduced by inverse-Fresnel filtering. The “true” width of the gap and the edge location are revealed.

Here is another example of a gap of width $21.1F$, where $F=15$ km (Voyager-like observation of the Encke gap).

Figure 1.4 Image plane transmittance magnitude of a gap in a gray screen with optical depth $\tau = 8$.
(See Appendix 1.2, GrayGapSimulation.m)

Figure 1.5 Reconstructed object plane transmittance magnitude of a gap in a gray screen with optical depth $\tau = 8$. A Kaiser window of $\beta = 3.5$ is used for reconstruction. (See Appendix 1.2, GrayGapSimulation.m)

It is obvious from the two preceding examples that a simple implementation of Eq. (1.10), which is applying an inverse-Fresnel filter to the measured data, can significantly eliminate the effect of diffraction and recover the fine features of the rings, such as, sharp edges. Also, the width of the gaps is more accurately measurable. Therefore, by inverse-Fresnel filtering the measured diffraction-limited data, more characteristics of the rings are revealed.

2 Fundamentals

This chapter details the derivation of forward and inverse-Fresnel transforms starting from the Fresnel-Huygens integral. These two transforms are implemented in MATLAB and tested for precision. The concept of resolution is also introduced and explained in detail, followed by examples that demonstrate how the reconstruction algorithm works, its effectiveness in removing diffraction from measured profiles, and the artifacts it introduces into the reconstructed profiles.

2.1 Fresnel – Huygens Principle

The Fresnel – Huygens principle states that if a scalar field U impinges on an opaque screen containing an aperture, each point within the aperture, itself, will act as a secondary generating source of scalar fields. Therefore, an observer at point p_i will experience a scalar field which is a superposition of scalar fields created by every single point in the aperture [2, 4].

Figure 2.1 Scalar waves impinging on an opaque screen containing an aperture

$U(p_o)$: The scalar field on aperture Σ in the absence of the opaque screen

$U(p_i)$: The scalar field observed at point p_i

$\vec{r}_{io}$: The spatial vector from p_i to p_o

$\hat{n}$: A unit vector normal to Σ at p_o , pointing away from the observer

$$U(p_i) = \iint_{\Sigma} \frac{1}{j\lambda} U(p_o) \cos(\hat{n}, \hat{r}_{io}) \frac{e^{jk r_{io}}}{r_{io}} ds \quad (2.1)$$

where,

$$k = \frac{2\pi}{\lambda} \quad (2.2)$$

λ : Wave length

k : Wave number

$\cos(\hat{n}, \hat{r}_{io})$: Obliquity factor

2.2 Fresnel Approximation

A simpler form of the Fresnel – Huygens integral, Eq. (2.1), can easily be derived using the paraxial approximations.

Assuming the configuration in Figure (2.2), if

$$z_i \gg \sqrt{(x_i - x_o)^2 + (y_i - y_o)^2} \quad (2.3)$$

holds true for all points of interest in the object and image planes, then the paraxial approximations (2.4), (2.5), and (2.11) can be used to simplify the Fresnel – Huygens integral into Eq. (2.12).

$$\frac{1}{r} \approx \frac{1}{z_i} \quad (2.4)$$

$$\cos(\hat{n}, \hat{r}_{io}) = \frac{z_i}{r} \approx 1 \quad (2.5)$$

$$e^{jkr_{io}} = e^{jkr} = e^{jk\sqrt{z_i^2 + (x_i - x_o)^2 + (y_i - y_o)^2}} = e^{jkz_i \sqrt{1 + \frac{(x_i - x_o)^2 + (y_i - y_o)^2}{z_i^2}}} \quad (2.6)$$

If

$$\alpha = \frac{(x_i - x_o)^2 + (y_i - y_o)^2}{z_i^2} \quad (2.7)$$

then

$$e^{jkr_{io}} = e^{jkz_i \sqrt{1 + \alpha}} \quad (2.8)$$

where according to Eq. (2.3)

$$\alpha \ll 1 \quad (2.9)$$

Figure 2.2 Illustration of the object and image plane configuration

Using the Taylor series expansion, if $\alpha \ll 1$

$$\sqrt{1+\alpha} \approx 1 + \frac{\alpha}{2} - \frac{1}{8}\alpha^2 + \dots \quad (2.10)$$

By applying the second order approximation, Eq. (2.6) will become

$$e^{jkr_{io}} = e^{jkz_i} e^{j\frac{k}{2z_i}[(x_i-x_o)^2+(y_i-y_o)^2]} \quad (2.11)$$

Therefore, the Fresnel – Huygens integral, Eq. (2.1), would be simplified to

$$U(x_i, y_i) = \frac{e^{jkz_i}}{j\lambda z_i} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} U(x_o, y_o) e^{j\frac{k}{2z_i}[(x_i-x_o)^2+(y_i-y_o)^2]} dx_o dy_o \quad (2.12)$$

2.3 One-Dimensional Approximation of the Fresnel – Huygens Integral

As explained in Chapter 1, a planetary ring acts as an enormously extended, thin and complex diffraction screen. This screen can be modeled in the x-y plane, extending from $-\infty$ to $+\infty$ in the y dimension, and having some limited width in the x dimension.

Therefore, the object plane field is independent of the y axis and is represented as

$$U(x_o, y_o) = U(x_o)U(y_o) \quad (2.13)$$

where

$$U(y_o) = 1 \quad (2.14)$$

For a plane wave incident normal to the diffraction screen, the Fresnel scale F is

$$F = \sqrt{\frac{\lambda z_i}{2}} \quad (2.15)$$

Using equations (2.13), (2.14), and (2.15), Eq. (2.12) may be rewritten as

$$U(x_i, y_i) = \frac{e^{jkz_i}}{2j} \int_{-\infty}^{+\infty} e^{j\pi/2[(\frac{y_i - y_o}{F})^2]} \frac{dy_o}{F} \int_{-\infty}^{+\infty} U(x_o) e^{j\pi/2[(\frac{x_i - x_o}{F})^2]} \frac{dx_o}{F} \quad (2.16)$$

If

$$\zeta = \frac{(y_i - y_o)}{F} \quad (2.17)$$

then Eq. (2.16) becomes

$$U(x_i, y_i) = \frac{e^{jkz_i}}{2j} \int_{-\infty}^{+\infty} e^{j\pi/2\zeta^2} d\zeta \int_{-\infty}^{+\infty} U(x_o) e^{j\pi/2[(\frac{x_i - x_o}{F})^2]} \frac{dx_o}{F} \quad (2.18)$$

where the solution to integral

$$\int_0^w e^{j\pi/2\zeta^2} d\zeta = C(w) + jS(w) = \bar{V}(w) \quad (2.19)$$

is a vector in the $C - S$ plane as illustrated below.

The integral in Eq. (2.19) is known as the “Fresnel integral,” and, is computed numerically. The C and S components are known as the “Fresnel-Cosine” and “Fresnel-Sine” integrals, respectively. Numerical values of the Fresnel integral have been plotted as a function of C and S in Figure (2.3) for $w = [-20:20]$. This plot is known as the Cornu spiral. The corresponding MATLAB codes `Fresnel.m` and `CornuSpiral.m` used for generating this figure can be found at Appendix 2.1 and Appendix 2.2, respectively.

From the Cornu spiral,

$$\int_{-\infty}^{+\infty} e^{j\pi/2 \zeta^2} d\zeta = 1 + j \quad (2.20)$$

and hence, Eq. (2.18) simplifies to

$$U(x_i, y_i) = U(x_i) = \frac{1-j}{2F} \left[\int_{-\infty}^{+\infty} U(x_o) e^{j\pi/2 \left[\frac{(x_i - x_o)^2}{F} \right]} dx_o \right] e^{jkz_i} \quad (2.21)$$

After normalization, Eq. (2.21) reduces to the forward-Fresnel transform in Eq. (1.2), as shown below.

Figure 2.3 Cornu spiral

2.4 Forward-Fresnel Transform

As previously explained in the introduction, the Cassini spacecraft transmits a coherent signal to the Earth, which is perturbed by the rings. However, for analytical purposes, it is more convenient to assume the opposite, which, according to the reciprocity theorem, yields the same results. The assumed configuration is shown in Figure (2.4) below [2]

Figure 2.4 Idealized normal incidence geometry

In the plane of a 1-D diffraction screen, the transmittance at point x_o is defined as [2]

$$T(x_o) = \frac{E(x_o)}{E_o} = \frac{U(x_o)}{U_{inc}} \quad (2.22)$$

$E(x_o)$: Electric field at point x_o in the presence of the ring (diffraction screen)

E_o : Electric field at point x_o in the absence of the ring (diffraction screen)

x_o : Distance in the object plane

Thus, transmittance at point x_i in the image plane can be defined as [2]

$$\hat{T}(x_i) = \frac{E(x_i)}{E_o e^{jkD}} \quad (2.23)$$

By using equations (2.22) and (2.23) in Eq. (2.21), it is derived that

$$\hat{T}(x_i) = \frac{1-j}{2F} \int_{-\infty}^{\infty} T(x_o) e^{j\frac{\pi}{2}[\frac{x_i-x_o}{F}]^2} dx_o \quad (2.24)$$

$\hat{T}(x)$: Transmittance measured by the spacecraft in the image plane

$T(x)$: Transmittance of the object plane (diffraction screen)

F : Fresnel scale which is dependent on the wavelength of the transmitted signal and the geometry of the observation

x_i : Distance in the image plane

If

$$h_f(x) = \frac{1-j}{2F} e^{j\frac{\pi}{2}[\frac{x}{F}]^2} \quad (2.25)$$

then

$$\hat{T}(x) = T(x) * h_f(x) \quad (2.26)$$

where ‘*’ denotes convolution and $h_f(x)$ is known as the forward-Fresnel filter.

The convolution integral defined in Eq. (2.24) is known as the forward-Fresnel transform.

2.5 Inverse-Fresnel Transform

$\hat{T}(x)$ is the measurement taken by the spacecraft, which reveals only some of the ring features; because, this measurement is diffraction-limited. The desired ring transmittance is thereby $T(x)$. Consequently, algorithms must be devised to recover $T(x)$ from $\hat{T}(x)$.

Taking the Fourier transform of Eq. (2.26) results in

$$\mathfrak{F}(\hat{T}(x)) = \mathfrak{F}(T(x))\mathfrak{F}(h_f(x)) \quad (2.27)$$

where

$$\mathfrak{F}(h_f(x)) = H_f(u) = \frac{1-j}{\sqrt{2}} e^{-j2\pi F^2 u^2} \quad (2.28)$$

As a consequence of Eq. (2.27),

$$\mathfrak{F}(T(x)) = \frac{\mathfrak{F}(\hat{T}(x))}{\mathfrak{F}(h_f(x))} \quad (2.29)$$

Thus,

$$T(x) = \hat{T}(x) * \mathfrak{F}^{-1}\left(\frac{\sqrt{2}}{1-j} e^{j2\pi F^2 u^2}\right) = \hat{T}(x) * \left(\frac{1+j}{2F} e^{-j\frac{\pi}{2}\left[\frac{x}{F}\right]^2}\right) \quad (2.30)$$

Therefore,

$$T(x) = \hat{T}(x) * h_I(x) \quad (2.31)$$

where

$$h_I(x) = \frac{1+j}{2F} e^{-j\frac{\pi}{2}\left[\frac{x}{F}\right]^2} \quad (2.32)$$

The convolution integral defined in Eq. (2.33) is known as the inverse-Fresnel transform; $h_t(x)$ is known as the inverse-Fresnel filter.

The inverse-Fresnel transform,

$$T(x_o) = \frac{1+j}{2F} \int_{-\infty}^{\infty} \hat{T}(x_i) e^{-j\frac{\pi}{2}[\frac{x_o-x_i}{F}]^2} dx_i \quad (2.33)$$

may then be used to reconstruct the desired ring transmittance, $T(x)$, from the diffraction-limited $\hat{T}(x)$.

2.6 Impulse Response

In Chapter 1, it was stated that the inverse-Fresnel transform, as characterized by Eq. (2.33), can not be implemented and employed in practice. For example, the boundaries on the integral extend to infinity, and, this implies the use of infinite number of diffraction-limited data points. In reality, only limited data segments can be processed.

Assuming that a data segment of length W is processed, Eq. (2.33) converts into

$$\tilde{T}(x_o) = \frac{1+j}{2F} \int_{x_o-w/2}^{x_o+w/2} \hat{T}(x_i) e^{-j\frac{\pi}{2}[\frac{x_o-x_i}{F}]^2} dx_i \quad (2.34)$$

To derive the impulse response of the above reconstruction algorithm, Eq. (2.34), it is assumed that the object plane transmittance consists of a very narrow gap in an opaque screen (ring plane). As a consequence, the ring profile can be modeled by Eq. (2.35), which is a delta-Dirac function.

$$T(x_o) = \delta(x_o - x^*) \quad (2.35)$$

The object plane transmittance is then fed into the forward-Fresnel transform in Eq. (2.24) in order to derive the diffraction-limited data, $\hat{T}(x)$, which is observed in the plane of the spacecraft.

Of course, if $\hat{T}(x)$ is fed into the ideal inverse-Fresnel transform of Eq. (2.33), the reconstructed object plane transmittance obtained will be the same as the original object plane transmittance in Eq. (2.35). However, in practice, the approximation to the inverse-Fresnel transform, Eq. (2.34) is used for reconstruction. Consequently, the reconstructed object plane transmittance will be

$$\tilde{T}(x_o) = \frac{W}{2F^2} \text{sinc}\left[\frac{W}{2F^2}(x_o - x^*)\right] e^{j\frac{\pi}{2}\left[\frac{x_o - x^*}{F}\right]^2} \quad (2.36)$$

$\tilde{T}(x_o)$ in the equation above is known as the impulse response. As it is observed, the impulse response is a sinc-function with its peak located at x^* . The Null-to-Null width of the main lobe is

$$\Delta R_w = \frac{4F^2}{W} \quad (2.37)$$

which is a conservative estimate of resolution. A less stringent definition is the 3dB width of the main lobe, or, the so-called equivalent width [2]

$$\Delta R_w = \frac{1}{|\tilde{T}(x^*)|^2} \int_{-\infty}^{+\infty} |\tilde{T}(x_o)|^2 dx_o \quad (2.38)$$

By substituting Eq. (2.36) into Eq. (2.38) and applying Parseval's theorem, it can be proven that the 3dB width is

$$\Delta R_w = \frac{2F^2}{W} \quad (2.39)$$

Eq. (2.39) clearly suggests that more data points or a larger data segment needs to be considered for the reconstruction process in order to achieve finer radial resolution, although, increasing the width of the data segment also increases the computational complexity of the programs utilized for reconstruction.

2.7 Filter Analysis

Codes implemented in MATLAB are used throughout this thesis to model the forward/inverse-Fresnel filters, the forward Fresnel transform, and the reconstruction algorithm (inverse-Fresnel transform). All the data provided to/by these codes are, of course, discrete data. Obviously, the filters are modeled by their sample data points. Because the inverse-Fresnel filter plays the main and vital role of reconstructing the desired object plane transmittance, it is very crucial not to corrupt the filter itself, by sampling it at the wrong frequency and causing aliasing. For this reason, it is important to analyze the frequency response of the filter.

According to the sampling theorem, if a signal $g(x)$ is bandlimited ($G(u) = \mathfrak{F}\{g(x)\} = 0$ for $|u| > U_m$), then $g(x)$ is uniquely determined by its samples $g(nX_s)$, $-\infty \leq n \leq +\infty$, if [3]

$$U_s \geq 2U_m \quad (2.40)$$

Eq. (2.40) is known as the Nyquist condition, and, $U_s = 2U_m$ is called the Nyquist frequency. If $g(x)$ is sampled at a frequency, U_s , less than the Nyquist frequency, then aliasing will be introduced into the signal and the signal will be corrupted.

If

$$g(x) \xrightarrow{\mathfrak{F}} G(u) \quad (2.41)$$

then,

$$g(nX_s) \xrightarrow{\mathfrak{F}} U_s \sum_n G(u - nU_s) \quad (2.42)$$

$$U_s = 1/X_s \quad (2.43)$$

where,

x : Spatial domain variable

u : Frequency domain variable

$\mathfrak{F}$: Fourier transform

X_s : Spatial sampling intervals

U_s : Sampling frequency

The inverse-Fresnel filter is a chirp signal, whose phase is $\theta = -\frac{\pi}{2}(\frac{x}{F})^2$. Thus,

its instantaneous frequency is

$$u = -\frac{1}{2\pi} \frac{d\theta}{dx} = \frac{x}{F^2} \quad (2.44)$$

which means that the frequency increases with x . According to Eq. (2.39), for a desired resolution of ΔR

$$W = \frac{2F^2}{\Delta R} \quad (2.45)$$

This means that the filter extends from $-\frac{F^2}{\Delta R}$ to $\frac{F^2}{\Delta R}$. From Eq. (2.44), the maximum frequency is $\frac{1}{2\Delta R}$. Hence, to avoid aliasing

$$U_s \geq \frac{1}{\Delta R} \quad (2.46)$$

Therefore,

$$X_s \leq \Delta R \quad (2.47)$$

But, Eq. (2.47) is only valid for inverse-Fresnel filter with infinite length and implies that if the unlimited width inverse-Fresnel filter is sampled at ΔR intervals, then, sample

points within $[-\frac{F^2}{\Delta R}, \frac{F^2}{\Delta R}]$ are free of any aliasing. The subsequent paragraphs and

figures will discuss and demonstrate why this criterion, Eq. (2.47), is not sufficient.

Another criterion is put forth for avoiding aliasing.

As previously discussed, the filter can only have finite number of data points.

This can be realized as the theoretical unlimited filter being multiplied by a rectangular window. In the frequency domain this translates into the convolution of the frequency responses of the filter and the window; the frequency response of the rectangular window is a sinc function. Windowing causes the magnitude of the filter's frequency response to oscillate around one, for some bandwidth centered at $u = 0$, and then start to decay. The further away the frequency is from $u = 0$, the less the magnitude of the signal is at that frequency, but never zero. Therefore, as suggested by Eq. (2.42), aliasing is an inevitable

phenomenon that will be introduced into the filter, as soon as the filter is sampled. But, its effect can be negligible, if, a window with lower side lobe levels than the rectangular window is used, and the sampling frequency is very high.

A compromise should be made between the window's side lobe level and the width of the main lobe, which defines the resolution. A Kaiser window is used for this purpose.

Another compromise should also be made between the computation complexity and aliasing. Because, the higher the sampling frequency, the less the aliasing, but the more the number of data points, and thus, the higher the computational complexity.

The magnitude of the frequency response of the inverse-Fresnel filter, windowed by a rectangular window of width $W = \frac{2F^2}{\Delta R_w}$, is displayed in Figures (2.5) - (2.9). The filter has been sampled at sampling intervals of $\frac{\Delta R_w}{100}$. These figures prove that, for negligible aliasing, the inverse-Fresnel filter must be sampled at least every $\frac{\Delta R_w}{2}$. The MATLAB code used for generating these plots can be found at Appendix 2.3 (InverseFresnelFilter.m).

Figure 2.5 Magnitude response of the inverse-Fresnel filter windowed by a rectangular window.

Figure (2.5) shows that the magnitude response of the filter is around -80dB at frequency $u = \frac{U_s}{2}$; which means that, sampling at intervals $X_s = \frac{\Delta R_w}{100}$ causes negligible aliasing.

Frequency response of the inverse-Fresnel filter using a rectangular window and normalized sampling interval $X_s=0.0002$, normalized resolution of 0.02

Figure 2.6 Magnitude response of the inverse-Fresnel filter windowed by a rectangular window.

Figure (2.6) is a zoomed caption of Figure (2.5) and illustrates that the magnitude response of the filter is around -60dB at frequency $u = U_s/20$; which means that,

sampling at intervals $X_s = \frac{\Delta R_w}{10}$ causes negligible aliasing also.

Frequency response of the inverse-Fresnel filter using a rectangular window and normalized sampling interval $X_s=0.0002$, normalized resolution of 0.02

Figure 2.7 Magnitude response of the inverse-Fresnel filter windowed by a rectangular window.

Figure (2.7) shows that the magnitude response of the filter is lower than -40dB at frequency $u = \frac{U_s}{100}$; which means that, sampling at intervals $X_s = \frac{\Delta R_w}{2}$, still causes negligible aliasing.

Figures (2.8) and (2.9) clearly demonstrate that the magnitude response of the filter is approximately at -6 dB at frequency $u = \frac{U_s}{200}$; which means that, sampling at intervals $X_s = \Delta R_w$ causes considerable aliasing .

Figure 2.8 Magnitude of frequency response of the inverse-Fresnel filter windowed by a rectangular window.

Figure 2.9 Magnitude of frequency response of the inverse-Fresnel filter windowed by a rectangular window.

The magnitude of the frequency response of the inverse-Fresnel filter windowed by a Kaiser Window of $\beta = 2.5$ and width $W = \frac{2F^2}{\Delta R_w}$ is displayed in Figures (2.10) - (2.12). The filter has been sampled at sampling intervals of $\frac{\Delta R_w}{100}$ and the MATLAB code used for generating these plots can be found at Appendix 2.3 (InverseFresnelFilter.m).

Comparing Figures (2.10) - (2.12) to their counterparts for a rectangular window in Figures (2.5), (2.7), and (2.9) shows that by using a Kaiser window leakage levels have reduced. However, as it has been mentioned before, this reduction comes at the cost of resolution deterioration.

Figure 2.10 Magnitude of frequency response of the inverse-Fresnel filter windowed by a Kaiser window of $\beta = 2.5$.

Figure (2.10) shows that the magnitude response of the filter is around -90dB at

frequency $u = \frac{U_s}{2}$; which means that, sampling at intervals $X_s = \frac{\Delta R_w}{100}$ causes

negligible aliasing.

Figure 2.11 Magnitude of frequency response of the inverse-Fresnel filter windowed by a Kaiser window of $\beta=2.5$.

Figure (2.11) shows that the magnitude response of the filter is around -55dB at

frequency $u = \frac{U_s}{100}$; which means that, sampling at intervals $X_s = \frac{\Delta R_w}{2}$ causes trivial

aliasing as well.

Figure (2.12) shows that the magnitude response of the filter is approximately at

-15 dB at frequency $u = \frac{U_s}{200}$; which means that, sampling at intervals $X_s = \Delta R_w$

causes considerable aliasing; even though, the aliasing in this case is about 10 dB less than if a rectangular window had been used.

Figure (2.13) shows that the magnitude response of the filter is approximately at -25 dB at frequency $u = U_s / 200$; which means that, sampling at intervals $X_s = \Delta R_w$ causes some aliasing even when a Kaiser window of $\beta = 4$ is used.

From the discussions and supporting figures in this section, it is concluded that, the inverse-Fresnel filter must be sampled at least every $\frac{\Delta R_w}{2}$ interval in order to create negligible aliasing.

Figure 2.12 Magnitude of frequency response of the inverse-Fresnel filter windowed by a Kaiser window of $\beta=2.5$.

Figure 2.13 Magnitude of frequency response of the inverse-Fresnel filter windowed by a Kaiser window of $\beta=4$.

2.8 Implementation of the Forward and Inverse-Fresnel Transforms

As it was earlier stated, the forward and inverse-Fresnel transforms are represented by Eqs. (2.24) and (2.33), which are convolution integrals. The filter slides over the large number of available data points and generates the desired data points as it is being applied to them. Figure (2.14) illustrates how the inverse-Fresnel filter slides over the sample points of the measured transmittance and outputs the sample points of the reconstructed object plane transmittance.

Figure 2.14 Filter moves over the image plane transmittance, filters the data, and outputs the object plane transmittance data points.

The algorithm used for implementing the forward and inverse-Fresnel transforms must be fast. According to Eq. (2.39), the longer the width of the filter is, the better is the resolution. This directly translates into larger number of data points needed for processing. Eqs. (2.24) and (2.33) are however, convolution integrals. If N number of data points are processed at a time, the number of complex multiplications performed by

the convolution algorithm is N^2 . But, if FFT (Fast Fourier Transform) algorithm is employed, the number of complex multiplications required is:

$$\begin{aligned}
 \mathfrak{I}(\hat{T}(x)) &\rightarrow \frac{N}{2} \log_2 N \\
 \mathfrak{I}(h_f(x)) &\rightarrow \frac{N}{2} \log_2 N \\
 \mathfrak{I}(T(x)) &= \mathfrak{I}(\hat{T}(x))\mathfrak{I}(h_f(x)) \rightarrow N \\
 T(x) &= \mathfrak{I}^{-1}(\mathfrak{I}(T(x))) \rightarrow \frac{N}{2} \log_2 N
 \end{aligned} \tag{2.48}$$

Therefore, the total number of complex multiplications required is $N + 3\frac{N}{2} \log_2 N$,

which, is immensely less than N^2 for large number of data points. Table (2.1) compares the computational complexities for both methods and shows the amount of time required to process the data by each algorithm, assuming that, the processor computes 10^7 complex multiplications per second.

As a result of the above discussions and Table (2.1), the FFT method will be used for implementing the forward and inverse-Fresnel transforms.

Table 2.1 Comparison of computational complexities between the convolution and FFT method

N	Convolution #of complex multiplications/time	FFT #of complex multiplications/time
2^8	65,536/ 6.5 ms	3,328/ 0.33 ms
2^{11}	4,194,304/ 419 ms	35,840/ 3.6 ms
2^{20}	2^{40} / 30.5 hrs	32,505,856/ 3.3 sec

SimulateThat.m is the MATLAB implementation of the forward-Fresnel transform and can be found at Appendix 2.4.

Reconstruct.m is the MATLAB implementation of the inverse-Fresnel transform and can be found at Appendix 2.5. The diffraction-limited image plane transmittance is provided to Reconstruct.m, and Reconstruct.m recovers the object plane transmittance which is almost free of diffraction.

2.9 Verifying Reconstruct.m and SimulateThat.m

The question to be answered in this section is, “How reliable is the reconstruction algorithm?” To verify that the reconstruction algorithm has been implemented correctly, a known object plane transmittance is assumed. The corresponding image plane transmittance is then simulated and supplied to the reconstruction algorithm. The reconstructed object plane transmittance is then compared with the expected result.

Exact theoretical image plane transmittances contain Fresnel integrals, which, take a long time for MATLAB to process. Therefore, because the forward and inverse-Fresnel algorithms are alike, the same FFT method used for reconstruction is also employed to generate the image plane transmittance, instead of using the Fresnel integral functions in MATLAB. However, to ensure that the simulated data is reliable, the simulated image plane transmittance should be compared against the theoretical image plane transmittance at least once. Peter Volegov’s MEX file (MATLAB executable), called fcs [5], is used instead of MATLAB’s Fresnel integral functions; the fcs routine is very fast, and therefore, allows for testing of SimulateThat.m. For more information regarding the fcs() function, please refer to Appendix 2.6.

The commands used to compile the fcs MEX file in MATLAB can be found at Appendix 2.7.

2.9.1 Impulse Response

An appropriate ring structure to consider for algorithm verification is that of a very narrow gap, or, an object plane transmittance of the form of a delta function as described by Eq. (2.35). The theoretical image plane transmittance would then be

$$\hat{T}(x_i) = \frac{1-j}{2F} e^{j\frac{\pi}{2}[\frac{x_i-x^*}{F}]^2} \quad (2.49)$$

which is the data provided to the reconstruction algorithm.

The plots below display the reconstructed object plane transmittance versus the theoretical impulse response of Eq. (2.36). The plots show no significant difference between the two. Also, the null-to-null beamwidth is twice the 3dB resolution as expected. The results confirm the accuracy of the MATLAB code Reconstuct.m.

The MATLAB code used for generating the plots below can be found in Appendix 2.8. Please note that all distance parameters have been normalized by the Fresnel scale.

Figure 2.15 Phase of the theoretical versus the reconstructed impulse response, normalized resolution of 0.1. There is a 180° phase jump, at zero-crossings.

Figure 2.16 Magnitude of the theoretical versus the reconstructed impulse response in dB, normalized resolution of 0.1.

Figure 2.17 A closer look at Figure (2.16).

Figure 2.18 A closer look at Figure (2.17). This plot undoubtedly shows that the null-to-null beamwidth is twice the resolution. The normalized resolution is 0.1.

Figures (2.19) - (2.21) show the comparison plots for normalized resolution of 0.2.

Figure 2.19 Phase of the theoretical versus the reconstructed impulse response, normalized resolution of 0.2.

Figure 2.20 Magnitude of the theoretical versus the reconstructed impulse response in dB, normalized resolution of 0.2

Figure 2.21 A closer look at Figure (2.20). This plot indeed shows that the null-to-null beamwidth is twice the resolution. The normalized resolution is 0.2.

Figures (2.22) - (2.23) demonstrate the effect of using a Kaiser window instead of a rectangular window on an impulse response. The Kaiser window worsens the resolution by almost a factor of 2, but, attenuates the side lobes significantly. Larger β values result in poorer resolution but less leakage.

Figure 2.22 Magnitude of the theoretical impulse response in dB for a rectangular window versus the reconstructed impulse response for a Kaiser window of $\beta = 4$.

Figure 2.23 A closer look at Figure (2.22).

2.9.2 Gray Edge Response

An appropriate object plane transmittance to consider for verifying the forward-Fresnel filter implementation is that of a gray edge. The image plane transmittance obtained by the MATLAB function `SimulateThat.m` will be compared with the expected theoretical response. The theoretical response contains Fresnel integrals and computing Fresnel integrals with MATLAB is time-consuming; however, it is essential to do this once for algorithm verification. Therefore, the `fcs()` function described at the beginning of section 2.9 will be used for the purpose of computing the Fresnel integrals. The comparison between the two results in this section will verify the correct implementation of the forward-Fresnel transform.

As mentioned in the above paragraph, the object plane transmittance considered for algorithm verification in this section is a gray edge, which is represented by,

$$T(x_o) = \begin{cases} e^{-\tau/2} & x_o < 0 \\ 1 & x_o > 0 \end{cases} \quad (2.50)$$

Figures (2.24) - (2.27) show the comparison between the two image plane transmittances obtained by `SimulateThat.m` and the theoretical formula. The comparison shows that the diffraction-limited pattern generated by `SimulateThat.m` is very close to the data expected by theoretical derivation. The MATLAB code, `VerifySimulateThat.m`, used to plot these figures can be found at Appendix 2.9.

Figures (2.28) - (2.30) show the object plane, diffraction-limited image plane, and the reconstructed object plane transmittances plotted on top of each other. Figure (2.31)

shows the phase of the reconstructed object plane transmittance. These plots show that the reconstruction algorithm almost eliminates the effect of diffraction. The MATLAB code, `GrayEdgeSimulation.m`, can be found at Appendix 2.10.

Figure 2.24 Comparison of the expected image plane transmittance phase and the phase obtained by `SimulateThat.m`, $\tau = 8$.

Figure 2.25 Comparison of the expected image plane transmittance magnitude and the one obtained by SimulateThat.m, tau = 8.

Figure 2.26 Comparison of the expected image plane transmittance magnitude in dB and the one obtained by SimulateThat.m, tau = 8.

Difference of the image plane transmittance magnitude obtained by the FFT method and theoretical response
Normalized sampling interval $X_s=0.01$, Normalized resolution of 0.1, Beta = 3.5

Figure 2.27 The difference of the image plane transmittance magnitudes obtained by theory and SimulateThat.m, tau = 8. This plot covers a broader distance than the previous plots in order to better demonstrate how negligible the error is.

Figure 2.28 Comparison of the object, image and reconstructed object plane transmittance magnitudes in dB.

Figure 2.29 Comparison of the Object, image and reconstructed object plane transmittance magnitudes.

Figure 2.30 Comparison of the object, image, and reconstructed object plane transmittance phases.

Figure 2.31 Phase of the reconstructed object plane transmittance is almost zero. The expected phase is zero.

Thus far, all figures have been limited in distance axis (horizontal scale) for better visualization of local transitions. However, if all the available data is plotted, it will be seen that inverse-Fresnel filtering the measured data creates an artificial jump in the reconstructed data at the location where the filter reaches the data edge, or, any sudden onset of the object plane transmittance profile. Figure (2.32) shows that forward-Fresnel filtering the object plane transmittance creates a jump at the location where the filter meets the data edge. Figures (2.33) and (2.34) show that the reconstructed data experiences a jump at the meeting point of the inverse-Fresnel filter and the data edge; the height of the jump is almost the same, despite of the method used for creating the diffracted data. The code for creating these three plots, `FilteringArtifact.m`, can be found at Appendix 2.11.

Figure 2.32 The normalized width of the forward-Fresnel filter used is 40. This plot shows that the simulated image plane transmittance has a sudden jump at distance -20, where the filter meets the data edge, compared with the theoretical image plane transmittance, which is continuous.

Figure 2.33 The normalized width of the inverse-Fresnel filter used is 40. This plot shows that the reconstructed profile has a sudden jump at distance -20, where the filter meets the data edge, despite of the fact that the reconstructed data was created from theoretical or simulated diffraction-limited data.

Figure 2.34 A close look at Figure (2.33) shows the height of the jump to be almost the same for both cases.

3 Linearly Tapered Edges

In the previous chapter, the diffracted and reconstructed profiles of a sharp edge were studied. At the location of a sharp ring edge, the transition of the optical depth to zero (transition from ring region to free-space region) is abrupt. This can only be true if the incident rays are normal to the plane of the ring, or if the ring has no thickness at all, none of which, is true in practice. Therefore, the objective of this chapter is to study and investigate the effect of a tapered edge (edge boundary that gradually transitions from ring region to free-space region) on diffracted and reconstructed data, and propose methods for measuring the ring thickness.

3.1 Optical Depth

Plane waves from the Earth reach Saturn's ring plane at an incident angle of B , known as the "ring opening" angle.

Figure 3.1 Ray incident on ring plane at an angle B

The optical depth, τ , is a measurement of the signal attenuation by ring material. This attenuation is also known as the transmittance of the ring. Assuming that τ_n

represents optical depth when the incident rays are normal to the surface of Saturn's ring plane, the relationship between τ and τ_n is

$$\frac{1}{\mu_0} = \frac{1}{\sin B} = \frac{\tau}{\tau_n} \quad (3.1)$$

When the incident rays reach the edge of a ring structure, the optical depth gradually decreases to zero; these ring structures are known to have “tapered edges” not “sharp edges.” By studying the reconstructed profile of these tapered edges, it is possible to measure the thickness of the rings.

3.2 Linear Approximation of Optical Depth at Ring Edges

As discussed above, it is possible to measure the thickness of rings by studying the reconstructed ring transmittance profile. To study this possibility, it is important to characterize the behavior of the optical depth at an edge of a ring structure.

Figure 3.2 Illustration of optical depth at ring edges

d : Ring thickness

L : Projected ring thickness

X_o : Location at which τ starts decaying from full value to zero

X_e : Edge of the ring

Assuming that ring material and density do not change over the distance L ,

$$\frac{\tau(X_o + dx)}{\tau(X_o)} = \frac{X_e - dx - X_o}{X_e - X_o} = \frac{L - dx}{L} \quad (3.2)$$

where

$$L = d \cot B \quad (3.3)$$

and

$$\tau(x) = \tau \quad \text{for } x \leq X_o \quad (3.4)$$

Therefore,

$$\tau(X_o + dx) = \tau \frac{(L - dx)}{L} \quad (3.5)$$

Using Taylor's first order approximation at $x = X_o$

$$\tau(X_o + dx) = \tau(X_o) + \left. \frac{\partial \tau(X)}{\partial x} \right|_{x=X_o} dx \quad (3.6)$$

By combining Equations (3.5) and (3.6), Eq. (3.7) which is a linear approximation of the optical depth in the region nearing an edge is derived.

$$\tau(x) = -\frac{\tau}{L}(x - X_o) + \tau \quad (3.7)$$

Therefore the optical depth $\tau(x)$ is:

$$\tau(x) = \begin{cases} \tau & x \leq X_o \\ -\frac{\tau}{L}(x - X_o) + \tau & X_o < x \leq X_e \\ 0 & X_e < x \end{cases} \quad (3.8)$$

By using Eq. (3.8), it is possible to input the profile of a linearly tapered edge as the object plane transmittance to the forward-Fresnel transform algorithm, create the diffracted profile which represents measured data by Cassini, reconstruct the object plane transmittance, and study the characteristics of the reconstructed profile in order to determine how to calculate the thickness of the rings by observing the reconstructed profiles.

Figures (3.3) and (3.4) demonstrate the difference between a sharp edge and a linearly tapered edge profile. The program used for generating these plots, LinearlyTaperedEdgeSimulation.m, can be found at Appendix 3.1.

These are the parameters used in both cases:

Desired normalized resolution $dR_n : 0.05$

Normalized sampling intervals $dx_n : 0.005$

$\tau : 8$

Figure 3.3 Profile of a sharp edge

Figure 3.4 Profile of a linearly tapered edge with normalized $L = 1$.

3.3 Effect of Ring Thickness on Measured and Reconstructed Profiles

It was shown in section 3.2 that optical depth varies in accordance with Eq. (3.8) at the vicinity of a ring edge. The magnitude profiles of the object transmittance for an abrupt edge, and a linearly tapered edge, are shown in Figures (3.3) and (3.4), respectively. Figure (3.4) shows a smooth transition from the ring area to the free-space region compared with the abrupt edge profile. The larger the thickness, the smoother the transition will be, and the smoother the transition, the less the diffraction in the measured transmittance. Therefore, the abruptness of the transition from ring area to free-space region can be determined by the amount of diffraction-like oscillations in the measured profile.

Figures (3.5), (3.6), and (3.8) show how the measured image plane transmittance becomes less diffracted as ring thickness increases. Figure (3.5) also illustrates that the profile of the measured data becomes closer to the reconstructed data as the thickness increases. Furthermore, the peak to peak level of the diffraction like oscillations decreases as the rings becomes thicker, and, as Figure (3.10) shows, the phase also becomes closer to zero with increasing thickness. Figure (3.12) zooms into Figure (3.6), and, demonstrates that the overshoots caused by Gibb's phenomenon decrease as the thickness of the rings increases.

Figures (3.7), (3.9), and (3.11) show the reconstructed object plane transmittance magnitudes in dB and normal scale, and the phase in degrees, respectively.

Figure 3.5 Measured image plane transmittance for linearly tapered edges with normalized $L = 0, 1, 2, 3$. Each individual plot has been shifted vertically for better comparison.

Figure 3.6 Measured image plane transmittance magnitudes in dB for linearly tapered edges of $L = 0, 1, 2, 3$. Profiles have been plotted on top of each other for better comparison.

Figure 3.7 Reconstructed object plane transmittance magnitudes in dB for linearly tapered edges of $L = 0, 1, 2, 3$. Profiles have been plotted on top of each other for better comparison.

Figure 3.8 Measured image plane transmittance magnitudes for linearly tapered edges of $L = 0, 1, 2, 3$. Profiles have been plotted on top of each other for better comparison.

Figure 3.9 Reconstructed object plane transmittance magnitudes in dB for linearly tapered edges of $L = 0, 1, 2, 3$. Profiles have been plotted on top of each other for better comparison.

Figure 3.10 Phase of the image plane transmittance for linearly tapered edges of $L = 0, 1, 2, 3$. Profiles have been plotted on top of each other for better comparison.

Figure 3.11 Reconstructed phase of the object plane transmittance for linearly tapered edges of $L = 0, 1, 2, 3$. Profiles have been plotted on top of each other for better comparison.

Figure 3.12 An insert of Figure (3.6)

Figures (3.5) – (3.12) in this section have been generated using the MATLAB function `LinearlyTaperedEdgeSimulationComparison.m`, which can be found at Appendix 3.2.

From the above discussions and supporting plots, it can be concluded that the thickness of the rings should be measurable by studying the characteristics of the measured and reconstructed data.

3.4 Transition Width of a Reconstructed Abrupt Edge Profile

In the reconstructed profile of an abrupt ring edge, the transition from the ring region to free-space region is no longer sharp and takes place over some distance. This is shown in Figure (3.13) below. The reason for this non-zero transition width is the limited resolution caused by the use of a finite data segment in the reconstruction process.

Figure 3.13 Comparison of the reconstructed profile of an abrupt edge versus the original edge profile. The reconstructed profile does not make a sudden sharp transition from the ring region to the free space region. In this case the transition takes place over a normalized distance of 0.2.

Figure (3.14) shows a closer view of Figure (3.13). It is seen that the transition width from point A to point B shown is $0.2F$, which is twice the radial resolution of $0.1F$.

Figure 3.14 A closer caption of Figure (3.13). The normalized transition distance from point A (ring region) to point B (free-space) is approximately 0.2, which is twice the radial resolution. The actual object plane transmittance in this plot does not seem sharp because the profile has been generated with normalized sampling intervals of 0.01, and, this gap between the last sample in the ring region and the first sample in free-space is visible in this plot.

Points A and B in Figure (3.14) are locations at which, the reconstructed profile reaches the ring and free-space regions, respectively.

Table (3.1) displays the relationship between the transition width and the resolution. The data in this table has been accumulated based on same technique of plotting the reconstructed profile of a sharp edge for different radial resolutions and measuring the transition width on the plot. Table (3.1) shows that the transition width is twice the resolution. This is because the distance between points A and B corresponds

with the null-to-null width of the sinc function impulse response in Eq. (2.36), and does not represent the 3-dB resolution.

Table 3.1 Normalized transition width of the reconstructed profile of a sharp edge for $\beta = 3.5$ versus normalized resolution.

Normalized Resolution	Normalized transition width
0.06	0.12
0.1	0.2
0.2	0.4
0.3	0.6
0.4	0.8

3.5 Measuring the Ring Thickness

As discussed in section 3.4, the reconstructed profile of a sharp edge is no longer abrupt, and the transition from the ring region to the free-space region takes place over a transition width twice the radial resolution. This outcome could have been predicted; when the reconstructed profile of an impulse becomes a main-lobe with width twice the radial resolution, accompanied by side lobes, it is also expected that an abrupt edge be lost and to expand over a width twice the resolution. In fact, ring variations over a length smaller than the radial resolution (3-dB width of the main-lobe) are lost. As a consequence, the projected ring thickness, L , can only be measured if it is equal or greater than the radial resolution.

In this section, two methods are proposed for calculating the projected ring thickness L . The first method, direct width measurement, is accurate if $L \geq 5 \times dR_n$,

where dR_n is the normalized radial resolution. The second method, slope measurement, concludes in more accurate results for $L \leq 5 \times dR_n$.

3.5.1 Direct Width Measurement

One method of measuring the projected ring thickness is to study the effect of the thickness on the transition width. For this purpose, many ring profiles with different projected ring thicknesses and resolutions were reconstructed and studied. The result of studying these plots showed that, if the projected ring thickness is five or more times larger than the radial resolution, then the width over which the transmittance increases from $e^{-\pi/2}$ to 1 (from ring region to free-space) is the expected projected ring thickness plus the radial resolution. In other words, from the reconstructed magnitude profile (in dBs), the transition width is measured and then the radial resolution is subtracted from it. If the result is five or more times the radial resolution, then the derived answer is acceptable, and is equal to the projected ring thickness L . Below are some examples.

W_n : Normalized transition width measured on the graph

L_n : Normalized projected ring thickness

dR_n : Normalized radial resolution

Magnitude of the reconstructed object plane transmittance in dB. Normalized sampling interval $X_s=0.001$.
 Normalized resolution of 0.1, Beta = 3.5, normalized taper length = 0.5

Figure 3.15 In this figure $W_n = 0.625$, $dR_n = 0.1$, therefore $L_n = W_n - dR_n = 0.525$. The expected L_n is 0.5.

Magnitude of the reconstructed object plane transmittance in dB. Normalized sampling interval $X_s=0.001$.
Normalized resolution of 0.1, Beta = 3.5, normalized taper length = 0.5

Figure 3.16 A closer look at this plot is shown in Figure (3.17) for more accurate measurements.

Magnitude of the reconstructed object plane transmittance in dB. Normalized sampling interval $X_s=0.001$.
Normalized resolution of 0.1, Beta = 3.5, normalized taper length = 1

Figure 3.17 In this figure $W_n = 1.1$, $dR_n = 0.1$, therefore $L_n = W_n - dR_n = 1$. The expected L_n is also 1.

Magnitude of the reconstructed object plane transmittance in dB. Normalized sampling interval $X_s=0.001$.
Normalized resolution of 0.04, Beta = 3.5, normalized taper length = 0.2

Figure 3.18 A closer look at this plot is shown in Figure (3.19) for more accurate measurements.

Magnitude of the reconstructed object plane transmittance in dB. Normalized sampling interval $X_s=0.001$.
Normalized resolution of 0.04, Beta = 3.5, normalized taper length = 0.2

Figure 3.19 In this figure $W_n = 0.25$, $dR_n = 0.04$, therefore $L_n = W_n - dR_n = 0.21$. The expected L_n is 0.2.

If by using the above method, the derived L_n is less than five times the radial resolution, then the calculated result is not reliable. To measure the thickness in these cases, the slope method explained in the next section should be employed.

3.5.2 Slope Method

As previously discussed, if the projected ring thickness derived by the “direct width measurement” method is less than five times the radial resolution, more accurate results will be obtained if the slope method is used instead. In this section, the derivative of the transmittance of a linearly tapered edge will be derived. The slope method is based on these results.

$$T(x) = e^{-\tau(x)/2} \quad (3.9)$$

$$\frac{d(T(x))}{dx} = -\frac{1}{2} \frac{d(\tau(x))}{dx} e^{-\tau(x)/2} = -\frac{1}{2} \frac{d(\tau(x))}{dx} T(x) \quad (3.10)$$

According to Eq. (3.8), in the linearly tapered region,

$$\frac{d(T(x))}{dx} = \frac{\tau}{2L} T(x) \quad (3.11)$$

Therefore, if the slope of the plot is measured at some point x , then L could be calculated by,

$$L = \frac{\tau}{2\left(\frac{dT(x)}{dx}\right)} T(x) \quad (3.12)$$

Figures (3.20) and (3.21) illustrate how Eq. (3.12) can be utilized to calculate the projected ring thickness.

Figure 3.20 This is the profile of a linearly tapered edge with a normalized projected ring thickness of 0.5. The constant level in this figure is 0.01832, it can be calculated from Eq. (3.9) that $\tau = 8$.

A “zoomed” version of this figure is shown in Figure (3.21) for more accurate measurements.

Figure 3.21 According to Eq. (3.12), $L_n = \frac{8 \times 0.01832}{(0.02 - 0.01832)} \times \frac{1}{2} = 0.48$, which is very close to 0.5.

As it is shown in Figure (3.22), the slope of the reconstructed profile is not the same as the slope of the actual expected ring transmittance profile at every point. However, from the studies carried out on various plots, it seems that the slope is approximately the same for both cases at a point distant by the radial resolution from the edge of where the linear taper in the ring region starts.

Figure 3.22 This figure illustrates how the slopes of the reconstructed profile and the actual profile are almost the same in some regions and differ in other regions.

Figures (3.23) and (3.24) show how the slope method can be used to derive a more accurate estimate of the normalized projected ring thickness.

Magnitude of the reconstructed object plane transmittance in dB. Normalized sampling interval $X_s=0.001$.
Normalized resolution of 0.04, Beta = 3.5, normalized taper length = 0.1

Figure 3.23 In this figure $W_n = 0.158$, $dR_n = 0.04$, therefore $L_n = W_n - dR_n = 0.118$. The expected L_n is 0.1. 0.118 is smaller than 5×0.04 , therefore the slope method will be used to derive L_n . The slope should be measured around point $0.13 - 0.04 = 0.09$. Figure (3.24) is a closer look at this figure at the vicinity of 0.09.

Magnitude of the reconstructed object plane transmittance. Normalized sampling interval $X_s=0.001$.
Normalized resolution of 0.04, Beta = 3.5, normalized taper length = 0.1

Figure 3.24 Using Eq. (3.12), L_n will be derived to be 0.11.

Figures (3.25) and (3.26) show another example in which the slope method is used to calculate the normalized projected ring thickness.

Figure 3.25 This figure predicts L_n to be $1.1 - 0.04 = 0.07$ which is less than 5×0.04 . Therefore, the slope method will be used for more accurate result. The slope should be measured around point $0.08 - 0.04 = 0.04$. Figure (3.26) is a closer look at this figure around point 0.04.

Figure 3.26 According to Eq. (3.12), $L_n = \frac{8 \times 0.092}{0.011} \times \frac{1}{2} = 0.0502$, which is indeed very close to 0.05.

4 Occultation Geometry and Calculation of Fresnel Scale

This chapter introduces the actual occultation geometry. A one-dimensional approximation of the geometry together with the Fresnel-Huygens integral is used to derive an expression for the Fresnel scale, using the stationary phase approximation. The expression demonstrates the dependency of the Fresnel scale on various occultation geometry parameters.

4.1 Geometry

Figure (4.1) shows the radio occultation geometry in a Saturnocentric coordinate system. The z axis points from the center of Saturn to its north pole. The unit vector $\hat{u}_i$ is directed along the line of sight from Earth to Saturn at the time of occultation. As explained in previous chapters, the wave is assumed incident from Earth, which is the opposite of the actual experimental configuration. This assumed configuration is easier to conceive and analyze. The xy plane coincides with the mean plane of the rings and is, of course, normal to the north pole of Saturn. The azimuthal angles are measured with respect to the x axis, where

$$\hat{u}_y = \frac{\hat{u}_z \times \hat{u}_i}{|\hat{u}_z \times \hat{u}_i|} \quad (4.1)$$

$$\hat{u}_x = \hat{u}_y \times \hat{u}_z \quad (4.2)$$

and $\hat{u}_x, \hat{u}_y, \hat{u}_z$ are unit vectors along the axis of a rectangular coordinate system; $\hat{u}_z$ is the direction of Saturn's north pole, and, $\hat{u}_i$ is a unit vector from Earth to the spacecraft.

Figure 4.1 Radio occultation geometry in a Saturnocentric coordinate system [1].

At the time of occultation, the line of sight intersects with the ring plane at $(x_o, y_o, 0)$, or, $(\rho_o, \phi_o, 0)$ in a cylindrical coordinate system. Angle B is known as the ring opening angle, and $\vec{R}_c$ is the spacecraft position vector.

4.2 One-Dimensional Approximation of the Measured Transmittance

Starting from the Fresnel-Huygens integral in Eq. (2.1),

$$E_c \approx E_0 \iint_{\text{RingSurface}} \frac{\mu_0 \vec{T}(\vec{\rho}) e^{jk\vec{u}_i \cdot \vec{\rho}}}{j\lambda} \frac{e^{jk|\vec{R}_c - \vec{\rho}|}}{|\vec{R}_c - \vec{\rho}|} d\vec{\rho} \quad (4.3)$$

Where [1]

E_c : Average complex field observed at $\bar{R}_c$

E_0 : Electric field at the ring plane, in the absence of the rings (normalization constant)

μ_0 : Obliquity factor, which, because the ring opening angle B hardly changes during the observation period, is assumed to be constant

$$\mu_0 = \sin(B) \quad (4.4)$$

λ : Wave length

k : Wave number ($2\pi/\lambda$)

$\bar{T}(\bar{\rho})e^{jk\hat{u}_i \cdot \bar{\rho}}$: Represents the complex amplitude of a secondary source of infinitesimal area $d\bar{\rho}$ located at radial position $\bar{\rho}$

$\bar{T}(\bar{\rho})$: Complex transmittance of ring at radial position $\bar{\rho}$

Therefore, the measured transmittance at $\bar{R}_c$ can be written as

$$\hat{T} = \left(\frac{E_c}{E_0}\right)e^{-jk\hat{u}_i \cdot \bar{R}_c} \quad (4.5)$$

Eqs. (4.3) and (4.5) combine to yield

$$\hat{T}(\rho_0) = \iint_{RingSurface} \frac{\mu_0}{j\lambda} \bar{T}(\bar{\rho}) e^{jk\hat{u}_i \cdot (\bar{\rho} - \bar{R}_c)} \frac{e^{jk|\bar{R}_c - \bar{\rho}|}}{|\bar{R}_c - \bar{\rho}|} d\bar{\rho} \quad (4.6)$$

Assuming near local circular symmetry of ring transmittance, $\bar{T}(\bar{\rho})$ reduces to a one-dimensional function which is only dependent on the magnitude ρ . Consequently, Eq. (4.6) transforms into the following one-dimensional equation.

$$\hat{T}(\rho_0) = \frac{\mu_0}{j\lambda} \int_0^\infty d\rho \rho T(\rho) \int_0^{2\pi} d\varphi \frac{e^{jk|\vec{R}_c - \vec{\rho}|}}{|\vec{R}_c - \vec{\rho}|} e^{jk\hat{u}_i \cdot (\vec{\rho} - \vec{R}_c)} \quad (4.7)$$

If

$$\Phi(\rho_0, \varphi_0; \rho, \varphi) = k(\hat{u}_i \cdot (\vec{\rho} - \vec{R}_c) + |\vec{R}_c - \vec{\rho}|) \quad (4.8)$$

then

$$\hat{T}(\rho_0) = \frac{\mu_0}{j\lambda} \int_0^\infty d\rho \rho T(\rho) \int_0^{2\pi} d\varphi \frac{e^{j\Phi(\rho_0, \varphi_0; \rho, \varphi)}}{|\vec{R}_c - \vec{\rho}|} \quad (4.9)$$

Derivations below are an attempt to simplify Eq. (4.8), in order to solve Eq. (4.9).

Eqs. (4.10) and (4.11) map the cylindrical coordinate system to the Cartesian (rectangular) coordinate system.

$$\hat{\rho} = \cos \varphi \hat{x} + \sin \varphi \hat{y} \quad (4.10)$$

$$\hat{\varphi} = -\sin \varphi \hat{x} + \cos \varphi \hat{y} \quad (4.11)$$

Because of Eq. (4.1), the line of sight has no projection on the y axis. Therefore,

$$\hat{u}_i = \cos B \hat{x} - \sin B \hat{z} \quad (4.12)$$

$$\hat{u}_i \cdot (\vec{\rho} - \vec{R}_c) = \hat{u}_i \cdot (\vec{\rho} - \vec{\rho}_0) + \hat{u}_i \cdot (\vec{\rho}_0 - \vec{R}_c) \quad (4.13)$$

$$\hat{u}_i \cdot (\vec{\rho} - \vec{\rho}_0) = \hat{u}_i \cdot \vec{\rho} - \hat{u}_i \cdot \vec{\rho}_0 \quad (4.14)$$

Combining Eqs. (4.10), (4.12), and (4.14) results into

$$\hat{u}_i \cdot (\vec{\rho} - \vec{\rho}_0) = -\cos B(\rho_0 \cos \varphi_0 - \rho \cos \varphi) \quad (4.15)$$

It is seen in Figure (4.1) that $\vec{\rho}_0 - \vec{R}_c = -\vec{D}$, which is a vector along the line of sight, thus,

$$\hat{u}_i \cdot (\vec{\rho}_0 - \vec{R}_c) = -|\vec{\rho}_0 - \vec{R}_c| \quad (4.16)$$

By combining Eqs. (4.13), (4.15) and (4.16)

$$\hat{u}_i \cdot (\vec{\rho} - \vec{R}_c) = -\cos B(\rho_0 \cos \varphi_0 - \rho \cos \varphi) - |\vec{R}_c - \vec{\rho}_0| \quad (4.17)$$

In general,

$$|\vec{A} - \vec{B}|^2 = |\vec{A}|^2 + |\vec{B}|^2 - 2\vec{A} \cdot \vec{B} \quad (4.18)$$

Therefore,

$$|\vec{R}_c - \vec{\rho}| = |\vec{R}_c - \vec{\rho}_0 - (\vec{\rho} - \vec{\rho}_0)| = \sqrt{|\vec{R}_c - \vec{\rho}_0|^2 + |\vec{\rho} - \vec{\rho}_0|^2 - 2(\vec{R}_c - \vec{\rho}_0) \cdot (\vec{\rho} - \vec{\rho}_0)} \quad (4.19)$$

$$(\vec{R}_c - \vec{\rho}_0) \cdot (\vec{\rho} - \vec{\rho}_0) = |\vec{R}_c - \vec{\rho}_0| \hat{u}_i \cdot (\vec{\rho} - \vec{\rho}_0) \quad (4.20)$$

Using Eq. (4.15)

$$(\vec{R}_c - \vec{\rho}_0) \cdot (\vec{\rho} - \vec{\rho}_0) = -|\vec{R}_c - \vec{\rho}_0| \cos B(\rho_0 \cos \varphi_0 - \rho \cos \varphi) \quad (4.21)$$

If

$$D = |\vec{R}_c - \vec{\rho}_0| \quad (4.22)$$

$$\xi = \frac{\cos B(\rho_0 \cos \varphi_0 - \rho \cos \varphi)}{D} \quad (4.23)$$

$$\eta = \frac{[\rho_0^2 + \rho^2 - 2\rho_0\rho \cos(\varphi - \varphi_0)]}{D^2} \quad (4.24)$$

then

$$\begin{aligned} |\vec{R}_c - \vec{\rho}| &= \sqrt{D^2 + \rho_0^2 + \rho^2 - 2\rho_0\rho \cos(\varphi - \varphi_0) + 2D \cos B(\rho_0 \cos \varphi_0 - \rho \cos \varphi)} \\ &= D\sqrt{1 + \eta + 2\xi} \end{aligned} \quad (4.25)$$

Thus, from Eqs. (4.8), (4.17), (4.22), (4.23), and (4.25), Eq. (4.26) below is obtained.

$$\Phi(\rho_0, \varphi_0; \rho, \varphi) = kD[\sqrt{1 + \eta + 2\xi} - (\xi + 1)] \quad (4.26)$$

Eq. (4.9) can be simplified further, using the stationary phase approximation, as discussed in the following section.

4.2.1 Stationary Phase Approximation

According to [6], the stationary phase approximation is particularly useful for providing a close estimate of oscillating integrals, especially of superchirp functions, $e^{jf(x)}$, for which, no closed form of the spectrum can be derived.

If [6]

$$I(k) = \int_{-\infty}^{+\infty} r(x) e^{jk \cdot f(x)} dx \quad (4.27)$$

where $r(x)$ and $f(x)$ are real-valued functions, and, k is a real-valued constant, at regions where the rate of oscillation of the exponential term is more rapid with respect to oscillations in $r(x)$, the area underneath the neighboring positive and negative lobes of $r(x)e^{jk \cdot f(x)}$ will approximately cancel. In those regions of the domain where $k \cdot f(x)$ is small, and hence the exponential term behaves like a unit constant, the area can be determined by the width of this region of the domain and the amplitude of the real-valued function $r(x)$. Therefore, the integral in Eq. (4.27) can be estimated by evaluating the integrand only in those regions where the exponential term oscillates slowly, or, wherever

the derivative of the phase function is approximately 0, $\frac{df}{dx} \cong 0$; these points at which

$\frac{df}{dx} \cong 0$ are called the stationary phase points of $f(x)$ [6].

To solve the integral in Eq. (4.9), using the stationary phase approximation, the solution to the equation below must be calculated.

$$\frac{\partial \Phi}{\partial \varphi} = 0 \quad (4.28)$$

The solution to Eq. (4.28) is computed numerically, using an iterative method.

From an assumed known approximation, $\varphi_{s,n}$, a better approximation $\varphi_{s,n+1}$ is obtained

[1]. The Newton-Raphson iteration method is used for this purpose.

If there exists a number x , where $f(x) = 0$, and the function f and its first derivative are continuous in the vicinity of x , then x_{k+1} will eventually converge to x , if the initial guess x_0 is within an acceptable range from x . Eq. (4.29) below is known as the Newton-Raphson iteration formula.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad (4.29)$$

Therefore, $\varphi_{s,n+1}$ is the solution to Eq. (4.28) and can be found iteratively using

$$\varphi_{s,n+1} = \varphi_{s,n} - \frac{\Phi'(\varphi_{s,n})}{\Phi''(\varphi_{s,n})} = \varphi_{s,n} - \frac{\Phi'_n}{\Phi''_n} \quad (4.30)$$

The relationship between Φ'_n and Φ''_n can be found by using the second order Taylor series expansion of function Φ about the stationary phase point $\varphi_{s,n}$.

$$\Phi \approx \Phi_n + \Phi'_n(\varphi - \varphi_{s,n}) + 0.5\Phi''_n(\varphi - \varphi_{s,n})^2 \quad (4.31)$$

If, $\varphi = \varphi_{s,n+1}$ in Eq. (4.31), then

$$\Phi(\varphi_{s,n+1}) = \Phi_{n+1} \approx \Phi_n + \Phi'_n(\varphi_{s,n+1} - \varphi_{s,n}) + 0.5\Phi''_n(\varphi_{s,n+1} - \varphi_{s,n})^2 \quad (4.32)$$

Using Eq. (4.30) to substitute $\varphi_{s,n+1} - \varphi_{s,n}$ with $-\frac{\Phi'_n}{\Phi''_n}$ in Eq. (4.32), results in

$$\Phi_{n+1} = \Phi_n - \frac{1}{2} \left(\frac{\Phi'_n}{\Phi''_n} \right)^2 \quad (4.33)$$

The iteration starts with an initial guess, the next iteration is calculated from Eq. (4.30), the corresponding Φ is calculated by Eq. (4.33), or, directly from Eq. (4.26) for more accurate results.

4.2.2 Quadratic Approximation of Φ

The quadratic approximation follows from the iteration $n = 1$ in Eqs. (4.30) and (4.33). The initial guess is

$$\varphi_{s,0} = \varphi_0 \quad (4.34)$$

which is the phase at the point of intercept of the line of sight with the planetary ring at the time of occultation.

Substituting this initial guess in Eq. (4.26) gives

$$\Phi_0 = kD[\sqrt{1 + \eta_0 + 2\xi_0} - (\xi_0 + 1)] \quad (4.35)$$

where

$$\xi_0 = -\cos B \cos \varphi_0 \left(\frac{\rho - \rho_0}{D} \right) \quad (4.36)$$

$$\eta_0 = \left(\frac{\rho - \rho_0}{D}\right)^2 \quad (4.37)$$

From Eq. (4.26)

$$\frac{\partial \Phi}{\partial \varphi} = kD \left[\left(\frac{\partial \xi}{\partial \varphi} + \frac{1}{2} \frac{\partial \eta}{\partial \varphi} \right) (1 + 2\xi + \eta)^{-\frac{1}{2}} - \frac{\partial \xi}{\partial \varphi} \right] \quad (4.38)$$

in which,

$$\frac{\partial \xi}{\partial \varphi} = \cos B \sin \varphi \frac{\rho}{D} \quad (4.39)$$

$$\frac{\partial \eta}{\partial \varphi} = 2 \sin(\varphi - \varphi_0) \frac{\rho \rho_0}{D^2} \quad (4.40)$$

Therefore,

$$\frac{\partial \Phi}{\partial \varphi} = kD \left[\left(\cos B \sin \varphi \frac{\rho}{D} + \sin(\varphi - \varphi_0) \frac{\rho \rho_0}{D^2} \right) (1 + 2\xi + \eta)^{-\frac{1}{2}} - \cos B \sin \varphi \frac{\rho}{D} \right] \quad (4.41)$$

Consequently,

$$\begin{aligned} \frac{\partial^2 \Phi}{\partial \varphi^2} = kD & \left[\left(\cos B \cos \varphi \frac{\rho}{D} + \cos(\varphi - \varphi_0) \frac{\rho \rho_0}{D^2} \right) (1 + 2\xi + \eta)^{-\frac{1}{2}} - \right. \\ & \left. \left(\cos B \sin \varphi \frac{\rho}{D} + \sin(\varphi - \varphi_0) \frac{\rho \rho_0}{D^2} \right)^2 (1 + 2\xi + \eta)^{-\frac{3}{2}} - \cos B \cos \varphi \frac{\rho}{D} \right] \end{aligned} \quad (4.42)$$

Thus,

$$\Phi'_0 = kD \left[\left(\cos B \sin \varphi_0 \frac{\rho}{D} \right) (1 + 2\xi_0 + \eta_0)^{-\frac{1}{2}} - \cos B \sin \varphi_0 \frac{\rho}{D} \right] \quad (4.43)$$

and,

$$\begin{aligned} \Phi''_0 = kD & \left[\left(\cos B \cos \varphi_0 \frac{\rho}{D} + \frac{\rho \rho_0}{D^2} \right) (1 + 2\xi_0 + \eta_0)^{-\frac{1}{2}} - \left(\cos B \sin \varphi_0 \frac{\rho}{D} \right)^2 (1 + 2\xi_0 + \eta_0)^{-\frac{3}{2}} \right. \\ & \left. - \cos B \cos \varphi_0 \frac{\rho}{D} \right] \end{aligned} \quad (4.44)$$

Assuming that

$$X = \frac{\rho - \rho_0}{D} \quad (4.45)$$

and,

$$X \ll 1 \quad (4.46)$$

Eqs. (4.35), (4.43), and (4.44) may be further simplified.

From Taylor series expansion, if $u \ll 1$, then

$$(1+u)^{1/2} \Big|_{u=0} \approx 1 + \frac{1}{2}u - \frac{1}{8}u^2 + O(u^3) \quad (4.47)$$

$$(1+u)^{-1/2} \Big|_{u=0} \approx 1 - \frac{1}{2}u + \frac{3}{8}u^2 + O(u^3) \quad (4.48)$$

$$(1+u)^{-3/2} \Big|_{u=0} \approx 1 - \frac{3}{2}u + \frac{15}{8}u^2 + O(u^3) \quad (4.49)$$

Thus,

$$\sqrt{1 + \eta_0 + 2\xi_0} = 1 + \frac{1}{2}(2\xi_0 + \eta_0) - \frac{1}{8}(2\xi_0 + \eta_0)^2 \quad (4.50)$$

Substituting Eqs. (4.36), (4.37), (4.45), and (4.50) into Eq. (4.35) results into

$$\Phi_0 = kD \left[\frac{1}{2}(-2 \cos B \cos \varphi_0 X + X^2) - \frac{1}{8}(-2 \cos B \cos \varphi_0 X + X^2)^2 + \cos B \cos \varphi_0 X \right] \quad (4.51)$$

With the assumption in Eq. (4.46), $O(X^3)$ and higher order terms can be assumed

negligible. Therefore,

$$\Phi_0 = \frac{kD}{2}(1 - \cos^2 B \cos^2 \varphi_0)X^2 \quad (4.52)$$

Using Eqs. (4.36), (4.37), (4.45), (4.48), and also neglecting $O(X^3)$ and higher order terms converts Eq. (4.43) to

$$\Phi'_0 = kD[(\cos B \sin \varphi_0 \frac{\rho}{D})(\cos B \cos \varphi_0 X - \frac{1}{2}X^2 + \frac{3}{2}\cos^2 B \cos^2 \varphi_0 X^2)] \quad (4.53)$$

$$\frac{\rho}{D} = \frac{\rho - \rho_0}{D} + \frac{\rho_0}{D} = X + \frac{\rho_0}{D} \quad (4.54)$$

Substituting Eq. (4.54) in Eq. (4.53) gives

$$\begin{aligned} \Phi'_0 = kD[& \cos^2 B \sin \varphi_0 \cos \varphi_0 \frac{\rho_0}{D} X + (\cos^2 B \sin \varphi_0 \cos \varphi_0 - \frac{1}{2}\cos B \sin \varphi_0 \frac{\rho_0}{D} + \\ & \frac{3}{2}\cos^3 B \cos^2 \varphi_0 \sin \varphi_0 \frac{\rho_0}{D})X^2 + (-\frac{1}{2}\cos B \sin \varphi_0 + \frac{3}{2}\cos^3 B \cos^2 \varphi_0 \sin \varphi_0)X^3] \end{aligned} \quad (4.55)$$

Using Eqs. (4.45), (4.48), (4.49), and (4.54) in Eq. (4.44), and also neglecting $O(X^3)$ and higher order terms gives

$$\begin{aligned} \Phi''_0 = kD[& (\frac{\rho_0}{D})^2 (1 - \cos^2 B \sin^2 \varphi_0) + (\frac{\rho_0}{D} + (\frac{\rho_0}{D})^2 \cos B \cos \varphi_0 + \cos^2 B \frac{\rho_0}{D} (\cos^2 \varphi_0 - \\ & 2 \sin^2 \varphi_0) - 3(\frac{\rho_0}{D})^2 \cos^3 B \cos \varphi_0 \sin^2 \varphi_0)X + (-\frac{1}{2}(\frac{\rho_0}{D})^2 + \frac{1}{2}\frac{\rho_0}{D} \cos B \cos \varphi_0 + \\ & \cos^2 B \cos^2 \varphi_0 (\frac{3}{2}(\frac{\rho_0}{D})^2 + 1) + \cos^2 B \sin^2 \varphi_0 (\frac{3}{2}(\frac{\rho_0}{D})^2 - 1) + \frac{3}{2}\frac{\rho_0}{D} \cos^3 B \cos^3 \varphi_0 - \\ & 6\frac{\rho_0}{D} \cos^3 B \sin^2 \varphi_0 \cos \varphi_0 - \frac{15}{2}(\frac{\rho_0}{D})^2 \cos^4 B \sin^2 \varphi_0 \cos^2 \varphi_0)X^2] \end{aligned} \quad (4.56)$$

From Eq. (4.30)

$$\varphi_1 = \varphi_0 - \frac{\Phi'_0}{\Phi''_0} \quad (4.57)$$

in which Φ'_0 and Φ''_0 are of the form $AX + BX^2 + O(X^3)$ and $C + DX + EX^2 + O(X^3)$, respectively. Therefore, when $X \ll 1$,

$$\frac{\Phi'_0}{\Phi''_0} \approx \frac{A}{C} X = \frac{\cos^2 B \sin \varphi_0 \cos \varphi_0}{(1 - \cos^2 B \sin^2 \varphi_0)} \frac{\rho - \rho_0}{\rho_0} \quad (4.58)$$

Also, for $X \ll 1$

$$\frac{\Phi'^2_0}{\Phi''_0} \approx \frac{A^2}{C} X^2 = \frac{(kD) \cos^4 B \sin^2 \varphi_0 \cos^2 \varphi_0}{(1 - \cos^2 B \sin^2 \varphi_0)} \left(\frac{\rho - \rho_0}{D}\right)^2 \quad (4.59)$$

Using Eq. (4.33)

$$\begin{aligned} \Phi_\varrho = \Phi_1 = \Phi_0 - \frac{1}{2} \left(\frac{\Phi'^2_0}{\Phi''_0} \right) &= \frac{kD}{2} \left[(1 - \cos^2 B \cos^2 \varphi_0) \left(\frac{\rho - \rho_0}{D} \right)^2 - \right. \\ &\left. \frac{\cos^4 B \sin^2 \varphi_0 \cos^2 \varphi_0}{(1 - \cos^2 B \sin^2 \varphi_0)} \left(\frac{\rho - \rho_0}{D} \right)^2 \right] = \frac{kD}{2} \frac{1 - \cos^2 B}{1 - \cos^2 B \sin^2 \varphi_0} \left(\frac{\rho - \rho_0}{D} \right)^2 \end{aligned} \quad (4.60)$$

where $k = \frac{2\pi}{\lambda}$, is the wave number.

If,

$$\Phi_\varrho = \frac{\pi}{2} \left(\frac{\rho - \rho_0}{F} \right)^2 \quad (4.61)$$

where F is the Fresnel scale, then

$$F^2 = \frac{\lambda D}{2} \frac{1 - \cos^2 B \sin^2 \varphi_0}{\sin^2 B} \quad (4.62)$$

Eq. (4.62) shows the dependency of the Fresnel scale on the occultation geometry parameters.

The above analytical calculations, which lead to the formulation of the Fresnel scale based on the occultation geometry parameters, are very lengthy and tedious. Therefore, the symbolic computational capability of MATLAB was also used to confirm

the obtained formula in Eq. (4.62). The commands used in MATLAB for verification of this formula can be found at Appendix 4.1.

Assuming that

$$I = \int_0^{2\pi} \frac{e^{j\Phi(\rho_0, \varphi_0, \rho, \varphi)}}{|\bar{R}_c - \bar{\rho}|} d\varphi \quad (4.63)$$

and that $\varphi = \varphi_s$ is the answer to $\frac{d\Phi}{d\varphi} = 0$ (φ_s is the stationary point of Φ), then the above

integral can be estimated as

$$I \approx \int_0^{2\pi} \frac{e^{j\Phi(\rho_0, \varphi_0, \rho, \varphi)|_{\varphi=\varphi_s}}}{|\bar{R}_c - \bar{\rho}_s|} d\varphi \quad (4.64)$$

where,

$$\rho_s = (\rho, \varphi_s) \quad (4.65)$$

Using the Taylor series expansion

$$\begin{aligned} \Phi(\rho_0, \varphi_0, \rho, \varphi) |_{\varphi=\varphi_s} &\approx \Phi(\rho_0, \varphi_0, \rho, \varphi_s) + (\varphi - \varphi_s) \Phi'(\rho_0, \varphi_0, \rho, \varphi) |_{\varphi=\varphi_s} + \\ &\frac{(\varphi - \varphi_s)^2}{2} \Phi''(\rho_0, \varphi_0, \rho, \varphi) |_{\varphi=\varphi_s} \end{aligned} \quad (4.66)$$

in which the second term is zero, because φ_s is the stationary phase response, and

$\Phi'(\rho_0, \varphi_0, \rho, \varphi_s) = 0$. Therefore,

$$\Phi(\rho_0, \varphi_0, \rho, \varphi) |_{\varphi=\varphi_s} \approx \Phi(\rho_0, \varphi_0, \rho, \varphi_s) + \frac{(\varphi - \varphi_s)^2}{2} \Phi''(\rho_0, \varphi_0, \rho, \varphi) |_{\varphi=\varphi_s} \quad (4.67)$$

Assuming that $|\bar{R}_c - \bar{\rho}_s| \approx D$, and replacing Eq. (4.67) in Eq. (4.64) gives

$$I \approx \int_0^{2\pi} \frac{e^{j\Phi(\rho_0, \varphi_0, \rho, \varphi_s)} e^{\frac{j\Phi''(\rho_0, \varphi_0, \rho, \varphi_s)(\varphi - \varphi_s)^2}{2}}}{D} d\varphi \approx \int_{-\infty}^{+\infty} \frac{e^{j\Phi(\rho_0, \varphi_0, \rho, \varphi_s)} e^{\frac{j\Phi''(\rho_0, \varphi_0, \rho, \varphi_s)(\varphi - \varphi_s)^2}{2}}}{D} d\varphi \quad (4.68)$$

If

$$\frac{\pi}{2}\eta^2 = \frac{(\varphi - \varphi_s)^2}{2} \Phi''(\rho_0, \varphi_0, \rho, \varphi_s) \quad (4.69)$$

then

$$d\eta = \sqrt{\frac{\Phi''(\rho_0, \varphi_0, \rho, \varphi_s)}{\pi}} d\varphi \quad (4.70)$$

Eq. (4.68) can then be written as

$$I \approx \frac{e^{j\Phi(\rho_0, \varphi_0, \rho, \varphi_s)}}{D} \sqrt{\frac{\pi}{\Phi''(\rho_0, \varphi_0, \rho, \varphi_s)}} \int_{-\infty}^{+\infty} e^{j\frac{\pi}{2}\eta^2} d\eta = \frac{e^{j\Phi(\rho_0, \varphi_0, \rho, \varphi_s)}}{D} \sqrt{\frac{\pi}{\Phi''(\rho_0, \varphi_0, \rho, \varphi_s)}} (1+j) \quad (4.71)$$

From Eqs. (4.57), (4.58), and (4.61)

$$\varphi_s = \varphi_0 - \frac{\cos^2 B \cos \varphi_0 \sin \varphi_0}{1 - \cos^2 B \sin^2 \varphi_0} \frac{(\rho - \rho_0)}{\rho_0} \quad (4.72)$$

$$\Phi(\rho_0, \varphi_0, \rho, \varphi_s) = \frac{\pi}{2} \left(\frac{\rho - \rho_0}{F} \right)^2 \quad (4.73)$$

Using Eq. (4.56)

$$\Phi''(\rho_0, \varphi_0, \rho, \varphi_s) \approx \Phi''(\rho_0, \varphi_0, \rho, \varphi_0) = kD(1 - \cos^2 B \sin^2 \varphi_0) \left(\frac{\rho_0}{D} \right)^2 \quad (4.74)$$

Therefore, Eqs. (4.71) and (4.74) result in

$$I = \frac{e^{j\frac{\pi}{2} \left(\frac{\rho - \rho_0}{F} \right)^2}}{D} \sqrt{\frac{\lambda}{2D(1 - \cos^2 B \sin^2 \varphi_0) \left(\frac{\rho_0}{D} \right)^2}} (1+j) \quad (4.75)$$

By using Eq. (4.62)

$$I = \frac{e^{\frac{j\pi}{2}(\frac{\rho-\rho_0}{F})^2}}{D} \sqrt{\frac{\lambda^2}{2 \times 2F^2 \sin^2 B (\frac{\rho_0}{D})^2}} (1+j) \quad (4.76)$$

in which

$$\mu_0 = \sin B \quad (4.77)$$

Therefore, I can be written as

$$I = \frac{e^{\frac{j\pi}{2}(\frac{\rho-\rho_0}{F})^2}}{\rho_0} \frac{(1+j)}{2F} \frac{\lambda}{\mu_0} \quad (4.78)$$

Substituting Eq. (4.78) in Eq. (4.9) results in

$$\hat{T}(\rho_0) = \frac{\mu_0}{j\lambda} \int_0^\infty d\rho \rho T(\rho) \frac{e^{\frac{j\pi}{2}(\frac{\rho-\rho_0}{F})^2}}{\rho_0} \frac{(1+j)}{2F} \frac{\lambda}{\mu_0} = \frac{1-j}{2F} \int_{-\infty}^{+\infty} \frac{\rho}{\rho_0} T(\rho) e^{\frac{j\pi}{2}(\frac{\rho-\rho_0}{F})^2} d\rho \quad (4.79)$$

in which $\frac{\rho}{\rho_0} \approx 1$ for all ρ in the ring plane. Hence,

$$\hat{T}(\rho_0) = \frac{1-j}{2F} \int_{-\infty}^{+\infty} T(\rho) e^{\frac{j\pi}{2}(\frac{\rho-\rho_0}{F})^2} d\rho \quad (4.80)$$

4.3 Geometry File Reader

To calculate ring transmittance, some information regarding the occultation geometry is needed. This data is contained in a text ASCII file. A sample of a geometry file is shown in Appendix 4.2.

A geometry file contains information, such as, time signal is transmitted from Cassini, time at which signal is received on Earth, angles φ_0 and B , distance D , and

radius ρ_0 , as depicted in Figure (4.1). The parameters in this file are used to calculate the Fresnel scale according to Eq. (4.62). Other parameters such as the speed of the spacecraft can also be calculated using the geometry file. Figures (4.2) and (4.3) show the Fresnel scale, and spacecraft velocity versus the radius of ring intercept point. The MATLAB program, GeometryFilereader.m, used for reading data from a text geometry file, and then calculating other parameters such as Fresnel scale and spacecraft velocity can be found at Appendix 4.3.

Figure 4.2 Fresnel scale calculated from Cassini Rev7, May 2-3, 2005 (DOY 122-123) geometry file

Figure 4.3 Velocity of spacecraft calculated from the geometry file parameters. The blue plot is the velocity which had been calculated previously, and, has been used as a point of reference in this graph, to verify that the velocity calculated by GeometryFileReader.m is correct.

Distances between sample data given to the reconstruction algorithm are uniform. However, the samples measured by the spacecraft are not uniformly spaced in radius (samples are uniform in time). Therefore, the data measured should be interpolated and re-sampled at desired points, uniformly distanced. This is also done in GeometryFileReader.m, which uses the cubic spline data interpolation in MATLAB. The spline method fits a curve that is characterized by piecewise polynomials of the third order through the available data samples (non-uniform in distance), and then returns the data corresponding with the desired sample points (uniform in distance).

5 Effect of Fresnel Scale Uncertainty on Reconstructed Data

As Cassini moves along its trajectory, the characteristic Fresnel scale which is dependent on the occultation geometry varies. However, a constant Fresnel scale is utilized in the inverse-Fresnel filter for reconstructing the ring profile. Therefore, an error always exists between the actual Fresnel scale and the Fresnel scale used in the inverse-Fresnel filter. This error creates distortion in the reconstructed data, and limits the radial resolution. The distortion can be controlled and minimized if the offset does not exceed a certain limit. The objective of this chapter is to study the effect of Fresnel scale uncertainty on radial resolution, and, the nature and magnitude of the distortion it creates in the reconstructed data. An effective algorithm will also be proposed to reduce this distortion.

5.1 Effect of Fresnel Scale Mismatch on Reconstructed Data

Under ideal circumstances, where no practical constraints such as, finite data width, and uncertainty in the characteristic Fresnel scale exist, the equation below always holds true.

$$\mathfrak{I}(h_f(x))\mathfrak{I}(h_r(x)) = 1 \quad (5.1)$$

where,

$h_f(x)$: Forward-Fresnel filter

$h_f(x)$: Inverse-Fresnel filter

Therefore, as Eq. (5.2) suggests below, the reconstructed data would be the same as the original data.

Figure 5.1 System above represents the relationships between the ring profile $T(x)$, measured diffraction-limited ring profile $\hat{T}(x)$, and the reconstructed ring profile $\tilde{T}(x)$

$$\tilde{T}(x) = \hat{T}(x) * h_f(x) = T(x) * h_f(x) * h_i(x) = T(x) \quad (5.2)$$

If the conditions in the system shown in Figure (5.1) are not ideal, then the reconstructed transmittance will differ from the actual ring transmittance.

Let's assume that the only non-ideal condition is that the Fresnel scale in $h_i(x)$ varies from the true characteristic Fresnel scale by an offset of δF . Therefore,

$$F_i = F + \delta F \quad (5.3)$$

where,

F_i : The Fresnel scale used in the inverse-Fresnel filter

F : Actual Fresnel scale

Then,

$$h_i(x) = h_{i\delta F}(x) = \frac{1+j}{2(F+\delta F)} e^{-j\frac{\pi}{2}(\frac{x}{F+\delta F})^2} \quad (5.4)$$

$$\Im(h_{i\delta F}(x)) = \frac{1+j}{\sqrt{2}} e^{j2\pi(F+\delta F)^2 u^2} \quad (5.5)$$

where U is the Fourier domain. According to Eq. (2.28)

$$\mathfrak{I}(h_f(x)) = H_f(u) = \frac{1-j}{\sqrt{2}} e^{-j2\pi F^2 u^2} \quad (5.6)$$

From Eqs. (5.5), and (5.6), it is derived that

$$\mathfrak{I}(h_{\delta F}(x))\mathfrak{I}(h_f(x)) = e^{j2\pi F^2 (2\frac{\delta F}{F} + (\frac{\delta F}{F})^2)u^2} \quad (5.7)$$

Assuming that,

$$\left| \frac{\delta F}{F} \right| \ll 1 \quad (5.8)$$

and

$$\gamma^2 = 2 \frac{\delta F}{F} \quad (5.9)$$

Eq. (5.7) can be rewritten as

$$\mathfrak{I}(h_{\delta F}(x))\mathfrak{I}(h_f(x)) = e^{j2\pi F^2 \gamma^2 u^2} \quad (5.10)$$

From Figure (5.1)

$$\mathfrak{I}(\tilde{T}(x)) = e^{j2\pi F^2 \gamma^2 u^2} \mathfrak{I}(T(x)) \quad (5.11)$$

Eq. (5.11) suggests that the effect of Fresnel scale error alone, translates into a phase offset in the Fourier transform of the reconstructed data, and, the smaller the error, the smaller this offset will be. It can easily be seen from this equation that this phase offset will become zero if no error exists.

5.2 Effect of Fresnel Scale Uncertainty on Radial Resolution

To study the effect of Fresnel scale error and the limitation it imposes on radial resolution, a ring with an impulse profile will be reconstructed with an inverse-Fresnel filter, whose Fresnel scale has an offset of δF compared with the actual Fresnel scale.

Assuming that the ring gap is so thin that it can be represented by a delta dirac impulse function, then the object plane ring transmittance can be written as,

$$T_\delta(\rho) = \delta(\rho - \rho^*) \quad (5.12)$$

ρ : Radial distance in a cylindrical coordinate system

According to Eq. (2.24), the diffraction-limited image plane transmittance is derived to be

$$\hat{T}(\rho_0) = \frac{1-j}{2F} e^{j\frac{\pi}{2}(\frac{\rho^* - \rho_0}{F})^2} \quad (5.13)$$

Let's obtain the reconstructed data by applying the inverse-Fresnel transform in Eq. (2.34), whose Fresnel scale has an error of δF in regards with the true Fresnel scale, to Eq. (5.13).

$$\begin{aligned} \tilde{T}(\rho) &= \frac{1+j}{2(F+\delta F)} \int_{\rho-W/2}^{\rho+W/2} \frac{1-j}{2F} e^{j\frac{\pi}{2}(\frac{\rho^* - \rho_0}{F})^2} e^{-j\frac{\pi}{2}(\frac{\rho - \rho_0}{F+\delta F})^2} d\rho_0 \\ &= \frac{1}{2F^2(1+\frac{\delta F}{F})} \int_{\rho-W/2}^{\rho+W/2} e^{j\frac{\pi}{2}(\frac{\rho^* - \rho_0}{F})^2} e^{-j\frac{\pi}{2}((\rho - \rho_0)^2 \frac{1}{F^2(1+\frac{\delta F}{F})^2})} d\rho_0 \end{aligned} \quad (5.14)$$

If $\left| \frac{\delta F}{F} \right| \ll 1$, then $(1 + \frac{\delta F}{F})^{-2} \cong 1 - 2 \frac{\delta F}{F}$, and Eq. (5.14) can be re-written as

$$\tilde{T}(\rho) = \frac{1}{2F^2} \int_{\rho - W/2}^{\rho + W/2} e^{j\frac{\pi}{2}(\frac{\rho - \rho_0}{F})^2} e^{-j\frac{\pi}{2}((\frac{\rho - \rho_0}{F})^2(1 - 2\frac{\delta F}{F}))} d\rho_0 \quad (5.15)$$

Assuming that $\gamma^2 = 2 \frac{\delta F}{F}$, and $\frac{\rho_0 - \rho}{F} = X$

$$\tilde{T}(\rho) = \frac{1}{2F} \int_{-W/2F}^{W/2F} e^{j\frac{\pi}{2}(\frac{\rho^2 - \rho_0^2 + 2(XF + \rho)(\rho - \rho_0)}{F^2} + \gamma^2 X^2)} dX \quad (5.16)$$

which results in the impulse response

$$\tilde{T}(\rho) = H_{\delta F}(\rho) = \frac{1}{2F} \int_{-W/2F}^{W/2F} e^{j\frac{\pi}{2}(\frac{\rho - \rho_0}{F})^2} e^{j\frac{\pi}{2}(\frac{2X(\rho - \rho_0)}{F} + \gamma^2 X^2)} dX \quad (5.17)$$

In [1], resolution is defined to be

$$\Delta R_{\delta F} = \frac{1}{|H_{\delta F}(\rho^*)|^2} \int_{-\infty}^{+\infty} |H_{\delta F}(\rho - \rho^*)|^2 d\rho \quad (5.18)$$

The integral $\int_{-\infty}^{+\infty} |H_{\delta F}(\rho - \rho^*)|^2 d\rho$ represents the energy in $H_{\delta F}$. $\int_{-\infty}^{+\infty} |H_{\delta F}(\rho)|^2 d\rho$ also

represents the energy in $H_{\delta F}$. Hence,

$$\int_{-\infty}^{+\infty} |H_{\delta F}(\rho - \rho^*)|^2 d\rho = \int_{-\infty}^{+\infty} |H_{\delta F}(\rho)|^2 d\rho \quad (5.19)$$

$$\int_{-\infty}^{+\infty} |H_{\delta F}(\rho)|^2 d\rho = \int_{-\infty}^{+\infty} H_{\delta F}(\rho) H_{\delta F}^*(\rho) d\rho \quad (5.20)$$

From Eqs. (5.17), (5.19), and (5.20)

$$\int_{-\infty}^{+\infty} |H_{\delta F}(\rho)|^2 d\rho = \int_{-\infty}^{+\infty} \frac{1}{2F} e^{j\frac{\pi}{2}(\frac{\rho-\rho^*}{F})^2} \int_{-W/2F}^{W/2F} e^{j\frac{\pi}{2}(\gamma^2 X^2 + \frac{2(\rho-\rho^*)X}{F})} dX \frac{1}{2F} e^{-j\frac{\pi}{2}(\frac{\rho-\rho^*}{F})^2} \times$$

$$\int_{-W/2F}^{W/2F} e^{-j\frac{\pi}{2}(\gamma^2 X'^2 + \frac{2(\rho-\rho^*)X'}{F})} dX' d\rho \quad (5.21)$$

$$\int_{-\infty}^{+\infty} |H_{\delta F}(\rho)|^2 d\rho = \frac{1}{4F^2} \int_{-W/2F}^{W/2F} dX \int_{-W/2F}^{W/2F} e^{j\frac{\pi}{2}\gamma^2(X^2 - X'^2)} dX' \int_{-\infty}^{+\infty} e^{j\frac{\pi(\rho-\rho^*)(X-X')}{F}} d\rho \quad (5.22)$$

The analysis for calculating the right-most integral is explained below.

$$\delta(t-t_0) \xleftrightarrow{\mathfrak{F}} e^{-j2\pi ft_0} \quad (5.23)$$

Taking the inverse-Fourier transform of the right hand side expression in Eq.

(5.23) should result in the expression on the left hand side of the equation. Therefore,

$$\int_{-\infty}^{+\infty} e^{-j2\pi ft_0} e^{j2\pi ft} df = \int_{-\infty}^{+\infty} e^{j2\pi f(t-t_0)} df = \delta(t-t_0) \quad (5.24)$$

If

$$f = \rho, \quad t = \frac{X}{2F}, \quad \text{and} \quad t_0 = \frac{X'}{2F}$$

then Eq. (5.24) becomes

$$\int_{-\infty}^{+\infty} e^{j\pi \frac{\rho}{F}(X-X')} d\rho = \delta\left(\frac{X}{2F} - \frac{X'}{2F}\right) \quad (5.25)$$

As a consequence,

$$\int_{-\infty}^{+\infty} e^{j\frac{\pi(\rho-\rho^*)(X-X')}{F}} d\rho = e^{-j\pi \frac{\rho^*}{F}(X-X')} \delta\left(\frac{X}{2F} - \frac{X'}{2F}\right)$$

$$= 2Fe^{-j\pi \frac{\rho^*}{F}(X-X')} \delta(X-X') \quad (5.26)$$

By inserting Eq. (5.26) into Eq. (5.22), $\int_{-\infty}^{+\infty} |H_{\delta F}(\rho)|^2 d\rho$ is simplified into

$$\int_{-\infty}^{+\infty} |H_{\delta F}(\rho)|^2 d\rho = \frac{1}{2F} \int_{-W/2F}^{W/2F} dX \quad (5.27)$$

Therefore, the numerator in Eq. (5.18) is calculated to be

$$\int_{-\infty}^{+\infty} |H_{\delta F}(\rho)|^2 d\rho = \frac{W}{2F^2} \quad (5.28)$$

To be able to obtain the resolution, the denominator in Eq. (5.18) also needs to be calculated. If ρ is substituted with ρ^* in Eq. (5.17), then

$$|H_{\delta F}(\rho^*)|^2 = \left| \frac{1}{2F} \int_{-W/2F}^{W/2F} e^{j\frac{\pi}{2}\gamma^2 X^2} dX \right|^2 \quad (5.29)$$

If variable X is replaced with $\frac{\eta}{\gamma}$ in Eq. (5.29) above, then

$$\begin{aligned} |H_{\delta F}(\rho^*)|^2 &= \left| \frac{1}{2F\gamma} \int_{-W\gamma/2F}^{W\gamma/2F} e^{j\frac{\pi}{2}\eta^2} d\eta \right|^2 \\ &= \frac{1}{4F^2\gamma^2} \left| C\left(\frac{W\gamma}{2F}\right) + jS\left(\frac{W\gamma}{2F}\right) - C\left(-\frac{W\gamma}{2F}\right) - jS\left(-\frac{W\gamma}{2F}\right) \right|^2 \end{aligned} \quad (5.30)$$

where C and S are the cosine and sine Fresnel integral functions as described in section 2.3, Eq. (2.19).

Knowing that

$$\begin{aligned} C(x) &= -C(-x) \\ S(x) &= -S(-x) \end{aligned} \quad (5.31)$$

and assuming that

$$Q = \frac{W\gamma}{2F} \quad (5.32)$$

Eq. (5.30) can be re-written as

$$|H_{\delta F}(\rho^*)|^2 = \frac{1}{F^2\gamma^2}(C^2(Q) + S^2(Q)) \quad (5.33)$$

From Eqs. (5.18), (5.19), (5.28), and (5.33)

$$\Delta R_{\delta F} = \frac{W}{2F^2} \times \frac{F^2\gamma^2}{C^2(Q) + S^2(Q)} \quad (5.34)$$

The equation above can be simplified further if $\Delta R_{\delta F}$ is divided and multiplied by $\frac{4F^2}{W^2}$.

$$\begin{aligned} \Delta R_{\delta F} &= \frac{W}{2F^2} \times \frac{F^2\gamma^2}{C^2(Q) + S^2(Q)} \times \frac{4F^2}{W^2} \times \frac{W^2}{4F^2} \\ &= \frac{2F^2}{W} \times \frac{Q^2}{C^2(Q) + S^2(Q)} \end{aligned} \quad (5.35)$$

$$\Delta R_{\delta F} / \gamma F = \frac{Q}{C^2(Q) + S^2(Q)} \quad (5.36)$$

Eq. (5.36) above has been used for plotting Figure (5.2), which shows how resolution is affected in the presence of an error in the knowledge of F. As it is observed in the plot, the resolution improves with increasing filter width (W), up to $Q \approx 1$, after which it worsens and then starts following an oscillating pattern. Hence, in the presence of a Fresnel scale offset, the best resolution is achieved at $Q \approx 1$, or, when $W = \frac{2F}{\gamma}$, for which $\Delta R_{\delta F} = 1.25\gamma F$. For this reason it is important that as the inverse-Fresnel filter is

being applied to the diffraction-limited data, it is updated with a new Fresnel scale when necessary, such that, the radial resolution always falls into the linear region of the plot ($Q \leq 1$).

Figure 5.2 This figure shows how the resolution changes as the filter length increases in the presence of an error in the knowledge of the characteristic Fresnel scale. The plot was generated by `VaryingFResolution.m`, and can be found at Appendix 5.1.

For example, if one needs to achieve a normalized radial resolution (with respect to F) of 0.1, then

$$\frac{\Delta R_{\delta F}}{F} = 1.25\gamma = 0.1 \quad (5.37)$$

Therefore, the maximum allowable γ to still achieve this resolution would be

$$\gamma = \frac{0.1}{1.25} = 0.08 \quad (5.38)$$

and, a normalized filter (with respect to F) of width 25 would be required.

$$\frac{W}{F} = \frac{2}{\gamma} = 25 \quad (5.39)$$

In other words, in the presence of an error with γ of 0.08 or $\frac{\delta F}{F} = 0.0032$, the best achievable normalized resolution is 0.1, and, the normalized filter width which needs to be applied to the data for obtaining this resolution is 25. If the filter width is increased beyond this point, resolution is deteriorated and will fall in the non-linear region of Figure 5.2 ($Q > 1$).

According to Eq. (2.39), if no Fresnel scale error existed, a normalized filter width of 20 would have been needed to obtain a normalized radial resolution of 0.1.

$$\begin{aligned} \frac{\Delta R_w}{F} &= \frac{2F}{W} \\ W_N &= \frac{2}{\Delta R_{wN}} \end{aligned} \quad (5.40)$$

Figure (5.3) shows the maximum resolution that can be achieved for a given $\frac{\delta F}{F}$.

Figure (5.4) shows the filter width required for obtaining the maximum resolution in the presence of an error $\frac{\delta F}{F}$. As discussed previously, the filter width can not be increased beyond this value. Figure (5.5) shows the number of points needed to represent the filter, if the sampling interval is at its largest allowed value of $\frac{\Delta R}{2}$. Figures (5.3), (5.4), and (5.5) were generated by NormalizedResolutionAnalysis4VaryingF.m, and can be found at Appendix 5.2.

Figure 5.3 Shows the maximum achievable normalized resolution (with respect to F) in the presence of error in the characteristic Fresnel scale.

Figure 5.4 This figure shows the normalized filter width required for obtaining the maximum normalized resolution in the presence of an error in the characteristic Fresnel scale.

Figure 5.5 As it was previously discussed in chapter 2, the sampling interval of the filter needs to be at least half of the required resolution ($dR/2$). This plot shows the number of filter sample points required for obtaining the maximum achievable normalized resolution in the presence of an error in the characteristic Fresnel scale. However, as the number of filter points increases, so does the computation complexity.

5.3 Filter Update Algorithms

Every ring transmittance data sample has a different Fresnel scale, because the spacecraft continuously moves and F depends on the occultation geometry. But inverse-Fresnel filter assumes a constant F for the block of data it is being applied to. Therefore, an error exists between the characteristic Fresnel scale used in the filter and the true Fresnel scale, which causes distortion in the reconstructed data. From discussions in section 5.2, it is valid to state that for minimizing this distortion, the inverse-Fresnel filter

should be updated with a new F , as it is being applied to the measured data, such that,

$\frac{\Delta R_{\delta F}}{\gamma F}$ always remains in the linear region of Figure (5.2). Below, two algorithms are

proposed for updating the filter's Fresnel scale.

5.3.1 UpdateF.m

This algorithm, assumes that the characteristic Fresnel scale of a block of data captured by the spacecraft (e.g. Cassini), either increases or decreases as the spacecraft moves. In other words, if F_m is the characteristic Fresnel scale matrix for the measured sample points, then the elements in F_m either follow an ascending or descending pattern. Thus, a recursive binary search can be used for updating of F .

Assuming that $F_0 < F_1 < F_2 < \dots < F_n$ in a block of data, maximum allowable δF is calculated for the largest Fresnel scale in the block from Eq.(5.37), and Eq.(5.9), because the minimum allowable δF (error in the knowledge of Fresnel scale) in the block belongs to the largest F . If

$$F_n - F_0 > \delta F \quad (5.41)$$

then the length of the block is reduced , a new δF is calculated and a new search begins within this block based on criterion 5.41. If

$$F_n - F_0 < \alpha \delta F \quad (5.42)$$

where, α is a number close to 1, then the length of the block will be increased. In this thesis,

$$\alpha = 0.98 \quad (5.43)$$

However, if

$$\alpha\delta F < F_n - F_0 < \delta F \quad (5.44)$$

a block of data has been found which can be assumed to have a constant F , which, is equals to an average of the Fresnel scales for all the data points within the found block.

The search is continued until the end of the original data block is reached.

For example in the block shown in Figure (5.6), if

$$F_0 < F_1 < F_2 < \dots < F_n \quad (5.45)$$

and for convenience

$$n = 2k \quad (5.46)$$

then,

$$\delta F = 0.5 \frac{\Delta R^2}{F_{\max} \times 1.25^2} \quad (5.47)$$

where, $F_{\max}$ is the maximum Fresnel scale in the block.

Figure 5.6 Block of data measured by Cassini

If condition in Eq. (5.41) persists, then, the above block is reduced to the block in

Figure (5.7) below.

Figure 5.7 Block in Figure (5.6) reduced in length

A new δF is then calculated from Eq. (5.47). If,

$$F_{n/2} - F_0 < 0.98\delta F \quad (5.48)$$

then, the block will expand to be

F0	F1	...	F _{3n/4}
----	----	-----	-------------------

Figure 5.8 Block in Figure (5.7) expanded in length

A new δF is again calculated. If,

$$F_{3n/4} - F_0 > \delta F \quad (5.49)$$

the block will be reduced to

F0	F1	...	F _{5n/8}
----	----	-----	-------------------

Figure 5.9 Block in Figure (5.8) reduced in length

A new δF is once more calculated. If,

$$0.98\delta F < F_{5n/8} - F_0 < \delta F \quad (5.50)$$

the block in Figure (5.8) is a block which can have a constant Fresnel scale equals the average value of all Fresnel scales within this block. The search is continued in the same way, until the end of the full data block is reached.

5.3.2 BlockUpdateF.m

This alternative algorithm does not assume anything about the Fresnel scale matrix F_m . Fresnel scale matrix is initially divided into blocks of length specified by the user. If δF within any block exceeds the maximum allowable value, then the user is asked to reduce the initial block length. Otherwise, the algorithm will merge as many

blocks as possible to create larger blocks so that none of the sample points within the block violate the rule that requires $\frac{\Delta R_{\delta F}}{\gamma F}$ to remain in the linear region of Figure (5.2).

5.3.3 TestUpdateAlgorithms.m

This algorithm utilizes data from Cassini's first diametric radio occultation of Saturn's rings and atmosphere (Rev7, May 2-3, 2005) to test the two algorithms above.

UpdateF.m, which uses a binary search algorithm for allocating blocks with fixed Fresnel scale, segments this block of data into:

Block 1: 1 to 1292, fixed F = 1.3125

Block 2: 1293 to 3755, fixed F = 1.477

Block 3: 3756 to 6406, fixed F = 1.6654

Block 4: 6407 to 9008, fixed F = 1.8349

Block 5: 9009 to 10877, fixed F = 1.9904

Block 6: 10878 to 12001, fixed F = 2.1382

1	1292	3755	6406	9008	10877	12001
1.3125	1.477	1.6654	1.8349	1.9904	2.1382	

BlockUpdateF.m, with initial block length of 2900 segments the data into:

Block 1: 1 to 2900, fixed F = 1.3847

Block 2: 2901 to 5800, fixed F = 1.6147

Block 3: 5801 to 8700, fixed F = 1.8059

Block 4: 8701 to 11600, fixed F = 2.0122

Block 5: 11601 to 12001, fixed $F = 2.185$

1	2900	5800	8700	11600	12001
1.3847	1.6147	1.8059	2.0122	2.185	

The original block of data, F_m , has Fresnel scales which monotonically increase as the sample index increases (in this case, as spacecraft moves closer to Saturn, also called Ingress). The original Fresnel scales for each sample point at the edge of the blocks above are:

$$F_m(1) = 1.2489$$

$$F_m(1292) = 1.3734$$

$$F_m(2900) = 1.5087$$

$$F_m(3755) = 1.5736$$

$$F_m(5800) = 1.7139$$

$$F_m(6406) = 1.7527$$

$$F_m(8700) = 1.8983$$

$$F_m(9008) = 1.9192$$

$$F_m(10877) = 2.0719$$

$$F_m(11600) = 2.1564$$

$$F_m(12001) = 2.2152$$

5.4 Effect of Fresnel Scale Mismatch on Impulse Response

In this section, experiments will be performed utilizing MATLAB to study the effect of an offset in the knowledge of the characteristic Fresnel scale on reconstructing the profile of a very thin gap, resembling a delta-dirac function, in the rings.

According to Eq. (5.37) and prior discussions in this chapter, for a normalized resolution of 0.1, the maximum acceptable γ is 0.08. Therefore $\frac{\delta F}{F}$ must be equals or smaller than 0.0032 to almost eliminate distortion.

Figures (5.10) – (5.13), show the reconstructed profiles of an impulse response, in the presence of an error, compared with the reconstructed transmittance in the presence of no error. In this case $\frac{\delta F}{F} = 0.01$, which is larger than 0.0032. Therefore, significant

distortion an be seen in the reconstructed data. For $\frac{\delta F}{F} = 0.01$ or $\gamma = 0.141$, the filter length used is 100 km (code uses Eq. (2.39) for calculating filter length), therefore

$$Q = \frac{W\gamma}{2F} = 1.41 \quad (5.51)$$

According to Figure (5.3), $\frac{\Delta R_{\delta F}}{\gamma F}$ would be 1.78, which would make the resolution 1.25 km. Figures (5.10) and (5.11) demonstrate that in the presence of an error δF in knowledge of F , distortion is created in the main lobe of the impulse response; the main lobe is widened, and, side lobes are at higher levels. Therefore, one can expect the resolution to worsen, as predicted, and oscillations to occur in the reconstructed data. Resolution will improve and oscillation levels will dampen as error decreases.

Figure 5.10 Reconstructed magnitude(dB) of an impulse response, using a filter of 100 km wide, with and without error in F.

Figure 5.11 Reconstructed magnitude of an impulse response, using a filter of 100 km wide, with and without error in F.

Figure 5.12 Reconstructed phase (degrees) of an impulse response, using a filter of 100 km wide, with and without error presence in F.

Figure 5.13 A zoom on a region in Figure (5.12)

Figure (5.12) shows the phase of the impulse response in the presence of an error compared with no error in F . There are several points to pay attention to:

- Because of δF existence, the zero-crossing points do not exactly overlap the theoretical zero-crossing points.
- At zero-crossings, there should be a 180 degrees change in the phase, because of the sign change in the impulse response. However, this is not true when an error of δF exists. Most of the times, sample points occur before and after the zero-crossings; thus, going from one point to the next, creates a phase change of less than, but close to 180 degrees.
- Sudden peaks which occur at zero crossings are due to infinitesimal computational errors. The real and imaginary parts of the reconstructed transmittance at zero crossings (or close to zero-crossings, due to sampling) are very small, and in the vicinity of zero. However, the ratio of the imaginary part to the real part can vary from a small number to tens of hundreds, which can sometimes result in large, but meaningless, phases. For example if imaginary part is 10^{-12} , and real part is 10^{-14} , phase is 89.5 degrees.
- In the presence of $\frac{\delta F}{F} = 0.01$ or $\gamma^2 = 0.02$, a phase offset of approximately $180^\circ(2\pi F^2 \gamma^2)$ exists compared with the case where $\delta F = 0$. This is an impromptu guess deduced from Eq. (5.11), which is illustrated in Figure (5.13) and is also supported with other examples in this chapter. The

phase offset does not obey this ad-hoc guess in the vicinity of the main-lobe, which is distorted and widened by the presence of δF .

Figure 5.14 Reconstructed magnitude(dB) of an impulse response, using a filter of 100 km wide, with and without error presence in F. Error is very small here.

Figure 5.15 Reconstructed magnitude of an impulse response, using a filter of 100 km wide, with and without error presence in F. Error is very small here.

Figure 5.16 Reconstructed phase (degrees) of an impulse response, using a filter of 100 km wide, with and without error presence in F. Error is very small here.

Figure 5.17 Closer caption of a region in Figure (5.16)

Figures (5.14) – (5.17), show the reconstructed profiles of an impulse in the presence of an error compared with the reconstructed transmittance in the presence of no error. In this case, $\frac{\delta F}{F} = 0.001$, which is smaller than 0.0032, thus, the distortion should be negligible. The magnitude responses in Figures (5.14) - (5.15) almost show no sign of distortion. The phase response however, does show a little distortion. The following can be concluded from Figure (5.16), and Figure (5.17):

- The offset between the zero-crossings has reduced as δF also reduced from 0.01 to 0.001.
- In the presence of $\frac{\delta F}{F} = 0.001$, or, $\gamma^2 = 0.002$, a phase offset of approximately $18^\circ(2\pi F^2 \gamma^2)$ exists compared with the case where $\delta F = 0$.

Figure (5.18) shows how the phase offset becomes negligible for very small errors in F. Figure (5.20) demonstrates this offset to be $0.018^\circ(2\pi F^2 \gamma^2 = 2\pi \times 25 \times 0.000002)$. Figure (5.19) shows how the zero-crossings of the two cases (with and without error) almost over-lap. Finally Figures (5.21) and (5.22) illustrate how the reconstructed profile merges with the theoretical profile, when no error exists in the knowledge of the characteristic Fresnel scale.

Figure 5.18 Reconstructed phase (degrees) of an impulse response, using a filter of 100 km wide, with and without error presence in F. Error is negligible here.

Figure 5.19 A closer caption of Figure (5.18)

Figure 5.20 A closer caption of Figure (5.19)

Figure 5.21 Reconstructed phase (degrees) of an impulse response, using a filter of 100 km wide, with no error in F.

Figure 5.22 A closer caption of Figure (5.21)

From discussions and supporting figures above, it can be concluded that in the presence of δF , radial resolution of reconstructed data worsens, peak-to-peak levels in the oscillations present in the reconstructed magnitude profiles increase, and an offset of approximately $2\pi F^2 \gamma^2$ exists in the reconstructed phase profile. However, all these distortions and offsets become small as the error δF becomes small.

5.5 Effect of Fresnel Scale Mismatch on Edges

In this section, figures will be used to further confirm that error in knowledge of F causes oscillations that appear as overshoots in the free-space signal level and undershoots in the gray screen region of the reconstructed magnitude. Significant error is

also viewed in the reconstructed phase. Supporting figures validate that these distortions in the reconstructed profile reduce as the error decreases.

5.5.1 Phase Distortion

This section examines the reconstructed phase in the presence of δF . The Fresnel scale is 5 km, and, the reconstruction is performed for a desired resolution of 0.5 km. Of course, using a Kaiser window with the β of 3.5 worsens the resolution to about 1 km. As discussed in the previous sections, the presence of δF also affects the resolution. From Eq. (5.37), a $\frac{\delta F}{F}$ of less than 0.0032 will keep the resolution in the linear region of Figure (5.2).

Figures (5.23) - (5.26) show how the phase distortion reduces as $\frac{\delta F}{F}$ reduces.

Figures (5.25) and (5.27), for which $\frac{\delta F}{F} \leq 0.0032$, show significantly less phase

distortion compared to Figures (5.24) and (5.23), for which $\frac{\delta F}{F} > 0.0032$.

Figure (5.27) shows that even with a very small error in the knowledge of Fresnel scale, ($\frac{\delta F}{F}$), a phase offset still exists. The phase offset oscillations start at a distance of 1400 km, which is expected, because, the filter width used in the algorithm is 100 km (according to Eq. (2.39)), and, the simulated data has a sudden jump at 1450 km. To prove that the cause of phase distortion is not the sudden jump that appears in the simulated diffraction-limited data, Figures (5.28) and (5.29) were created by

reconstructing the ring profile from actual theoretical diffraction-limited edge profile. Figures (5.27) and (5.28) show how a sudden jump is seen in the reconstructed phase profile at the location where the filter intersects with the edge, as it is applied to the diffraction-limited data.

For fast computation of Fresnel integrals that inevitably appear in the image plane transmittance formula, the $fcs()$ function (discussed in Chapter 2) is used. This function however, is only accurate to some number of decimal points, hence, Figure (5.29) also shows phase distortion for the case where δF is zero, which would actually not have been the case if $fcs()$ function was more accurate. As a matter of fact, the phase distortion in Figure (5.29) seems to be like Figure (5.28), which, suggests that the error in the $fcs()$ function is big enough to be behaving as if a $\frac{\delta F}{F}$ of 10^{-7} exists. However, Figure (5.30) demonstrates how the reconstructed phase becomes almost zero (10^{-12}) for $\delta F = 0$.

Figure 5.23 Comparing reconstructed phase versus simulated diffraction-limited phase and the actual phase. $\frac{\delta F}{F} = 0.2, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$.

Figure 5.24 Comparing reconstructed phase versus simulated diffraction-limited phase and the actual phase. $\frac{\delta F}{F} = 0.01, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$.

Figure 5.25 Comparing reconstructed phase versus simulated diffraction-limited phase and the actual phase. $\frac{\delta F}{F} = 0.001, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$.

Figure 5.26 A closer look at Figure (5.25)

Figure 5.27 Phase reconstructed from simulated diffraction-limited data

for $\frac{\delta F}{F} = 10^{-7}$, $\Delta R = 0.5 \text{ km}$, $F = 5 \text{ km}$. Even for such infinitesimal error, phase distortion exists.

Figure 5.28 Phase reconstructed from theoretical diffraction-limited data

for $\frac{\delta F}{F} = 10^{-7}$, $\Delta R = 0.5 \text{ km}$, $F = 5 \text{ km}$.

Figure 5.29 Phase reconstructed from theoretical diffraction-limited data

for $\frac{\delta F}{F} = 0$, $\Delta R = 0.5 \text{ km}$, $F = 5 \text{ km}$.

Figure 5.30 Phase reconstructed from simulated diffraction-limited data

for $\frac{\delta F}{F} = 0$, $\Delta R = 0.5 \text{ km}$, $F = 5 \text{ km}$.

5.5.2 Magnitude Distortion

Figures (5.31) and (5.36) show how the reconstructed profile resembles that of the diffraction-limited profile for a $\frac{\delta F}{F}$ of 0.2, which is much greater than 0.0032 derived from Eq. (5.37); the oscillations are therefore expected.

$\frac{\delta F}{F}$ in Figures (5.32), (5.33), (5.37), (5.38), and (5.39) is 0.01, which is still greater than 0.0032. Figures (5.32) and (5.39) demonstrate the under-shoot, and the oscillations in the gray ring region. The overshoot in the free space region is not very well seen in Figures (5.32) and (5.37). A closer view as illustrated in Figures (5.33) and (5.38) shows the free-space overshoot more clearly.

Figures (5.34), (5.35), (5.40), and (5.41) verify that the oscillations, under-shoots, and over-shoots have almost disappeared.

Figure 5.31 Comparing reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude. $\frac{\delta F}{F} = 0.2, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$. According to Eq. (5.37), $\frac{\delta F}{F} > 0.0032$ is unacceptable and causes significant oscillations.

Figure 5.32 Comparing reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude. $\frac{\delta F}{F} = 0.01, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$. According to Eq. (5.37), $\frac{\delta F}{F} > 0.0032$ is unacceptable and causes significant oscillations.

Figure 5.33 A closer look at Figure (5.32)

Figure 5.34 Comparing reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude. $\frac{\delta F}{F} = 0.001$, $\Delta R = 0.5 \text{ km}$, $F = 5 \text{ km}$. According to Eq. (5.37), $\frac{\delta F}{F} > 0.0032$ is unacceptable and causes significant oscillations. In this case, $\frac{\delta F}{F} = 0.001$.

Figure 5.35 A closer look at Figure (5.34)

Figure 5.36 Comparing reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude. $\frac{\delta F}{F} = 0.2$, $\Delta R = 0.5$ km, $F = 5$ km. According to Eq. (5.37), $\frac{\delta F}{F} > 0.0032$ is unacceptable and causes significant oscillations.

Figure 5.37 Comparing reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude. $\frac{\delta F}{F} = 0.01, \Delta R = 0.5 \text{ km}, F = 5 \text{ km}$. According to Eq. (5.37), $\frac{\delta F}{F} > 0.0032$ is unacceptable and causes significant oscillations.

Figure 5.38 A closer look at Figure (5.37), highlighting the overshoot in the free-space region.

Figure 5.39 A closer look at Figure (5.37), highlighting the undershoot in the gray ring region.

Figure 5.40 Comparing reconstructed magnitude versus simulated diffraction-limited magnitude and the actual magnitude. $\frac{\delta F}{F} = 0.001$, $\Delta R = 0.5 \text{ km}$, $F = 5 \text{ km}$. According to Eq. (5.37), $\frac{\delta F}{F} > 0.0032$ is unacceptable and causes significant oscillations. In this case, $\frac{\delta F}{F} = 0.001$.

Figure 5.41 A closer look at Figure (5.40)

5.6 Applying the Reconstruction Algorithm to Three Gaps with Different Fresnel Scales

The purpose of this section is to test the reconstruction algorithm. To ensure that the reconstruction algorithm effectively updates its Fresnel scale, the algorithm is applied to simulated image plane transmittance diffraction-limited data, which is “measured” in the presence of three gaps in a gray screen. The region for each gap has its own Fresnel scale. It will be shown that the data reconstructed by the reconstruction algorithm, which employs the Fresnel scale update technique, contains significantly less distortion than the reconstruction algorithm which utilizes a fixed Fresnel scale.

5.6.1 Gaps with the same Widths

In this section, it is assumed that the ring plane (gray screen) contains three gaps at distances 2000, 4000 and 6000 km. Each gap has a width of 100 km. The Fresnel scale is 10 km. The desired resolution in the reconstruction algorithm is 0.4 km. From prior discussions, it is already known that the effect of using a Kaiser window and error in the knowledge of Fresnel scale degrades the actual achieved resolution. The Fresnel scales for the first, second, and third gaps are 10, 10.003, and 10.006, respectively. From Eq. (5.37), the Fresnel scale should be updated before a maximum δF of 0.00512 is reached.

Figures (5.42) - (5.51) show the reconstructed profiles for when the reconstruction algorithm updates the Fresnel scale, as the inverse-Fresnel filter is applied to the image plane transmittance data. Figures (5.52) – (5.59) however, show the reconstructed profiles when the inverse-Fresnel filter uses a fixed Fresnel scale of 10.006 for reconstruction.

Figure (5.43) shows less phase distortion compared with Figure (5.53). Figures (5.54) and (5.55) show more under-shoot for the first and second gaps compared to Figures (5.45) and (5.46). The undershoot for the third gap in Figure (5.56) is almost the same as that of Figure (5.47), and the reason for this is that the fixed Fresnel reconstruction algorithm uses the Fresnel scale of the third gap. Consequently, the overshoot for the first and second gaps, as demonstrated in Figure (5.57) and Figure (5.58), is larger than the overshoot in Figures (5.49) and (5.50), and, is about the same for the third gap in both cases (Figures (5.51) and (5.59)).

Figure 5.42 Comparing reconstructed phase versus diffraction-limited and actual phase for three gaps with the same width of 100km, but, different Fresnel scales. A reconstruction algorithm that updates the Fresnel scale has been used.

Figure 5.43 A closer look at Figure (5.42)

Figure 5.44 Comparing reconstructed magnitude versus diffraction-limited and actual magnitude for three gaps with the same width of 100km, but, different Fresnel scales. A reconstruction algorithm that updates the Fresnel scale has been used.

Figure 5.45 A closer look at the reconstructed profile of the first gap in Figure (5.44).

Figure 5.46 A closer look at the reconstructed profile of the second gap in Figure (5.44).

Figure 5.47 A closer look at the reconstructed profile of the third gap in Figure (5.44)

Figure 5.48 Reconstructed magnitude for three gaps with the same width of 100km, but different Fresnel scales. A reconstruction algorithm that updates the Fresnel scale has been used.

Figure 5.49 A closer look at the reconstructed profile of the first gap in Figure (5.48).

Figure 5.50 A closer look at the reconstructed profile of the second gap in Figure (5.48).

Figure 5.51 A closer look at the reconstructed profile of the third gap in Figure (5.48).

Figure 5.52 Comparing reconstructed phase versus diffraction-limited and actual phase for three gaps with the same width of 100 km, but different Fresnel scales. A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.

Figure 5.53 A closer look at Figure (5.52)

Figure 5.54 A close look at the reconstructed magnitude profile of the first gap. A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.

Figure 5.55 A close look at the reconstructed magnitude profile of the second gap. A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.

Figure 5.56 A close look at the reconstructed magnitude profile of the first gap. A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.

Figure 5.57 A close look at the reconstructed magnitude profile of the first gap. A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.

Figure 5.58 A close look at the reconstructed magnitude profile of the second gap. A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.

Figure 5.59 A close look at the reconstructed magnitude profile of the third gap. A reconstruction algorithm that uses a fixed Fresnel scale of 10.006 km has been used.

5.6.2 Gaps with Different Widths

It was illustrated with supporting figures in section 5.6.1, that, using a reconstruction algorithm which updates the Fresnel scale of the inverse-Fresnel filter accordingly with Eq. (5.37) can be very effective in reducing distortion. In the previous section, three gaps with the same width of $10F$, each with different Fresnel scales, were used. The purpose of this section is to illustrate that the reconstruction algorithm can reconstruct gaps smaller than F . Hence, three gaps with same characteristics as mentioned above, except with widths of 100, 10, and 5 km, respectively, are simulated and then reconstructed. It is obvious from Figures (5.60) – (5.67) that the reconstructed object plane consists of three gaps, first one starting at 2000 km with a width of 100 km, second one starting at 4000km with a width of 10 km, and the third one starting at 6000 km with a width of 5 km. Hence, it is once more validated that the reconstruction algorithm can effectively eliminate diffraction from measured data.

Figure 5.60 Comparing reconstructed phase versus diffraction-limited and actual phase for three gaps with widths of 100($10F$), 10(F), and 5($F/2$) km respectively, but different Fresnel scales. A reconstruction algorithm that updates the Fresnel scale has been used.

Figure 5.61 A closer look at the first gap (located at 2000-2100km) in Figure (5.60). A reconstruction algorithm that updates the Fresnel scale has been used.

Figure 5.62 A closer look at the second gap (located at 4000-4010km) in Figure (5.60). A reconstruction algorithm that updates the Fresnel scale has been used.

Figure 5.63 A closer look at the third gap (located at 6000-6005km) in Figure (5.60). A reconstruction algorithm that updates the Fresnel scale has been used. True $F=10.006$ for this gap.

Figure 5.64 Comparing reconstructed magnitude versus diffraction-limited and actual magnitude for three gaps with widths of $100(10F)$, $10(F)$, and $5(F/2)$ km respectively, but different Fresnel scales. A reconstruction algorithm that updates the Fresnel scale has been used.

Figure 5.65 A closer look at the first gap (located at 2000-2100km) in Figure (5.64). A reconstruction algorithm that updates the Fresnel scale has been used.

Figure 5.66 A closer look at the second gap (located at 4000-4010km) in Figure (5.64). A reconstruction algorithm that updates the Fresnel scale has been used.

Figure 5.67 A closer look at the third gap (located at 6000-6005 km) in Figure (5.64). A reconstruction algorithm that updates the Fresnel scale has been used.

6 Summary and Conclusions

Work presented in this thesis describes the need, and investigates and implements algorithms, for processing data captured by a spacecraft occulting Saturn's rings in order to reveal detailed ring features.

Radio occultation is a measurement technique used to profile planetary ring structure. During radio occultation, a spacecraft orbiting Saturn's rings transmits a sinusoidal signal which is perturbed by ring material, and then received on Earth. The perturbed signal intensity reveals radial ring structure. However, measurements of structure that vary on a scale smaller or comparable to the Fresnel scale are diffraction-limited. Hence, detailed profiles of many ring features, such as, edges, narrow gaps and ringlets, density and bending waves, and other yet to be revealed phenomena remain masked by their diffraction patterns. Therefore, techniques must be used to eliminate these diffractions from measured data, in order to reveal the hidden ring features. The scope of this thesis is to study and implement such techniques following approaches originally introduced in [1].

"Inverse-Fresnel filtering" of the measured data, Eq. (1.7), is a technique used for eliminating such diffractions and reconstructing ring profiles [1]. But, Eq. (1.7) is an ideal form of the inverse-Fresnel transform. In real life, many limitations and constraints are imposed upon the ideal Inverse-Fresnel transform of Eq. (1.7), which, introduce artifacts into the reconstructed ring profiles. It is important to investigate the impact of these artifacts on the reconstruction process so as to ensure reliable interpretation of the results. Some of these constraints are, limited data width, variant Fresnel scale (variant

observation geometry), system phase stability, system thermal noise, and phase approximations [1].

One of the real-life constraints imposed upon the ideal inverse-Fresnel transform is limited data width which limits the boundaries of the integral in Eq. (1.7) (inverse-Fresnel transform). Therefore, limited data width causes limited radial resolution and leakage. This is illustrated by deriving and studying the impulse response of an inverse-Fresnel transform with limited data width. To achieve finer resolution, data segments of larger width must be used. The larger the data width is, the more the data points, and hence the more the computational complexity of the reconstruction algorithm.

Both the forward and inverse-Fresnel transforms were implemented using MATLAB. The Fast Fourier transform was used for the implementations to reduce the computational complexity of these transforms, because of large size of the data blocks used to achieve fine radial resolution. The MATLAB implementations of the forward and inverse-Fresnel transforms were thoroughly tested for accuracy by comparing the results produced by the implementations versus expected theoretical results, when possible. Using the MATLAB implementations, it is shown how limited data width limits radial resolution and causes leakage. It is also shown that leakage (and hence, aliasing) can be reduced if filters are windowed using a Kaiser window, and that reducing the leakage worsens the radial resolution. Thus, a trade-off is made between the resolution and leakage.

Another artifact caused by limited filter width in the reconstructed ring profile is a jump at the location where the filter reaches any sudden onset in the ring structure. The

artificialness of the jump in the reconstructed profile can be understood because the jump is small (< 2 dBs) and is located at a distance half the filter width from a significant onset in the profile (see, for example, Figure 2.33).

Also investigated are aliasing effects introduced into the reconstructed signal when the filter used in the reconstruction algorithm is sampled at the wrong rate. It is shown by studying MATLAB generated simulations that it is necessary to sample the filter and the measured data at least every $\frac{\Delta R_w}{2}$, in order to neglect the effect of aliasing, where ΔR_w is the desired spatial resolution.

The thesis proposes two methods, direct width measurement and slope method, for measuring ring thickness. The methods are based on models of the behavior of the optical depth (which is a measurement of signal attenuation by ring material) in the vicinity of ring edges. Given a ring of finite edge thickness, at obliqueness incidence angle, the optical depth transition from ring material to free-space is not abrupt and takes place over some width. The transition is linear and the ring edges are called linearly tapered edges. By characterizing linearly tapered edges in MATLAB, and simulating the diffracted data generated by such edges, and studying the effect of the transition width on diffracted and reconstructed ring profiles, it is illustrated how the two methods above can be used to measure edge thickness. It is observed in the many numerical simulations conducted that, as a general rule of thumb, the larger the transition width (the thicker the ring edge), the less the diffraction present in the measured data, as is physically expected.

The thesis also introduces the actual occultation geometry. Following [1], a one-dimensional approximation of the geometry together with the Fresnel-Huygens integral is used to derive from first principles an expression for the Fresnel scale using the stationary phase approximation. The expression demonstrates the dependency of the Fresnel scale on various occultation geometry parameters, such as, wavelength, ring opening angle, distance of spacecraft from mean ring plane, and angle of the ring intercept point with the line-of-sight (the line-of-sight connects the Earth to the spacecraft). A detailed derivation of the forward-Fresnel transform is also carried out, starting from the Fresnel-Huygens integral, and using the quadratic phase approximation to the stationary phase point of the phase term in the Fresnel-Huygens integral.

A geometry file reader is implemented in MATLAB to extract information regarding occultation geometry from data contained in a text ASCII file based on the actual spacecraft trajectory (provided by Dr. Essam Marouf). The ASCII file contains information such as, time signal is transmitted from Cassini, time at which signal is received on Earth, and other geometry parameters required for calculating the Fresnel scale. The Fresnel scale is then calculated from these geometry files and used in the inverse-Fresnel transform for reconstruction of ring profiles.

As Cassini moves along its trajectory, the characteristic Fresnel scale which is dependent on the occultation geometry varies. But, a constant Fresnel scale is utilized in the inverse-Fresnel filter for reconstructing the ring profile. Therefore, an error always exists between the actual Fresnel scale and the Fresnel scale used in the inverse-Fresnel filter.

Following the approach outlined in [1], a detailed derivation of the reconstruction filter impulse response, when an error exists in the knowledge of the Fresnel scale, is derived from first principles. It is shown how this error limits radial resolution and creates distortion in the reconstructed data, in the form of severe oscillations and overshoots. Hence, it is important to control this error by updating the Fresnel scale in order to minimize the distortions. The thesis implements two different but equally effective algorithms for updating the Fresnel scale of the inverse-Fresnel filter. One method applies to block of measured data for which the characteristic variation of the Fresnel scale follows an ascending or descending pattern. It uses a recursive binary search algorithm for updating of Fresnel scale, which is a fast algorithm. The other method is “adaptive” and makes no assumptions about the manner of variation of the Fresnel scale with radial distance. The user inputs some initial block length. The algorithm merges as many blocks as possible such that the offset in the Fresnel scale of the new block does not exceed the maximum allowable value. If the error in any block exceeds the maximum allowable value, the user is asked to decrease the block length. Both these algorithms have been applied to actual Cassini geometry data (Rev7) to show their correctness in updating the Fresnel scale. A reconstruction algorithm that uses the Fresnel updating methods is implemented using MATLAB and is applied to simulated diffraction-limited data, with variable Fresnel scale, for reconstruction. It is shown that the algorithm suppresses diffraction-like oscillations introduced into the reconstructed ring profile due to variation in the Fresnel scale with ring radius.

Many detailed computational simulations of diffraction-limited data for idealistic ring profiles (forward-Fresnel transform) and their reconstruction (inverse-Fresnel transform) were presented throughout the thesis. The many examples and simulations demonstrate how the Inverse-Fresnel filter implemented in MATLAB being applied to the measured data, while updating the Fresnel scale, almost eliminates diffraction effects and reconstructs the original ring profile, revealing many of the hidden ring features and characteristics.

Further study can be carried out, in order to reduce the computational complexity for very large data blocks while improving the radial resolution. Because of time limitations, study of the effect of some of the constraints such as, system phase stability, additive thermal noise, and phase approximations remained out of the scope of this thesis.

This thesis has used the quadratic phase approximation for deriving the Forward-Fresnel transform, using the stationary phase method. Even though the quadratic phase approximation, results in almost diffraction-less reconstructed ring profiles, the approximation is valid only under certain conditions [1]. A good case for further studies would be to use higher order phase approximations to handle a more broad range of observational geometry and to implement the inverse-Fresnel filter in MATLAB (or C++) with these higher order phase approximations included.

References

- [1] E. A. Marouf, G. L. Tyler, and P. A. Rosen, "Profiling Saturn's Rings by Radio Occultation," *ICARUS* 68, pp. 120-166, 1986.
- [2] E. A. Marouf, Lecture notes for Computed Imaging (EE 264, San Jose State University), 2004.
- [3] S. K. Mitra, "Digital Signal Processing, a Computer-Based Approach," *McGraw-Hill*, 1998.
- [4] T. J. Harries, Lecture handouts for Optics (PHY2208, University of Exeter), [Online document], [cited 2006 Apr 12], Available at:
<http://newton.ex.ac.uk/teaching/resources/th/phy2208/lect13.pdf>
- [5] MATLAB Central File Exchange Website, [cited 2006 Jun 05], Available at:
<http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=6580&objectType=file>
- [6] "1-D Spectra via Method of Stationary Phase," [Online document], [cited 2006 Aug 22], Available at:
http://www.cis.rit.edu/class/simg738/Handouts/stationary_phase.pdf
- [7] The MathWorks Support Website, [cited 2006 Aug 23], Available at:
<http://www.mathworks.com/support/tech-notes/1600/1605.html#intro>

Appendices:
MATLAB Codes for Each Chapter

1 MATLAB Codes for Chapter 1

1.1 GapandEdgeSimulation.m

```
%-----
% GapandEdgeSimulation.m
% This program creates diffraction-limited data, That, measured by the
% spacecraft, and reconstructs the almost diffraction-less ring
% transmittance, T. The ring profile in this program consists of a gap
% of width 2.4F km and an edge distanced 18.5F km from the middle of
% the gap. The ring itself is assumed to behave like a gray screen with
% optical depth of tau.
% That: Diffraction limited image plane transmittance
% F: Fresnel scale
% dR: Desired resolution in km
% Beta : Kaiser window parameter used for windowing the filter
% T: The reconstructed object plane (ring) transmittance
%
%-----

clear all;
close all;

F = 15;                % Fresnel scale in km
dR = 2;                % Resolution in km
dxo = 0.25;            % Sampling interval
if(dR < dxo)
    error('Sampling intervals must be finer than the desired
    resolution');
end
Beta = 3.5;

%-----
% Creating data for object plane transmittance
%-----

tau = 8;                % Optical depth
Xo_start = 0;           % Starting location for object plane
                        % transmittance
Xo_stop = 100*F;
Nxo = floor((Xo_stop-Xo_start)/dxo)+1;    % Number of data points
                                           % used to create the object
                                           % plane transmittance
Xo = Xo_start*ones(1,Nxo)+[0:Nxo-1]*dxo;  % Distance sample points
if(Nxo ~= length(Xo))
    error('Error! Check Nxo and Xo');
end

% Object plane transmittance
```



```

To = ones(1,Nxo);
Xo_GapStart = Xo_start + 40*F;
Xo_GapEnd = Xo_GapStart + 2.4*F;
Xo_edge = Xo_GapStart + (Xo_GapEnd-Xo_GapStart)/2 + 18.5*F;

indx01 = find (Xo < Xo_GapStart);
To(indx01) = ones(size(indx01))*exp(-tau/2);
indx02 = find (Xo_GapEnd < Xo & Xo < Xo_edge);
To(indx02) = ones(size(indx02))*exp(-tau/2);

magTo = abs(To);
magTo dB=20*log10(magTo);
angTo = angle(To)*180/pi;

%-----
% Creating the diffraction pattern
%-----

[That, Xi_startn, dxin, Nh] = SimulateThat(To, Xo_start/F, dR/F,
dxo/F, Beta);
Xi_start = Xi_startn*F;
dxi = dxin * F;
Xi = Xi_start*ones(size(That)) + [0:length(That)-1]*dxi;

magThat = abs(That);
magThat dB=20*log10(magThat);
angThat = angle(That)*180/pi;

Xi_GapStart = Xo_GapStart;
Xi_GapEnd = Xo_GapEnd;
Xi_edge = Xo_edge;
indx1 = find (Xi_GapStart-Nh*dxi<Xi & Xi<Xi_edge+Nh*dxi );

%-----
% Reconstructing the object plane transmittance
%-----

Fmatrix = F*ones(1,length(That));
Fm = UpdateF(Fmatrix,dR);
[T, X] = ReconstructF(That,Fm, Xi_start,dR, dxi, Beta);

magT = abs(T);
magT dB=20*log10(magT);
angT = angle(T)*180/pi;

X_GapStart = Xi_GapStart;
X_GapEnd = Xi_GapEnd;
X_edge = Xi_edge;

dx = dxi;
indx = find (X_GapStart-Nh*dx<X & X<X_edge+Nh*dx );

if(Beta ~= 0)
    dR = 2*dR;

```

```

end

%-----
% Plotting the actual object plane transmittance
%-----

figure(1)
plot(Xo, magTo);
title('Actual object plane transmittance magnitude');
xlabel('distance, km');

figure(2)
plot(Xo, magTodB);
title('Actual object plane transmittance magnitude (dB)');
xlabel('distance, km');
ylabel('dB');

figure(3)
plot(Xo, angTo);
title('Actual object plane transmittance phase');
xlabel('distance, km');
ylabel('degrees');

%-----
% Plotting the image plane transmittance
%-----

figure(4)
plot(Xi(indxi), magThat(indxi));
title(sprintf('Magnitude of the image plane transmittance Xs=%f km,
resolution of %f km\nF=%d km, Beta = %1.1f', dxi, dR, F, Beta));
xlabel('distance, km');

figure(5)
plot(Xi(indxi), magThatdB(indxi));
title(sprintf('Magnitude of the image plane transmittance in dB Xs=%f
km, resolution of %f km\nF=%d km, Beta = %1.1f', dxi, dR, F, Beta));
xlabel('distance, km');
ylabel('dB');

figure(6)
plot(Xi(indxi), angThat(indxi));
title(sprintf('Phase of the image plane transmittance Xs=%f km,
resolution of %f km\nF=%d km, Beta = %1.1f', dxi, dR, F, Beta));
xlabel('distance, km');
ylabel('degrees');

%-----
% Plotting the reconstructed object plane transmittance
%-----

figure(7)
plot(X(indx), magT(indx));

```

```

title(sprintf('Magnitude of the reconstructed object plane
transmittance Xs=%f km, resolution of %f km\nF=%d km, Beta =
%1.1f',dx,dR,F,Beta))
xlabel('distance,km');

figure(8)
plot(X(indx), magTdB(indx));
title(sprintf('Magnitude of the reconstructed object plane
transmittance in dB Xs=%f km, resolution of %f km\nF=%d km, Beta =
%1.1f',dx,dR,F,Beta))
xlabel('distance,km');
ylabel('dB');

figure(9)
plot(X(indx), angT(indx));
title(sprintf('Phase of the reconstructed object plane transmittance
Xs=%f km, resolution of %f km\nF=%d km, Beta = %1.1f',dx,dR,F,Beta))
xlabel('distance,km');
ylabel('degrees');

%-----
% Plotting all three on the same plot
%-----
indx0 = find (Xo_GapStart-Nhf*dxo<Xo & Xo<Xo_edge+Nhf*dxo );
figure(10)
plot(Xo(indxo),magTo(indxo),'k',Xi(indxi),magThat(indxi),'r',X(indx),magT(indx),'b');
xlabel('distance,km');
ylabel('magnitude');
title(sprintf('F = %d km, Xs = %f km, Beta = %1.1f, tau = %1.1f, R = %f
km',F,dx, Beta, tau, dR));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');

figure(11)
plot(Xo(indxo),magTodB(indxo),'k',Xi(indxi),magThatdB(indxi),'r',X(indx),magTdB(indx),'b');
xlabel('distance,km');
ylabel('magnitude (dB)');
title(sprintf('F = %d km, Xs = %f km, Beta = %1.1f, tau = %1.1f, R = %f
km',F,dx, Beta, tau, dR));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');

figure(12)
plot(Xo(indxo),angTo(indxo),'k',Xi(indxi),angThat(indxi),'r',X(indx),angT(indx),'b');
xlabel('distance,km');
ylabel('degrees');
title(sprintf('F = %d km, Xs = %f km, Beta = %1.1f, tau = %1.1f, R = %f
km',F,dx, Beta, tau, dR));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');

```

1.2 GrayGapSimulation.m

```
%-----
% GrayGapSimulation.m
% This program creates diffraction-limited data, That, measured by the
% spacecraft, and reconstructs the almost diffraction-less ring
% transmittance, T. The ring profile in this program consists of a gap
% of width 21.1F km. The ring itself is assumed to behave like a gray
% screen with optical depth of tau.
% That: Diffraction limited image plane transmittance
% F: Fresnel scale
% dR: Desired resolution in km
% Beta : Kaiser window parameter used for windowing the filter
% T: The reconstructed object plane (ring) transmittance
% All elements with subscripts of n have been normalized to F.
%-----

clear all;
close all;

F = 15;                % Fresnel scale in km
dRn = 2/15;            % Normalized resolution
dxon = 0.1/15;         % Normalized sampling interval
if(dRn < dxon)
    error('Sampling intervals must be finer than the desired
    resolution');
end
Beta = 3.5;

%-----
% Creating data for object plane transmittance
%-----

tau = 8;                % Optical depth
Xo_startn = 0;          % Starting location for object plane
                        % transmittance
Xo_stopn = 120.1;
Nxo = floor((Xo_stopn-Xo_startn)/dxon)+1;
Xon = Xo_startn*ones(1,Nxo)+[0:Nxo-1]*dxon;
if(Nxo ~= length(Xon))
    error('Error! Check Nxo and Xon');
end

% Object plane transmittance
To = ones(1,Nxo);
Xo_edgeStartn = Xo_startn+0.5*(Xo_stopn-Xo_startn)-0.5*21.1;
Xo_edgeStopn = Xo_edgeStartn + 21.1;

indxol = find (Xon < Xo_edgeStartn);
```

```

To(indxo1) = ones(size(indxo1))*exp(-tau/2);
indxo2 = find (Xon> Xo_edgeStopn);
To(indxo2) = ones(size(indxo2))*exp(-tau/2);

magTo = abs(To);
magTodB=20*log10(magTo);
angTo = angle(To)*180/pi;

%-----
% Creating the diffraction pattern
%-----

[That, Xi_startn, dxin, Nh] = SimulateThat(To, Xo_startn,dRn, dxon,
Beta);
Xin = Xi_startn*ones(size(That)) + [0:length(That)-1]*dxin;

magThat = abs(That);
magThatdB=20*log10(magThat);
angThat = angle(That)*180/pi;

Xi_edgeStartn = Xo_edgeStartn;
Xi_edgeStopn = Xo_edgeStopn;
indxi = find (Xi_edgeStartn-Nhf*dxin<Xin & Xin<Xi_edgeStopn+Nhf*dxin );

%-----
% Reconstructing the object plane transmittance
%-----

Fmatrix = F*ones(1,length(That));
Fm = UpdateF(Fmatrix,dRn*F);
[T, X] = ReconstructF(That, Fm, Xi_startn*F, dRn*F, dxin*F, Beta);

magT = abs(T);
magTdB=20*log10(magT);
angT = angle(T)*180/pi;

X_edgeStart = Xi_edgeStartn*F;
X_edgeStop = Xi_edgeStopn*F;
dx = dxin*F;
indx = find (X_edgeStart-Nhf*dx<X & X<X_edgeStop+Nhf*dx );

if(Beta ~= 0)
    dRn = 2*dRn;
end

%-----
% Plotting the actual object plane transmittance
%-----

figure(1)
plot(Xon*F, magTo);
title('Actual object plane transmittance magnitude');
xlabel('distance, km');

```

```

figure(2)
plot(Xon*F, magTodB);
title('Actual object plane transmittance magnitude (dB)');
xlabel('distance, km');
ylabel('dB');

figure(3)
plot(Xon*F, angTo);
title('Actual object plane transmittance phase');
xlabel('distance, km');
ylabel('degrees');

%-----
% Plotting the image plane transmittance
%-----

figure(4)
plot(Xin(indxi)*F, magThat(indxi));
title(sprintf('Magnitude of the image plane transmittance, sampling
interval Xs=%g km \nF = %g km, resolution of %g km, Beta =
%1.1f',dxin*F, F,dRn*F,Beta));
xlabel('distance, km');

figure(5)
plot(Xin(indxi)*F, magThatdB(indxi));
title(sprintf('Magnitude of the image plane transmittance in dB,
sampling interval Xs=%g km \nF = %g km, resolution of %g km, Beta =
%1.1f',dxin*F, F,dRn*F,Beta));
xlabel('distance, km');
ylabel('dB');

figure(6)
plot(Xin(indxi)*F, angThat(indxi));
title(sprintf('Phase of the image plane transmittance, sampling
interval Xs=%g km \nF = %g km, resolution of %g km, Beta =
%1.1f',dxin*F, F,dRn*F,Beta));
xlabel('distance, km');
ylabel('degrees');

%-----
% Plotting the reconstructed object plane transmittance
%-----

figure(7)
plot(X(indx), magT(indx));
title(sprintf('Magnitude of the reconstructed object plane
transmittance, sampling interval Xs=%g km , \nF = %g km, resolution of
%g km, Beta = %1.1f',dx, F,dRn*F,Beta));
xlabel('distance, km');
figure(8)
plot(X(indx), magTdB(indx));
title(sprintf('Magnitude of the reconstructed object plane
transmittance in dB, sampling interval Xs=%g km , \nF = %g km,
resolution of %g km, Beta = %1.1f',dx, F,dRn*F,Beta));

```

```

xlabel('distance, km');
ylabel('dB');

figure(9)
plot(X(indx), angT(indx));
title(sprintf('Phase of the reconstructed object plane transmittance,
sampling interval Xs=%g km, \nF = %g km, resolution of %g km, Beta =
%1.1f',dx, F,dRn*F,Beta))
xlabel('distance, km');
ylabel('degrees');

%-----
%Plotting all three on the same plot
%-----

dxn = dxin;
indx0 = find(Xo_edgeStartn-Nhf*dxn<Xon & Xon<Xo_edgeStopn+Nhf*dxon);
figure(10)
plot(Xon(indxo)*F,magTo(indxo),'k',Xin(indxi)*F,magThat(indxi),'r',X(indx),magT(indx),'b');
xlabel('distance, km');
ylabel('magnitude');
title(sprintf('Xs = %g km, Beta = %1.1f, tau = %1.1f, dR = %g km, F =
%g km, ',dxn*F, Beta, tau, dRn*F, F));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');

figure(11)
plot(Xon(indxo)*F,magTodB(indxo),'k',Xin(indxi)*F,magThatdB(indxi),'r',X(indx),magTdB(indx),'b');
xlabel('distance, km');
ylabel('magnitude (dB)');
title(sprintf('Xs = %g km, Beta = %1.1f, tau = %1.1f, dR = %g km, F =
%g km, ',dxn*F, Beta, tau, dRn*F, F));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');

figure(12)
plot(Xon(indxo)*F,angTo(indxo),'k',Xin(indxi)*F,angThat(indxi),'r',X(indx),angT(indx),'b');
xlabel('distance, km');
ylabel('degrees');
title(sprintf('Xs = %g km, Beta = %1.1f, tau = %1.1f, dR = %g km, F =
%g km, ',dxn*F, Beta, tau, dRn*F, F));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');

```

2 MATLAB Codes for Chapter 2

2.1 Fresnel.m

```
function f=Fresnel(x);
% Fresnel.m
% Fresnel function Creates C(x)+jS(x)
f=mfun('FresnelC',x)+j*mfun('FresnelS',x);
```

2.2 CornuSpiral.m

```
% CornuSpiral.m
% Plots the Fresnel integral in the real and imaginary axis (Cornu
spiral)
clear all;
close all;

w = [-20:.02:20];
C = mfun('FresnelC',w);
S = mfun('FresnelS',w);
plot(S,C);
axis equal
hold on

w1 = [-3:3];
C1 = mfun('FresnelC',w1);
S1 = mfun('FresnelS',w1);
plot(S1,C1,'ro');

grid
hold off
xlabel('C(w)')
ylabel('S(w)')
title('Cornu Spiral')
```

2.3 InverseFresnelFilter.m

```
% InverseFresnelFilter.m
% Inverse-Fresnel filter analysis

clear all;
close all;

dRn = .02; % Normalized resolution dR/F, dR = resolution
Wn = 2/dRn; % Normalized filter Length Wn = W/F
% W = 2F^2/dR
K = .01; % Normalized Sampling intervals = KdRn
```



```

X = [-0.5*Wn:K*dRn:0.5*Wn];           % K*dRn = Normalized spatial
                                       % sampling intervals
N = length(X);           % Number of filter sample points
hi = ((1+j)/2)*exp((-j*0.5*pi*(X).^2)); % Inverse-Fresnel filter

%-----
% Analyzing hi when no window is used (rectangular window)
%-----

Nfft = 2^(nextpow2(N)); % Using 2^n Fourier points for faster
                        % processing
wk = ([-Nfft/2:(Nfft/2)-1]/Nfft)*2*pi; % Frequency bins
H = K*dRn*fft(hi,Nfft); % Multiplied by a factor to convert magnitude
                        % to continuous Fourier transform
H = H.*exp(j*wk*(N-1)/2); % Compensating for the shift in
                        % hi
H = fftshift(H);

figure(1)
plot(wk/(2*pi),abs(H));
grid
xlabel('U/Us');
ylabel('|H|');
title(sprintf('Frequency response of the inverse-Fresnel filter using a
rectangular window and \n normalized sampling interval Xs=%g,
normalized resolution of %g\n',K*dRn,dRn));

figure(2)
plot(wk/(2*pi),20*log10(abs(H)));
grid
xlabel('U/Us');
ylabel('|H|dB');
title(sprintf('Frequency response of the inverse-Fresnel filter using a
rectangular window and \n normalized sampling interval Xs=%g,
normalized resolution of %g\n',K*dRn,dRn));

figure(3)
plot(wk/(2*pi),angle(H)*180/pi);
grid
xlabel('U/Us');
ylabel('Angle, degrees');
title(sprintf('Phase response of the inverse-Fresnel filter using a
rectangular window and \n normalized sampling interval Xs=%g,
normalized resolution of %g\n',K*dRn,dRn));

%-----
% Analyzing hi when kaiser window is used
%-----

Beta = 2.5;
Kaiswin = kaiser(N,Beta);
hi = hi.*Kaiswin;

```

```

H1 = K*dRn*fft(hi,Nfft);
H1 = H1.*exp(j*wk*(N-1)/2);
H1 = fftshift(H1);
figure(4)
plot(wk/(2*pi),abs(H1));
grid
xlabel('U/Us');
ylabel('|H|');
title(sprintf('Frequency response of the inverse-Fresnel filter using a
Kaiser window of Beta = %1.1f\n normalized sampling interval Xs=%g,
normalized resolution of %g\n',Beta,K*dRn,2*dRn));

figure(5)
plot(wk/(2*pi),20*log10(abs(H1)));
grid
xlabel('U/Us');
ylabel('|H|dB');
title(sprintf('Frequency response of the inverse-Fresnel filter using a
kaiser window of Beta = %1.1f\n normalized sampling interval Xs=%g,
normalized resolution of %g\n',Beta,K*dRn,2*dRn));

figure(6)
plot(wk/(2*pi),angle(H1)*180/pi);
grid
xlabel('U/Us');
ylabel('Angle, degrees');
title(sprintf('Phase response of the inverse-Fresnel filter using a
kaiser window of Beta = %1.1f\n normalized sampling interval Xs=%g,
normalized resolution of %g\n',Beta,K*dRn,2*dRn));

```

2.4 SimulateThat.m

```

function [That, Xi_startn, dxin, Nhf] = SimulateThat(To, Xo_startn,
dRn, dxon, Beta)
%-----
% SimulateThat.m
% That: Image plane transmittance
% To: Object plane transmittance
% Xo_startn: Normalized starting location of object plane transmittance
% dRn: Normalized resolution
% dxon: Normalized sampling intervals in the object plane
% Beta: Kaiser window parameter, used for creating the forward fresnel
% filter
% That: Diffraction-limited image plane transmittance
% Xi_startn: Normalized starting location of image plane transmittance
% dxin: Normalized sampling intervals in the image plane
% Nhf: Nhf+1 is the number of forward-Fresnel filter data points
%
%
%-----
%-----

```

```

% Defining the parameters
%-----

No = length(To)-1          % No+1 is length of To
dR_Thatn = dRn;           % Normalized resolution
dxin = dxon;              % Sampling intervals are the same in both
                           % planes
L_hfn = 2/dR_Thatn;       % Normalized Length of filter to achieve
                           % dR_That, L_hf = 2*F^2/dR_That
Nhfn = ceil(L_hfn/dxin)    % N_hf+1 is the numer of filter points
if(Nhfn >= No)
    error('Length of forward Fresnel filter must be smaller than the
    length of the object plane transmittance.\n Either increase the
    object plane transmittance data points or reduce the resolution');
end
Xhfn = [-Nhfn/2:Nhfn/2]*dxin; % Normalized sample points
                           % of function hf
hf = ((1-j)/(2))*exp((j*pi*(Xhfn).^2)/2); % Forward-Fresnel filter
Kaiswin = kaiser(length(Xhfn),Beta);      % Kaiser window
hf = hf.*Kaiswin; % Filter windowed with a Kaiser window
S = abs(sum(hf));
hf = hf/S; % Normalizing hf so that it reconstructs a
           % constant level exactly

%-----
% Using fftfilt method for filtering
%-----

That = fftfilt(hf,To);

% Uses the overlap-add method, always uses
% power 2 fft points. It always chooses blocks
% in such a way that it minimizes
% overall fft calculations.
That = That(length(hf):length(To)); % Only keeps the
% reconstructed samples which have had enough
% data points for reconstruction
Xi_startn = Xo_startn+Nhfn*dxin/2;
Nhfn = length(hf)-1;

```

2.5 Reconstruct.m

```

function [T, X_startn, dxn, Nhi] = Reconstruct(That, Xi_startn, dRn,
dxin, Beta)
%-----
%
% This function performs a reconstruction algorithm on the
% diffraction-limited data That and returns T.
% That: Diffraction limited image plane transmittance
% Xi_startn: Normalized starting location of That data
% dRn: Normalized resolution (km) (dRn = dR/F)
% Beta: If greater than zero, it is the Kaiser window parameter, if it
% is zero then a rectangular window is used for reconstruction.

```

```

% T: The reconstructed object plane transmittance
% X_startn: Normalized starting location of T data
% Nhi: Nhi+1 is the number of inverse-Fresnel filter data points
% dxn: Normalized sampling intervals in the object plane
% dxin: Normalized sampling intervals in the image plane
%
%-----

Ni = length(That)-1      % Ni+1 samples in the image plane are
                        % considered for reconstructing the object
                        % plane transmittance
dxn = dxin;              % Normalized sampling distance = dx/F
Leffn = 2/dRn;           % Normalized effective filter length
                        % Leff = (2*F^2)/dR
Nhi = ceil(Leffn/dxn)    % Nhi+1 is length of the filter
if(Nhi >= Ni)
    error('Length of inverse Fresnel filter must be smaller that the
    length of the image plane transmittance.\n Either increase the
    image plane transmittance data points or reduce the resolution');
end

%-----
% Defining the reconstruction filter
%-----

X = [-Nhi/2:Nhi/2]*dxn;      % Range of function hi
hi = ((1+j)/(2))*exp((-j*pi*(X).^2)/2); % Inverse-Fresnel filter
if(Beta>0)
    Kaiswin = kaiser(Nhi+1,Beta); % Kaiser window
    hi = hi.*Kaiswin';          % Windowing the filter
end
S = abs(sum(hi));
hi = hi/S;                  % Normalizing hi so that it reconstructs a
                        % constant level exactly

%-----
% Using add_overlap method to calculate the object plane transmittance
%-----

tic
T = fftfilt(hi,That);
                        % Uses the overlap-add method, always uses
                        % power 2 fft points. It always chooses blocks
                        % to minimize overall fft calculations.
t_FFT = toc              % Time consumed by fftfilt

T = T(length(hi):length(That)); % Only considering points that have had
                        % enough data points for reconstruction
X_startn = Xi_startn+Nhi*dxn/2;
Nhi = length(hi)-1;

```

2.6 Peter Volegov's fcs() function

FCS Computes Fresnel integrals.

[C, S] = fcs(x) returns Fresnel integrals C and S for argument x,
x must be double and real.

F = fcs(x) returns complex F = C+i*S

Algorithm:

This function uses an improved method for computing Fresnel integrals
with an error of less than 1×10^{-9} , described in:

Klaus D. Mielenz, Computation of Fresnel Integrals. II

J. Res. Natl. Inst. Stand. Technol. 105, 589 (2000), pp 589-590

Copyright (c) 2002 by Peter L. Volegov (volegov@unm.edu)

All Rights Reserved.

2.7 Commands used for compiling the fcs mex file in MATLAB

```
>> mex -setup
```

Please choose your compiler for building external interface (MEX)
files:

Would you like mex to locate installed compilers [y]/n? y

Select a compiler:

[1] Borland C/C++ version 5.02 in C:\BC5

[2] Digital Visual Fortran version 6.0 in C:\Program Files\Microsoft
Visual Studio

[3] Lcc C version 2.4 in C:\MATLAB6P5\sys\lcc

[4] Microsoft Visual C/C++ version 7.0 in C:\Program Files\Microsoft
Visual Studio .NET

[5] Microsoft Visual C/C++ version 6.0 in C:\Program Files\Microsoft
Visual Studio

[0] None

Compiler: 4

Please verify your choices:

Compiler: Microsoft Visual C/C++ 7.0

Location: C:\Program Files\Microsoft Visual Studio .NET

Are these correct?([y]/n): y

The default options file:

"C:\Documents and Settings\Owner\Application

Data\MathWorks\MATLAB\R13\mexopts.bat"

is being updated from C:\MATLAB6P5\BIN\WIN32\mexopts\msvc70opts.bat...

```
>> mex -O fcs.c -DWIN32
```

2.8 ImpulseResponse.m

```
%-----
% To = delta(x-x_middle), the object plane transmittance is a delta
% function.
% The diffracted transmittance(That) in the image plane is given to the
% reconstruction algorithm and the object plane transmittance is
% recovered.(T)
%
%-----

clear all;
close all;

F = 5;                % Fresnel scale in km
dRn = .2;             % Normalized Resolution
dxon = .001;          % Normalized Sampling intervals
if(dRn < dxon)
    error('Sampling intervals must be finer than the desired
    resolution');
end
Beta = 0;

%-----
% Creating the diffraction-limited pattern
%-----
dxin = dxon;          % Equal sampling intervals in the object and
                      % image planes
Ln = (2)/dRn;          % Normalized length of filter used
Nhf = ceil(Ln/dxin);   % Nhf+1 is the number of sample points in the
                      % forward-Fresnel filter
X_middlen = 0;
Xi_startn = -Nhf*dxin;
Xin = Xi_startn*ones(1,2*Nhf+1) + [0:2*Nhf]*dxin; % Normalized distance
                      % samples
That = ((1-j)/(2*F))*exp(j*0.5*pi*((Xin-X_middlen).^2)); % Image
                      % plane transmittance
magThat = abs(That);   % Magnitude of That
magThatdB=20*log10(magThat); % Magnitude of That in dB scale
angThat = angle(That)*180/pi; % Phase of That in degrees

%-----
% Reconstructing the object plane transmittance
%-----
% T : Reconstructed transmittance
% X_startn : Normalized distance at which T starts
% dxn : Sampling intervals in the object plane
% Nhi : Number of inverse-Fresnel filter samples used for
% reconstruction
```

```

[T, X_startn, dxn, Nhi] = Reconstruct(That, Xi_startn, dRn, dxin,
Beta);
Tm = max(abs(T));
T=T/Tm; % Normalizing T to 1.
Xn = X_startn*ones(size(T)) + [0:length(T)-1]*dxn;
magT = abs(T);
magTdB=20*log10(magT);
angT = angle(T)*180/pi;

%-----
% Theoretical response
% -----

Xthn = Xn;
% Theoretical response, normalized to one
Tth = dRn*((Nhi*dxn)/2)*sinc(((Nhi*dxn)/2)*(Xthn-
X_middlen)).*exp(j*pi*0.5*((Xthn-X_middlen).^2));

%-----
% Redefining resolution for kaiser window
%-----
% If a kaiser window is used, then the achieved resolution is almost
% twice the desired resolution
if(Beta ~= 0)
    dRn = 2*dRn;
end

%-----
% Plotting the image plane transmittance
%-----

figure(1)
plot(Xin, magThat);
title(sprintf('Magnitude of the image plane transmittance, Normalized
sampling interval Xs=%g , \n Normalized resolution of %g, Beta =
%1.1f', dxin, dRn, Beta))
xlabel('Normalized distance');
grid

figure(2)
plot(Xin, magThatdB);
title(sprintf('Magnitude of the image plane transmittance in
dB, Normalized sampling interval Xs=%g , \n Normalized resolution of %g,
Beta = %1.1f', dxin, dRn, Beta))
xlabel('Normalized distance');
ylabel('dB');
grid

figure(3)
plot(Xin, angThat);
title(sprintf('Phase of the image plane transmittance, Normalized
sampling interval Xs=%g , \n Normalized resolution of %g, Beta =
%1.1f', dxin, dRn, Beta))
xlabel('Normalized distance');

```

```

ylabel('Degrees');
grid

%-----
% Plotting the reconstructed object plane transmittance
%-----

figure(4)
plot(Xn, magT);
title(sprintf('Magnitude of the reconstructed object plane
transmittance, Normalized sampling interval Xs=%g , \n Normalized
resolution of %g, Beta = %1.1f',dxn,dRn,Beta))
xlabel('Normalized distance');
grid

figure(5)
plot(Xn, magTdB);
title(sprintf('Magnitude of the reconstructed object plane
transmittance in dB, Normalized sampling interval Xs=%g , \n Normalized
resolution of %g, Beta = %1.1f',dxn,dRn,Beta))
xlabel('Normalized distance');
ylabel('dB');
grid

figure(6)
plot(Xn, angT);
title(sprintf('Phase of the reconstructed object plane transmittance,
Normalized sampling interval Xs=%g , \n Normalized resolution of %g,
Beta = %1.1f',dxn,dRn,Beta))
xlabel('Normalized distance');
ylabel('Degrees');
grid

%-----
% Plotting the theoretical reconstructed object plane transmittance
%-----

magTth= abs(Tth);
magTthdB=20*log10(magTth);
angTth = angle(Tth)*180/pi;

figure(7)
plot(Xthn, magTth);
title(sprintf('Magnitude of the theoretical reconstructed object plane
transmittance, Normalized sampling interval Xs=%g , \n Normalized
resolution of %g, Beta = %1.1f',dxn,dRn,Beta))
xlabel('Normalized distance');
grid

figure(8)
plot(Xthn, magTthdB);
title(sprintf('Magnitude of the theoreticalreconstructed object plane
transmittance in dB , Normalized sampling interval Xs=%g , \n
Normalized resolution of %g, Beta = %1.1f',dxn,dRn,Beta))

```



```

xlabel('Normalized distance');
ylabel('dB');
grid

```

```

figure(9)
plot(Xthn, angTth);
title(sprintf('Phase of the theoretical reconstructed object plane
transmittance , Normalized sampling interval Xs=%g , \n Normalized
resolution of %g, Beta = %1.1f',dxn,dRn,Beta))
xlabel('Normalized distance');
ylabel('Degrees');
grid

```

```

%-----
% Plotting the reconstructed and theoretical on the same plot
%-----

```

```

figure(10)
plot(Xn,magT,'k',Xn,magTth,'r--');
xlabel('Normalized distance');
ylabel('Magnitude');
title(sprintf('Comparison of the reconstructed and the theoretical
impulse response magnitude\nNormalized sampling interval Xs=%g , \n
Normalized resolution of %g, Beta = %1.1f',dxn,dRn,Beta));
legend('Reconstructed impulse response','Theoretical impulse
response');
grid

```

```

figure(11)
plot(Xn,magTdB,'k',Xn,magTthdB,'r--');
xlabel('Normalized distance');
ylabel('Magnitude (dB)');
title(sprintf('Comparison of the reconstructed and the theoretical
impulse response magnitude (dB)\nNormalized sampling interval Xs=%g ,
\n Normalized resolution of %g, Beta = %1.1f',dxn,dRn,Beta));
legend('Reconstructed impulse response','Theoretical impulse
response');
grid

```

```

figure(12)
plot(Xn,angT,'k',Xn,angTth,'r--');
xlabel('Normalized distance');
ylabel('Degrees');
title(sprintf('Comparison of the reconstructed and the theoretical
impulse response phase\nNormalized sampling interval Xs=%g , \n
Normalized resolution of %g, Beta = %1.1f',dxn,dRn,Beta));
legend('Reconstructed impulse response','Theoretical impulse
response');
grid

```

2.9 VerifySimulateThat.m

```
%-----
% VerifySimulateThat.m
% Code for comparing implementation of the forward-Fresnel transform
% using the FFT algorithm versus the theoretical analysis.
%-----

clear all;
close all;

dRn = .05;           % Normalized Resolution
dxon = .01;          % Normalized sampling intervals
if(dRn < dxon)
    error('Sampling intervals must be finer than the desired
    resolution');
end
Beta = 3.5;

%-----
% Object plane transmittance (Gray edge: gray region before Xo_edgen,
% free-space region after Xo_edgen)
%-----

tau = 8;             % Optical depth
Xo_startn = -100;    % Normalized starting distance of object plane
                    % transmittance
Xo_stopn = 100;
Nx0 = floor((Xo_stopn-Xo_startn)/dxon)+1; % Number of object plane
                    % sample points
Xon = Xo_startn*ones(1,Nx0)+[0:Nx0-1]*dxon; % Normalized distance
                    % samples
if(Nx0 ~= length(Xon))
    error('Error! Check Nx0 and Xon');
end
To = ones(1,Nx0);    % Create an array on ones first (all free
                    % space)
Xo_edgen = Xo_startn + 0.5 * (Xo_stopn - Xo_startn); % Decide where
                    % the edge is
                    % located at
indx0 = find (Xon < Xo_edgen); % Find locations
                    % corresponding to the gray
                    % region.
To(indx0) = ones(size(indx0))*exp(-tau/2); % Force the object plane to
                    % match the characteristics of a gray edge in
                    % the gray region.

magTo = abs(To);      % Calculate magnitude of To
magTo dB = 20*log10(magTo); % Magnitude in dB
angTo = angle(To)*180/pi; % Phase in degrees

%-----
% Creating the diffraction pattern using fft method
```

```

%-----

% Create the diffraction pattern
[That, Xi_startn, dxin, Nh] = SimulateThat(To, Xo_startn, dRn, dxon,
Beta);

% Available normalized distance samples
Xin = Xi_startn*ones(size(That)) + [0:length(That)-1]*dxin;

magThat = abs(That); % Calculate magnitude of
% That
magThatdB=20*log10(magThat); % Magnitude in dB
angThat = angle(That)*180/pi; % Phase in degrees

Xi_edgen = Xo_edgen; % The edge in the image plane is located at the
% same place as the image plane.

% Limit the data we will view for efficiency
indxi = find (Xi_edgen-Nhf*dxin<Xin & Xin<Xi_edgen+Nhf*dxin );

%-----
% Creating the diffraction pattern using the theoretical response
%-----
% Theoretical diffraction pattern
That_expected = 0.5*(1-j)*((exp(-tau/2)+1)*(0.5+0.5*j) + (-exp(-
tau/2)+1)*fcs(Xin(indxi)));

magThat_expected = abs(That_expected);
magThatdB_expected=20*log10(magThat_expected);
angThat_expected = angle(That_expected)*180/pi;

%-----
% Comparison plots
%-----

if (Beta > 0) % In this case, achieved resolution will be
    dRn = 2 * dRn; % almost twice as worse as the desired
end % resolution

figure(1)
plot(Xin(indxi), magThat(indxi), 'k', Xin(indxi), magThat_expected, 'r-
-');
title(sprintf('Magnitude of the image plane transmittance obtained by
the FFT method vs theoretical response\n Normalized sampling interval
Xs=%g, Normalized resolution of %g, Beta = %1.1f',dxin,dRn,Beta));
xlabel('Normalized distance');
ylabel('Magnitude');
legend('FFT algorithm','Theoretical response');

figure(2)
plot(Xin(indxi), magThatdB(indxi), 'k', Xin(indxi), magThatdB_expected,
'r--');
title(sprintf('Magnitude of the image plane transmittance in dB
obtained by the FFT method vs theoretical response\n Normalized

```

```

sampling interval Xs=%g, Normalized resolution of %g, Beta =
%1.1f',dxin,dRn,Beta));
xlabel('Normalized distance');
ylabel('dB');
legend('FFT algorithm','Theoretical response');

figure(3)
plot(Xin(indxi), angThat(indxi), 'k', Xin(indxi), angThat_expected, 'r-
-');
title(sprintf('Phase of the image plane transmittance obtained by the
FFT method vs theoretical response\n Normalized sampling interval
Xs=%g, Normalized resolution of %g, Beta = %1.1f',dxin,dRn,Beta));
xlabel('Normalized distance');
ylabel('Degrees');
legend('FFT algorithm','Theoretical response');

figure(4)
d = That(indxi) - That_expected;
plot(Xin(indxi),abs(d));
title(sprintf('Difference of the image plane transmittance magnitude
obtained by the FFT method and theoretical response\n Normalized
sampling interval Xs=%g, Normalized resolution of %g, Beta =
%1.1f',dxin,dRn,Beta));
xlabel('Normalized distance');

```

2.10 GrayEdgeSimulation.m

```

%-----
% GrayEdgeSimulation.m
% This program generates the reconstructed object plane transmittance
% of a gray edge.
%
%-----

clear all;
close all;

dRn =0.05;           % Normalized resolution
dxon = .005;         % Normalized sampling intervals
if(dRn < dxon)
    error('Sampling intervals must be finer than the desired
    resolution');
end
Beta = 3.5;

%-----
% Creating the object plane transmittance of a gray edge
%-----

tau = 8;              % Optical depth
Xo_startn = -100;     % Normalized starting distance of object plane
                    % transmittance

```

```

Xo_stopn = 100;
Nxo = floor((Xo_stopn-Xo_startn)/dxon)+1;    % Number of object plane
                                           % sample points
Xon = Xo_startn*ones(1,Nxo)+[0:Nxo-1]*dxon;
if(Nxo ~= length(Xon))
    error('Error! Check Nxo and Xon');
end
To = ones(1,Nxo);
Xo_edgen = Xo_startn + 0.5 * (Xo_stopn - Xo_startn);
indx0 = find (Xon < Xo_edgen);
To(indx0) = ones(size(indx0))*exp(-tau/2);    % Object plane
                                           % transmittance(gray edge)

magTo = abs(To);
magTodB=20*log10(magTo);
angTo = angle(To)*180/pi;

%-----
% Creating the diffraction pattern
%-----

[That, Xi_startn, dxin, Nh] = SimulateThat(To, Xo_startn, dRn, dxon,
Beta);
Xin = Xi_startn*ones(size(That)) + [0:length(That)-1]*dxin;

magThat = abs(That);
magThatdB=20*log10(magThat);
angThat = angle(That)*180/pi;

Xi_edgen = Xo_edgen;
indx1 = find (Xi_edgen-Nhf*dxin<Xin & Xin<Xi_edgen+Nhf*dxin );

%-----
% Reconstructing the object plane transmittance
%-----

[T, X_startn, dxn, Nhi] = Reconstruct(That, Xi_startn, dRn, dxin,
Beta);

% Available normalized distance samples
Xn = X_startn*ones(size(T)) + [0:length(T)-1]*dxn;

magT = abs(T);
magTdB=20*log10(magT);
angT = angle(T)*180/pi;

X_edgen = Xi_edgen;

% Limit the data we will view for efficiency
indx = find (X_edgen-Nhi*dxn<Xn & Xn<X_edgen+Nhi*dxn );

if(Beta ~= 0)
    dRn = 2*dRn;
end

```

```

%-----
% Plotting the actual object plane transmittance
%-----

figure(1)
plot(Xon, magTo);
title('Actual object plane transmittance magnitude');
xlabel('Normalized distance');
grid

figure(2)
plot(Xon, magTodB);
title('Actual object plane transmittance magnitude (dB)');
xlabel('Normalized distance');
ylabel('dB');
grid

figure(3)
plot(Xon, angTo);
title('Actual object plane transmittance phase');
xlabel('Normalized distance');
ylabel('degrees');
grid

%-----
% Plotting the image plane transmittance
%-----

figure(4)
plot(Xin(indxi), magThat(indxi));
title(sprintf('Magnitude of the image plane transmittance, Normalized
sampling interval Xs=%g \n Normalized resolution of %g, Beta =
%1.1f', dxin, dRn, Beta));
xlabel('Normalized distance');
grid

figure(5)
plot(Xin(indxi), magThatdB(indxi));
title(sprintf('Magnitude of the image plane transmittance in dB,
Normalized sampling interval Xs=%g , \n Normalized resolution of %g,
Beta = %1.1f', dxin, dRn, Beta));
xlabel('Normalized distance');
ylabel('dB');
grid

figure(6)
plot(Xin(indxi), angThat(indxi));
title(sprintf('Phase of the image plane transmittance, Normalized
sampling interval Xs=%g , \n Normalized resolution of %g, Beta =
%1.1f', dxin, dRn, Beta));
xlabel('Normalized distance');
ylabel('degrees');
grid

```

```

%-----
% Plotting the reconstructed object plane transmittance
%-----

figure(7)
plot(Xn(indx), magT(indx));
title(sprintf('Magnitude of the reconstructed object plane
transmittance, Normalized sampling interval Xs=%g , \n Normalized
resolution of %g, Beta = %1.1f',dxn,dRn,Beta))
xlabel('Normalized distance');
grid

figure(8)
plot(Xn(indx), magTdB(indx));
title(sprintf('Magnitude of the reconstructed object plane
transmittance in dB, Normalized sampling interval Xs=%g , \n
Normalized resolution of %g, Beta = %1.1f',dxn,dRn,Beta))
xlabel('Normalized distance');
ylabel('dB');
grid

figure(9)
plot(Xn(indx), angT(indx));
title(sprintf('Phase of the reconstructed object plane transmittance,
Normalized sampling interval Xs=%g , \n Normalized resolution of %g,
Beta = %1.1f',dxn,dRn,Beta))
xlabel('Normalized distance');
ylabel('degrees');
grid

%-----
% Plotting all three on the same plot
%-----
% Limit data in object plane to the same locations as image plan
indx0 = find(Xo_edgen-Nhi*dxn<Xon & Xon<Xo_edgen+Nhi*dxon);

figure(10)
plot(Xon(indxo),magTo(indxo),'k',Xin(indxi),magThat(indxi),'r',Xn(indx),
magT(indx),'b');
xlabel('Normalized distance');
ylabel('magnitude');
title(sprintf('Xs = %g, Beta = %1.1f, tau = %1.1f, dRn = %g ',dxn,
Beta, tau, dRn));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');
grid

figure(11)
plot(Xon(indxo),magTodB(indxo),'k',Xin(indxi),magThatdB(indxi),'r',Xn(i
ndx),magTdB(indx),'b');
xlabel('Normalized distance');
ylabel('magnitude (dB)');

```

```

title(sprintf('Xs = %g, Beta = %1.1f, tau = %1.1f, dRn = %g ',dxn,
Beta, tau, dRn));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');
grid

figure(12)
plot(Xon(indxo),angTo(indxo),'k',Xin(indxi),angThat(indxi),'r',Xn(indx)
,angT(indx),'b');
xlabel('Normalized distance');
ylabel('degrees');
title(sprintf('Xs = %g, Beta = %1.1f, tau = %1.1f, dRn = %g ',dxn,
Beta, tau, dRn));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');
grid

```

2.11 FilteringArtifact.m

```

%-----
% FilteringArtifact.m
% Displays artifact caused by filtering in the reconstructed data.
%
%-----

clear all;
close all;

dRn = 0.05;           % Normalized resolution
dxon = .005;          % Normalized sampling intervals
if(dRn < dxon)
    error('Sampling intervals must be finer than the desired
    resolution');
end
Beta = 3.5;

%-----
% Creating the object plane transmittance of a gray edge
%-----

tau = 8;               % Optical depth
Xo_startn = -100;      % Normalized starting distance of object plane
                    % transmittance
Xo_stopn = 100;
Nxo = floor((Xo_stopn-Xo_startn)/dxon)+1; % Number of object plane
                    % sample points
Xon = Xo_startn*ones(1,Nxo)+[0:Nxo-1]*dxon;
if(Nxo ~= length(Xon))
    error('Error! Check Nxo and Xon');
end
To = ones(1,Nxo);
Xo_edgen = Xo_startn + 0.5 * (Xo_stopn - Xo_startn);

```



```

indx0 = find (Xon < Xo_edgen);
To(indxo) = ones(size(indxo))*exp(-tau/2); % Object plane
                                           % transmittance(gray edge)

magTo = abs(To);
magTodB=20*log10(magTo);
angTo = angle(To)*180/pi;

%-----
% Creating the diffraction pattern
%-----

[That, Xi_startn, dxin, Nh] = SimulateThat(To, Xo_startn, dRn, dxon,
Beta);
Xin = Xi_startn*ones(size(That)) + [0:length(That)-1]*dxin;

magThat = abs(That);
magThatdB=20*log10(magThat);
angThat = angle(That)*180/pi;

Xi_edgen = Xo_edgen;

%-----
% Actual image plane transmittance (theoretical)
%-----

That_th = 0.5*(1-j)*((exp(-tau/2)+1)*(0.5+0.5*j) + (-exp(-
tau/2)+1)*fcs(Xin - Xi_edgen));
magThat_th = abs(That_th);
magThatdB_th=20*log10(magThat_th);
angThat_th = angle(That_th)*180/pi;

%-----
% Reconstructing the object plane transmittance from simulated data
%-----

[T, X_startn, dxn, Nh] = Reconstruct(That, Xi_startn, dRn, dxin,
Beta);

% Available normalized distance samples
Xn = X_startn*ones(size(T)) + [0:length(T)-1]*dxn;

magT = abs(T);
magTdB=20*log10(magT);
angT = angle(T)*180/pi;

X_edgen = Xi_edgen;

%-----
% Reconstructing the object plane transmittance from theoretical data
%-----

[T_th, X_startnth, dxn_th, Nh_th] = Reconstruct(That_th, Xi_startn,
dRn, dxin, Beta);

```

```

% Available normalized distance samples
Xn_th = X_startnth*ones(size(T_th)) + [0:length(T_th)-1]*dxn_th;

magT_th = abs(T_th);
magTdB_th=20*log10(magT_th);
angT_th = angle(T_th)*180/pi;

%-----
% Plotting the reconstructed data from theoretical and simulated data
% on top of each other
%-----
if(Beta ~= 0)
    dRn = 2*dRn;
end
figure(1)
plot(Xn,magT,'k',Xn_th,magT_th,'r');
xlabel('Normalized distance');
ylabel('Magnitude');
title(sprintf('Reconstructed profile for Xs = %g, Beta = %1.1f, tau = %1.1f, dRn = %g ',dxn, Beta, tau, dRn));
legend('From simulated data','From theoretical data');
grid

figure(2)
plot(Xn,magTdB,'k',Xn_th,magTdB_th,'r');
xlabel('Normalized distance');
ylabel('Magnitude (dB)');
title(sprintf('Reconstructed profile for Xs = %g, Beta = %1.1f, tau = %1.1f, dRn = %g ',dxn, Beta, tau, dRn));
legend('From simulated data','From theoretical data');
grid

figure(3)
plot(Xn,angT,'k',Xn_th,angT_th,'r');
xlabel('Normalized distance');
ylabel('Degrees');
title(sprintf('Reconstructed profile for Xs = %g, Beta = %1.1f, tau = %1.1f, dRn = %g ',dxn, Beta, tau, dRn));
legend('From simulated data','From theoretical data');
grid

%-----
% Plotting the diffracted data from theoretical and simulated data on
% top of each other
%-----
figure(4)
plot(Xin,magThatdB_th,'r',Xin,magThatdB,'k');
xlabel('Normalized distance');
ylabel('Magnitude (dB)');
title(sprintf('Comparison of theoretical and simulated diffraction-
limited data for an edge\nXs = %g, Beta = %1.1f, tau = %1.1f, dRn = %g ',dxn, Beta, tau, dRn));
legend('Theoretical', 'Simulated');

```

grid

3 MATLAB Codes for Chapter 3

3.1 LinearlyTaperedEdgeSimulation.m

```
%-----  
% LinearlyTaperedEdgeSimulation.m  
% This program loads the diffraction-limited data That of a linearly  
% tapered edge, and reconstructs the almost diffractionless data T.  
% That: Diffraction limited image plane transmittance  
% dRn: Normalized desired resolution  
% Beta : Kaiser window parameter used for windowing the filter  
% T: The reconstructed object plane transmittance  
%-----  
  
clear all;  
close all;  
  
dRn = .05;           % Normalized resolution  
dxon = 0.005;        % Normalized sampling intervals  
if(dRn < dxon)  
    error('Sampling intervals must be finer than the desired  
    resolution');  
end  
Beta = 3.5;  
  
%-----  
% Creating object plane transmittance  
%-----  
  
tau = 8;             % Optical depth  
Xo_startn = -120;    % Normalized starting distance of object plane  
                    % transmittance  
Xo_stopn = 120;  
L_tapern = 1;         % Normalized taper width  
Nxo = floor((Xo_stopn-Xo_startn)/dxon)+1;  
Xon = Xo_startn*ones(1,Nxo)+[0:Nxo-1]*dxon;  
if(Nxo ~= length(Xon))  
    error('Error! Check Nxo and Xon');  
end  
  
% Creating the object plane transmittance of a linearly tapered edge,  
% which starts to transition from ring region to free-space region at  
% Xo_edgen - L_tapern and fully enters free-space region at Xo_edgen.  
  
To = ones(1,Nxo);  
Xo_edgen = Xo_startn + 0.5 * (Xo_stopn - Xo_startn);  
Xo_TaperStartn = Xo_edgen - L_tapern;  
indxo = find (Xon < Xo_TaperStartn);
```

```

To(indxo) = ones(size(indxo))*exp(-tau/2);
indxo = find (Xo_TaperStartn <= Xon & Xon < Xo_edgen);
% Eq. (3.8)
TauTapered = -tau*ones(size(indxo)).*((Xon(indxo)-
Xo_TaperStartn*ones(size(indxo)))/(L_tapern))+tau*ones(size(indxo));
%tau(x)=-tau*(x-x_taperstart)/(x_edge-x_taperstart)+tau
To(indxo) = ones(size(indxo)).*exp(-TauTapered/2);

magTo = abs(To);
magTodB=20*log10(magTo);
angTo = angle(To)*180/pi;

%-----
% Creating the diffraction pattern
%-----

[That, Xi_startn, dxin, Nh] = SimulateThat(To, Xo_startn, dRn, dxon,
Beta);
Xin = Xi_startn*ones(size(That)) + [0:length(That)-1]*dxin;

magThat = abs(That);
magThatdB=20*log10(magThat);
angThat = angle(That)*180/pi;

Xi_edgen = Xo_edgen;
indx = find (Xi_edgen-Nh*dxin<Xin & Xin<Xi_edgen+Nh*dxin );

%-----
% Reconstructing the object plane transmittance
%-----

[T, X_startn, dxn, Nhi] = Reconstruct(That, Xi_startn, dRn, dxin,
Beta);
Xn = X_startn*ones(size(T)) + [0:length(T)-1]*dxn;

magT = abs(T);
magTdB=20*log10(magT);
angT = angle(T)*180/pi;

X_edgen = Xi_edgen;
indx = find (X_edgen-Nhi*dxn<Xn & Xn<X_edgen+Nhi*dxn );

if(Beta ~= 0)
    dRn = 2*dRn;
end

%-----
% Plotting the actual object plane transmittance
%-----

figure(1)
plot(Xon, magTo);
title('Actual object plane transmittance magnitude');
xlabel('Normalized distance');

```

```

figure(2)
plot(Xon, magTodB);
title('Actual object plane transmittance magnitude (dB)');
xlabel('Normalized distance');
ylabel('dB');

figure(3)
plot(Xon, angTo);
title('Actual object plane transmittance phase');
xlabel('Normalized distance');
ylabel('Degrees');

%-----
% Plotting the image plane transmittance
%-----

figure(4)
plot(Xin(indxi), magThat(indxi));
title(sprintf('Magnitude of the image plane transmittance, Normalized
sampling interval Xs=%g \n Normalized resolution of %g, Beta = %1.1f,
Normalized taper length = %g',dxin,dRn,Beta,L_tapern))
xlabel('Normalized distance');

figure(5)
plot(Xin(indxi), magThatdB(indxi));
title(sprintf('Magnitude of the image plane transmittance in dB,
Normalized sampling interval Xs=%g , \n Normalized resolution of %g,
Beta = %1.1f, Normalized taper length = %g',dxin,dRn,Beta,L_tapern))
xlabel('Normalized distance');
ylabel('dB');

figure(6)
plot(Xin(indxi), angThat(indxi));
title(sprintf('Phase of the image plane transmittance, Normalized
sampling interval Xs=%g , \n Normalized resolution of %g, Beta = %1.1f,
Normalized taper length = %g',dxin,dRn,Beta,L_tapern))
xlabel('Normalized distance');
ylabel('Degrees');
%-----
% Plotting the reconstructed object plane transmittance
%-----

figure(7)
plot(Xn(indx), magT(indx));
title(sprintf('Magnitude of the reconstructed object plane
transmittance, Normalized sampling interval Xs=%g , \n Normalized
resolution of %g, Beta = %1.1f, Normalized taper length =
%g',dxn,dRn,Beta,L_tapern))
xlabel('Normalized distance');

figure(8)
plot(Xn(indx), magTdB(indx));

```

```

title(sprintf('Magnitude of the reconstructed object plane
transmittance in dB, Normalized sampling interval Xs=%g , \n
Normalized resolution of %g, Beta = %1.1f, Normalized taper length =
%g',dxn,dRn,Beta,L_tapern))
xlabel('Normalized distance');
ylabel('dB');

figure(9)
plot(Xn(indx), angT(indx));
title(sprintf('Phase of the reconstructed object plane transmittance,
Normalized sampling interval Xs=%g , \n Normalized resolution of %g,
Beta = %1.1f, Normalized taper length = %g',dxn,dRn,Beta,L_tapern))
xlabel('Normalized distance');
ylabel('Degrees');

%-----
% Plotting all three on the same plot
%-----

indx0 = find(Xo_edgen-Nhi*dxn<Xon & Xon<Xo_edgen+Nhi*dxn);
figure(10)
plot(Xon(indxo),magTo(indxo),'k',Xin(indxi),magThat(indxi),'r',Xn(indx)
,magT(indx),'b');
xlabel('Normalized distance');
ylabel('Magnitude');
title(sprintf('Xs = %g, Beta = %1.1f, tau = %1.1f, dRn = %g ',dxn,
Beta, tau, dRn));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');

figure(11)
plot(Xon(indxo),magTodB(indxo),'k',Xin(indxi),magThatdB(indxi),'r',Xn(i
ndx),magTdB(indx),'b');
xlabel('Normalized distance');
ylabel('Magnitude (dB)');
title(sprintf('Xs = %g, Beta = %1.1f, tau = %1.1f, dRn = %g ',dxn,
Beta, tau, dRn));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');

figure(12)
plot(Xon(indxo),angTo(indxo),'k',Xin(indxi),angThat(indxi),'r',Xn(indx)
,angT(indx),'b');
xlabel('Normalized distance');
ylabel('Degrees');
title(sprintf('Xs = %g, Beta = %1.1f, tau = %1.1f, dRn = %g ',dxn,
Beta, tau, dRn));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');

```

3.2 LinearlyTaperedEdgeSimulationComparison.m

```
%-----
% LinearlyTaperedEdgeSimulationComparison.m
% This program loads the diffraction-limited data That of a linearly
% tapered edge, and reconstructs the almost diffractionless data T.
% That: Diffraction limited image plane transmittance
% dRn: Normalized desired resolution
% Beta : Kaiser window parameter used for windowing the filter
% T: The reconstructed object plane transmittance
%-----

clear all;
close all;

dRn = .05;           % Normalized resolution
dxon = 0.005;        % Normalized sampling intervals
if(dRn < dxon)
    error('Sampling intervals must be finer than the desired
    resolution');
end
Beta = 3.5;

%-----
% Creating object plane transmittance
%-----

tau = 8;             % Optical depth
Xo_startn = -100;    % Normalized starting distance of object plane
                    % transmittance
Xo_stopn = 100;
Nxo = floor((Xo_stopn-Xo_startn)/dxon)+1;
Xon = Xo_startn*ones(1,Nxo)+[0:Nxo-1]*dxon;
if(Nxo ~= length(Xon))
    error('Error! Check Nxo and Xon');
end

% Creating four object plane transmittances for linearly tapered edges,
% which start to transition from ring region to free-space region at
% Xo_edgen - TaperLengthn and fully enter free-space region at
% Xo_edgen.
% Four different taper lengths, TaperLengthn, are considered for
% analysis.
To = ones(1,Nxo);
Xo_edgen = Xo_startn + 0.5 * (Xo_stopn - Xo_startn);
TaperLengthn = [0 1 2 3];
Xo_TaperStartn = Xo_edgen - TaperLengthn;

for k=1:length(TaperLengthn)
    indxo = find (Xon < Xo_TaperStartn(k));
    To(indxo) = ones(size(indxo))*exp(-tau/2);
    indxo = find (Xo_TaperStartn(k) <= Xon & Xon < Xo_edgen);
    TauTapered = -tau*ones(size(indxo)).*((Xon(indxo) -
```

```

Xo_TaperStartn(k)*ones(size(indxo)))/(Xo_edgen-
Xo_TaperStartn(k))+tau*ones(size(indxo)); %tau(x)=-tau*(x-
x_taperstart)/(x_edgen-x_taperstart)+tau
To(indxo) = ones(size(indxo)).*exp(-TauTapered/2);

magTo(k,:) = abs(To);
magTodB(k,:) = 20*log10(magTo(k,:));
angTo(k,:) = angle(To)*180/pi;

%-----
% Creating the diffraction pattern associated with each object
% plane transmittance
%-----
[That(k,:), Xi_startn, dxin, Nhif] = SimulateThat(To, Xo_startn,
dRn, dxon, Beta);
Xin = Xi_startn*ones(size(That(k,:))) + [0:length(That(k,:))-
1]*dxin;

magThat(k,:) = abs(That(k,:));
magThatdB(k,:) = 20*log10(magThat(k,:));
angThat(k,:) = angle(That(k,:))*180/pi;
end

Xi_edgen = Xo_edgen;
indxi = find (Xi_edgen-Nhf*dxin<Xin & Xin<Xi_edgen+Nhf*dxin );

%-----
% Reconstructing the object plane transmittance
%-----

for k=1:length(TaperLengthn)
    [T(k,:), X_startn, dxn, Nhif] = Reconstruct(That(k,:), Xi_startn,
dRn, dxin, Beta);
    Xn = X_startn*ones(size(T(k,:))) + [0:length(T)-1]*dxn;

    magT(k,:) = abs(T(k,:));
    magTdB(k,:) = 20*log10(magT(k,:));
    angT(k,:) = angle(T(k,:))*180/pi;
end
X_edgen = Xi_edgen;
indx = find (X_edgen-Nhi*dxn<Xn & Xn<X_edgen+Nhi*dxn );

if(Beta ~= 0)
    dRn = 2*dRn;
end

%-----
% Plotting the actual object plane transmittance
%-----

figure(1)
plot(Xon, magTo(1,:), 'r', Xon, magTo(2,:), 'b', Xon,
magTo(3,:), 'k', Xon, magTo(4,:), 'g');

```



```

title('Actual object plane transmittance magnitude');
xlabel('Normalized distance');
legend(sprintf('L=%f',TaperLengthn(1)),sprintf('L=%f',TaperLengthn(2)),
sprintf('L=%f',TaperLengthn(3)),sprintf('L=%f',TaperLengthn(4)));

figure(2)
plot(Xon, magTodB(1,:), 'r', Xon, magTodB(2,:), 'b', Xon,
magTodB(3,:), 'k', Xon, magTodB(4,:), 'g');
title('Actual object plane transmittance magnitude (dB)');
xlabel('Normalized distance');
ylabel('dB');
legend(sprintf('L=%f',TaperLengthn(1)),sprintf('L=%f',TaperLengthn(2)),
sprintf('L=%f',TaperLengthn(3)),sprintf('L=%f',TaperLengthn(4)));

figure(3)
plot(Xon, angTo(1,:), 'r', Xon,
angTo(2,:), 'b', Xon, angTo(3,:), 'k', Xon, angTo(4,:), 'g');
title('Actual object plane transmittance phase');
xlabel('Normalized distance');
ylabel('Degrees');
legend(sprintf('L=%f',TaperLengthn(1)),sprintf('L=%f',TaperLengthn(2)),
sprintf('L=%f',TaperLengthn(3)),sprintf('L=%f',TaperLengthn(4)));

%-----
% Plotting the image plane transmittance
%-----

figure(4)
plot(Xin(indxi), magThat(1,indxi), 'r', Xin(indxi),
magThat(2,indxi), 'b', Xin(indxi), magThat(3,indxi), 'k', Xin(indxi),
magThat(4,indxi), 'g');
title(sprintf('Magnitude of the image plane transmittance for tapered
edges of different L.\nNormalized sampling interval Xs=%g, normalized
resolution of %g, Beta = %1.1f',dxin,dRn,Beta));
xlabel('Normalized distance');
legend(sprintf('L=%f',TaperLengthn(1)),sprintf('L=%f',TaperLengthn(2)),
sprintf('L=%f',TaperLengthn(3)),sprintf('L=%f',TaperLengthn(4)));

figure(5)
plot(Xin(indxi), magThatdB(1,indxi), 'r', Xin(indxi),
magThatdB(2,indxi), 'b', Xin(indxi), magThatdB(3,indxi), 'k', Xin(indxi),
magThatdB(4,indxi), 'g');
title(sprintf('Magnitude of the image plane transmittance in dB for
tapered edges of different L.\nNormalized sampling interval Xs=%g,
normalized resolution of %g, Beta = %1.1f',dxin,dRn,Beta));
xlabel('Normalized distance');
ylabel('dB');
legend(sprintf('L=%f',TaperLengthn(1)),sprintf('L=%f',TaperLengthn(2)),
sprintf('L=%f',TaperLengthn(3)),sprintf('L=%f',TaperLengthn(4)));

figure(6)
plot(Xin(indxi), angThat(1,indxi), 'r', Xin(indxi),
angThat(2,indxi), 'b', Xin(indxi), angThat(3,indxi), 'k', Xin(indxi),
angThat(4,indxi), 'g');

```

```

title(sprintf('Phase of the image plane transmittance for tapered edges
of different L.\nNormalized sampling interval Xs=%g, normalized
resolution of %g, Beta = %1.1f',dxin,dRn,Beta))
xlabel('Normalized distance');
ylabel('Degrees');
legend(sprintf('L=%f',TaperLengthn(1)),sprintf('L=%f',TaperLengthn(2)),
sprintf('L=%f',TaperLengthn(3)),sprintf('L=%f',TaperLengthn(4)));
%-----
% Plotting the reconstructed object plane transmittance
%-----

figure(7)
plot(Xn(indx), magT(1,indx), 'r',Xn(indx),magT(2,indx), 'b',Xn(indx),
magT(3,indx), 'k',Xn(indx), magT(4,indx), 'g');
title(sprintf('Magnitude of the reconstructed object plane
transmittance for tapered edges of different L.\nNormalized sampling
interval Xs=%g, normalized resolution of %g, Beta =
%1.1f',dxn,dRn,Beta))
xlabel('Normalized distance');
legend(sprintf('L=%f',TaperLengthn(1)),sprintf('L=%f',TaperLengthn(2)),
sprintf('L=%f',TaperLengthn(3)),sprintf('L=%f',TaperLengthn(4)));

figure(8)
plot(Xn(indx), magTdB(1,indx), 'r',Xn(indx),
magTdB(2,indx), 'b',Xn(indx),magTdB(3,indx), 'k',Xn(indx),
magTdB(4,indx), 'g');
title(sprintf('Magnitude of the reconstructed object plane
transmittance in dB for tapered edges of different L.\nNormalized
sampling interval Xs=%g, normalized resolution of %g, Beta =
%1.1f',dxn,dRn,Beta))
xlabel('Normalized distance');
ylabel('dB');
legend(sprintf('L=%f',TaperLengthn(1)),sprintf('L=%f',TaperLengthn(2)),
sprintf('L=%f',TaperLengthn(3)),sprintf('L=%f',TaperLengthn(4)));

figure(9)
plot(Xn(indx), angT(1,indx), 'r',Xn(indx),angT(2,indx), 'b',Xn(indx),
angT(3,indx), 'k',Xn(indx), angT(4,indx), 'g');
title(sprintf('Phase of the reconstructed object plane transmittance
for tapered edges of different L.\nNormalized sampling interval Xs=%g,
normalized resolution of %g, Beta = %1.1f',dxn,dRn,Beta))
xlabel('Normalized distance');
ylabel('Degrees');
legend(sprintf('L=%f',TaperLengthn(1)),sprintf('L=%f',TaperLengthn(2)),
sprintf('L=%f',TaperLengthn(3)),sprintf('L=%f',TaperLengthn(4)));

%-----
%Plotting all three on the same plot
%-----

indx0 = find(Xo_edgen-Nhi*dxn<Xon & Xon<Xo_edgen+Nhi*dxon);
figure(10)
plot(Xon(indxo),magTo(1,indx0), 'k',Xin(indxi),magThat(1,indx0), 'r',Xn(i
ndx),magT(1,indx), 'b');

```

```

hold
plot(Xon(indxo),1+magTo(2,indxo),'k',Xin(indxi),1+magThat(2,indxi),'r',
Xn(indx),1+magT(2,indx),'b');
plot(Xon(indxo),2+magTo(3,indxo),'k',Xin(indxi),2+magThat(3,indxi),'r',
Xn(indx),2+magT(3,indx),'b');
plot(Xon(indxo),3+magTo(4,indxo),'k',Xin(indxi),3+magThat(4,indxi),'r',
Xn(indx),3+magT(4,indx),'b');
xlabel('Normalized distance');
ylabel('Magnitude');
title(sprintf('Xs = %f , Beta = %1.1f, tau = %1.1f, dRn = %f ',dxn,
Beta, tau, dRn));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');
hold off;

```

```

figure(11)
plot(Xon(indxo),magTodB(1,indxo),'k',Xin(indxi),magThatdB(1,indxi),'r',
Xn(indx),magTdB(1,indx),'b');
hold
plot(Xon(indxo),40+magTodB(2,indxo),'k',Xin(indxi),40+magThatdB(2,indxi),
'r',Xn(indx),40+magTdB(2,indx),'b');
plot(Xon(indxo),80+magTodB(3,indxo),'k',Xin(indxi),80+magThatdB(3,indxi),
'r',Xn(indx),80+magTdB(3,indx),'b');
plot(Xon(indxo),120+magTodB(4,indxo),'k',Xin(indxi),120+magThatdB(4,indxi),
'r',Xn(indx),120+magTdB(4,indx),'b');
xlabel('Normalized distance');
ylabel('Magnitude (dB)');
title(sprintf('Xs = %f , Beta = %1.1f, tau = %1.1f, dRn = %f ',dxn,
Beta, tau, dRn));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');
hold off;

```

```

figure(12)
plot(Xon(indxo),angTo(1,indxo),'k',Xin(indxi),angThat(1,indxi),'r',Xn(i
ndx),angT(1,indx),'b');
hold
plot(Xon(indxo),360+angTo(2,indxo),'k',Xin(indxi),360+angThat(2,indxi),
'r',Xn(indx),360+angT(2,indx),'b');
plot(Xon(indxo),2*360+angTo(3,indxo),'k',Xin(indxi),2*360+angThat(3,indxi),
'r',Xn(indx),2*360+angT(3,indx),'b');
plot(Xon(indxo),3*360+angTo(4,indxo),'k',Xin(indxi),3*360+angThat(4,indxi),
'r',Xn(indx),3*360+angT(4,indx),'b');
xlabel('Normalized distance');
ylabel('Degrees');
title(sprintf('Xs = %f, Beta = %1.1f, tau = %1.1f, dRn = %f',dxn, Beta,
tau, dRn));
legend('Actual object plane transmittance','Image plane
transmittance','Reconstructed object plane transmittance');
hold off;

```

4 MATLAB Codes for Chapter 4

4.1 Code for verifying the Fresnel scale formula

-VERIFYING EQ. (4.41)

```
>> syms B p0 p phi phi0 D X
>> syms K
>> z = cos(B)*(p0*cos(phi0)-p*cos(phi))/D;
>> n = (p0^2+p^2-2*p*p0*cos(phi-phi0))/D^2;
>> ang = K*D*(sqrt(1+2*z+n)-(1+z));
>> df = diff(ang, phi)
```

df =

```
K*D*(1/2/(1+2*cos(B)*(p0*cos(phi0)-p*cos(phi))/D+(p0^2+p^2-2*p*p0*cos(phi-phi0))/D^2)^(1/2)*(2*cos(B)*p*sin(phi)/D+2*p*p0*sin(phi-phi0)/D^2)-cos(B)*p*sin(phi)/D)
```

```
>> pretty(df)
```

$$K D \left[\frac{1}{2} \sqrt{\frac{\cos(B) p \sin(\phi)}{D} + 2 \frac{p p_0 \sin(\phi - \phi_0)}{D^2}} + \frac{\cos(B) (p_0 \cos(\phi_0) - p \cos(\phi))}{2 D} + \frac{p_0^2 + p^2 - 2 p p_0 \cos(\phi - \phi_0)}{D^2} \right]^{1/2} - \frac{\cos(B) p \sin(\phi)}{D}$$

-VERIFYING EQ. (4.42)

```
>> df2 = diff(ang,phi,2)
```

df2 =

```
K*D*(-1/4/(1+2*cos(B)*(p0*cos(phi0)-p*cos(phi))/D+(p0^2+p^2-2*p*p0*cos(phi-phi0))/D^2)^(3/2)*(2*cos(B)*p*sin(phi)/D+2*p*p0*sin(phi-phi0)/D^2)^2+1/2/(1+2*cos(B)*(p0*cos(phi0)-p*cos(phi))/D+(p0^2+p^2-2*p*p0*cos(phi-phi0))/D^2)^(1/2)*(2*cos(B)/D*p*cos(phi)+2*p*p0*cos(phi-phi0)/D^2)-cos(B)/D*p*cos(phi))
```

```
>> pretty(df2)
```

$$K D \left[-\frac{1}{4} \sqrt{\frac{\cos(B) p \sin(\phi)}{D} + 2 \frac{p p_0 \sin(\phi - \phi_0)}{D^2}}^2 + \frac{\cos(B) (p_0 \cos(\phi_0) - p \cos(\phi))}{2 D} \right]$$

$$\begin{aligned}
& D \\
& + \frac{p_0^2 + p^2 - 2 p p_0 \cos(\phi - \phi_0)}{D^2} \sqrt[3]{2} + \frac{1}{2} \sqrt{\quad} \\
& 2 \frac{\cos(B) p \cos(\phi)}{D} + 2 \frac{p p_0 \cos(\phi - \phi_0)}{D^2} \sqrt{\quad} \sqrt{\quad} \sqrt{1} \\
& + 2 \frac{\cos(B) (p_0 \cos(\phi_0) - p \cos(\phi))}{D} \\
& + \frac{p_0^2 + p^2 - 2 p p_0 \cos(\phi - \phi_0)}{D^2} \sqrt[3]{2} - \frac{\cos(B) p \cos(\phi)}{D} \sqrt{\quad}
\end{aligned}$$

-VERIFYING EQ. (4.52)

```
>> syms K D z0 n0 B phi0 p0 X
>> z0 = -cos(B)*cos(phi0)*X;
>> n0 = X^2;
>> ang0 = K*D*(sqrt(1+2*z0+n0)-(1+z0));
>> chi0 = taylor(ang0, 2, X)
```

chi0 =

0

```
>> chi0 = taylor(ang0, 3, X)
```

chi0 =

$K \cdot D \cdot (1/2 - 1/2 \cos(B)^2 \cos(\phi_0)^2) \cdot X^2$

```
>> pretty(chi0)
```

$$K D \left(\frac{1}{2} - \frac{1}{2} \cos(B)^2 \cos(\phi_0)^2 \right) X^2$$

-VERIFYING EQ. (4.53) to the second order term

```
>> dang0 = K*D*((X + p0/D)*cos(B)*sin(phi0))*(1+2*z0+n0)^(-1/2) - (X +
p0/D)*cos(B)*sin(phi0));
>> dchi0 = taylor(dang0, 3, X)
```

dchi0 =

$K \cdot p_0 \cdot \cos(B)^2 \cdot \sin(\phi_0) \cdot \cos(\phi_0) \cdot X + K \cdot D \cdot (p_0/D \cdot \cos(B) \cdot \sin(\phi_0) \cdot (-1/2 + 3/2 \cos(B)^2 \cos(\phi_0)^2) + \cos(B)^2 \sin(\phi_0) \cos(\phi_0)) \cdot X^2$

```
>> pretty(dchi0)
```

$$\frac{K p_0 \cos(B)^2 \sin(\phi_0) \cos(\phi_0) X + K D \left(\frac{p_0 \cos(B) \sin(\phi_0) (-1/2 + 3/2 \cos(B)^2 \cos(\phi_0)^2)}{D} + \cos(B)^2 \sin(\phi_0) \cos(\phi_0) \right) X^2}{1}$$

-VERIFYING EQ. (4.56)

```
>> dang02 = K*D*((X+p0/D)*cos(B)*cos(phi0)+(X*p0/D+(p0/D)^2))*(1+2*z0+n0)^(-1/2)-((X+p0/D)*cos(B)*sin(phi0))^2*(1+2*z0+n0)^(-3/2)-(X+p0/D)*cos(B)*cos(phi0);
>> dchi02 = taylor(dang02, 3, X )
```

dchi02 =

```
K*D*(p0^2/D^2-
p0^2/D^2*cos(B)^2*sin(phi0)^2)+K*D*((p0/D*cos(B)*cos(phi0)+p0^2/D^2)*cos(B)*cos(phi0)+p0/D-3*p0^2/D^2*cos(B)^3*sin(phi0)^2*cos(phi0)-2*p0/D*cos(B)^2*sin(phi0)^2)*X+K*D*((p0/D*cos(B)*cos(phi0)+p0^2/D^2)*(-1/2+3/2*cos(B)^2*cos(phi0)^2)+(cos(B)*cos(phi0)+p0/D)*cos(B)*cos(phi0)-p0^2/D^2*cos(B)^2*sin(phi0)^2*(-3/2+15/2*cos(B)^2*cos(phi0)^2)-cos(B)^2*sin(phi0)^2-6*p0/D*cos(B)^3*sin(phi0)^2*cos(phi0))*X^2
```

```
>> pretty(dchi02)
```

$$K D \left(\frac{p_0^2}{D^2} - \frac{p_0^2 \cos(B)^2 \sin(\phi_0)^2}{D^2} \right) + K D \left(\frac{p_0 \cos(B) \cos(\phi_0)}{D} + \frac{p_0^2}{D^2} \cos(B) \cos(\phi_0) + \frac{p_0}{D} - 3 \frac{p_0^2 \cos(B)^3 \sin(\phi_0)^2 \cos(\phi_0)}{D^2} - 2 \frac{p_0 \cos(B)^2 \sin(\phi_0)^2 \cos(\phi_0)}{D} \right) X + K \left(\frac{p_0 \cos(B) \cos(\phi_0)}{D} + \frac{p_0^2}{D^2} (-1/2 + 3/2 \cos(B)^2 \cos(\phi_0)^2) + \cos(B) \cos(\phi_0) + \frac{p_0}{D} \right)$$

$$\frac{p_0^2 \cos^2(B) \sin^2(\phi_0) (-3/2 + 15/2 \cos^2(B) \cos^2(\phi_0))}{D^2} - \cos^2(B) \sin^2(\phi_0) - 6 \frac{p_0^3 \cos^3(B) \sin^2(\phi_0) \cos(\phi_0)}{D^2} \Bigg| X^2$$

-VERIFYING EQ. (4.59)

```
>> L = (dang0)^2/dang02;
>> taylor(L, 3, X)
```

ans =

```
K/D*p0^2*cos(B)^4*sin(phi0)^2*cos(phi0)^2/(p0^2/D^2-
p0^2/D^2*cos(B)^2*sin(phi0)^2)*X^2
```

```
>> pretty(ans)
```

$$\frac{K p_0^2 \cos^4(B) \sin^2(\phi_0) \cos^2(\phi_0) X^2}{D^2 \frac{p_0^2}{D^2} - \frac{p_0^2 \cos^2(B) \sin^2(\phi_0)}{D^2}}$$

- VERIFYING EQ.(4.60), only the second order term is used

```
>> angQ = ang0 - (1/2)*((dang0)^2)/(dang02);
>> taylor(angQ, 4, X)
```

ans =

```
(K*D*(1/2-1/2*cos(B)^2*cos(phi0)^2) -
1/2*K/D*p0^2*cos(B)^4*sin(phi0)^2*cos(phi0)^2/(p0^2/D^2-
p0^2/D^2*cos(B)^2*sin(phi0)^2))*X^2+(K*D*(1/2*cos(B)*cos(phi0)-
1/2*cos(B)^3*cos(phi0)^3) -
1/2*(2*K*p0*cos(B)^2*sin(phi0)*cos(phi0)*(p0/D*cos(B)*sin(phi0)*(-
1/2+3/2*cos(B)^2*cos(phi0)^2)+cos(B)^2*sin(phi0)*cos(phi0))+K*cos(B)^4*sin(phi0)
)^2*cos(phi0)^2*D/(-
1+cos(B)^2*sin(phi0)^2)*((p0/D*cos(B)*cos(phi0)+p0^2/D^2)*cos(B)*cos(phi0)+p0/D
-3*p0^2/D^2*cos(B)^3*sin(phi0)^2*cos(phi0)-
2*p0/D*cos(B)^2*sin(phi0)^2))/(p0^2/D^2-p0^2/D^2*cos(B)^2*sin(phi0)^2))*X^3
```

```
>> pretty(ans)
```

$$\frac{K D (1/2 - 1/2 \cos^2(B) \cos^2(\phi_0))}{\dots}$$

$$\begin{aligned}
& - \frac{1}{2} \frac{K p_0^2 \cos^4(B) \sin^2(\phi_0) \cos^2(\phi_0)}{D \left(\frac{p_0^2}{2} - \frac{p_0^2 \cos^2(B) \sin^2(\phi_0)}{D} \right)} X^2 + \frac{K D \left(\frac{1}{2} \cos(B) \cos(\phi_0) - \frac{1}{2} \cos^3(B) \cos^3(\phi_0) \right) - \frac{1}{2} \left(2 K p_0 \cos^2(B) \sin(\phi_0) \cos(\phi_0) \right)}{D \left(\frac{p_0^2}{2} - \frac{p_0^2 \cos^2(B) \sin^2(\phi_0)}{D} \right)} \\
& + \frac{p_0 \cos(B) \sin(\phi_0) \left(-\frac{1}{2} + \frac{3}{2} \cos^2(B) \cos^2(\phi_0) \right)}{D} \\
& + \cos^2(B) \sin(\phi_0) \cos(\phi_0) \left(+ K \cos^4(B) \sin^2(\phi_0) \cos^2(\phi_0) D \right) \\
& \left(\frac{p_0 \cos(B) \cos(\phi_0)}{D} + \frac{p_0^2}{2D} \right) \cos(B) \cos(\phi_0) + \frac{p_0}{D} \\
& - 3 \frac{p_0^2 \cos^3(B) \sin^2(\phi_0) \cos^2(\phi_0)}{D^2} - 2 \frac{p_0 \cos^2(B) \sin^2(\phi_0)}{D} \\
& - 1 + \cos^2(B) \sin^2(\phi_0) \left(\frac{p_0^2}{2D} - \frac{p_0^2 \cos^2(B) \sin^2(\phi_0)}{D^2} \right) X^3
\end{aligned}$$

4.2 Example of contents of a geometry file

This example is taken from "Rev7, May 3-3, 2005 (DOY 122-123)", radio occultation file.

First column: SCET (Spacecraft Event Time, past midnight, secs)

Second column: ERT (Earth Receive time, Greenwich mean time, secs)

Third column: Radius of intercept point (km)

Fourth column: Impact Parameter

Fifth column: Angle φ_0 relative to an internal reference (degrees)

Sixth column: Angle φ_0 with respect to projected Earth direction (degrees)

Seventh column: Difference between above two angles

Eighth column: Ring opening angle (B)

Ninth column: Distance from spacecraft to intercept point (D)

4318.6367	9000.0000	202077.7805	201991.6113	253.8931	88.2049
4319.6365	9001.0000	202069.9303	201983.8013	253.8935	88.2053
4320.6364	9002.0000	202062.0794	201975.9905	253.8939	88.2057
4321.6362	9003.0000	202054.2278	201968.1789	253.8943	88.2061
4322.6361	9004.0000	202046.3755	201960.3666	253.8947	88.2065
4323.6359	9005.0000	202038.5224	201952.5536	253.8951	88.2069
4324.6358	9006.0000	202030.6686	201944.7399	253.8955	88.2073
4325.6357	9007.0000	202022.8140	201936.9254	253.8959	88.2077

165.6882	-23.5770	87120.6154
165.6882	-23.5770	87135.1713
165.6882	-23.5770	87149.7268
165.6882	-23.5770	87164.2820
165.6882	-23.5770	87178.8370
165.6882	-23.5770	87193.3916
165.6882	-23.5770	87207.9458
165.6882	-23.5770	87222.4998

4.3 GeometryFileReader.m

```
% -----  
% This program reads geometric parameters from a text file  
% -----  
clear all; close all;  
format long e;  
fid = fopen('C:\Documents and  
Settings\Owner\Desktop\Rev7Ing_Geometry.txt');  
  
% Create a 14 row matrix by reading to the end of file of  
% Rev7Ing_Geometry.txt  
A=fscanf(fid,'%g %g %g %g %g %g %g %g %g %g %g %g %g %g', [14 inf]);  
B=A';  
% Convert to column vectors, to look like the  
% text file  
SCET = B(:,1); % Spacecraft event time (past midnight, secs)  
ERT = B(:,2); % Earth receive time (Greenwich mean time,  
% secs)  
p0 = B(:,3); % Radius of intercept point (km)  
IP = B(:,4); % Impact parameter (km)  
phi0_Rel = B(:,5); % Angle phi0 relative to an inertial reference  
% (degrees)  
phi0 = B(:,6); % Angle phi0 with respect to projected Earth  
% direction (degrees)  
PhiDiff = B(:,7); % Difference between phi0_Rel and Phi0  
% (degrees)  
Betta = abs(B(:,8)); % Ring opening angle (degrees)
```

```

D = B(:,9);          % Distance from spacecraft to intercept point
                      % (km)
F = B(:,10);         % Fresnel scale (km), this should be calculated
                      % from geometry file. This has already been
                      % calculated and inserted in the sample
                      % geometry file being used.
dFdt = B(:,11);      % dF/dt (km/Sec), must be calculated
dp0dt = B(:,12);     % dp0/dt (km/sec), must be calculated
dp0dt2 = B(:,13);    % dp0dt/dt (km/sec^2), must be calculated
d = B(:,14);         % d^2(dp0dt)/dt^2 , must be calculated
fclose(fid);
%-----
% Double checking the Fresnel scale by calculating and checking
% wavelength
%-----
F_calc = sqrt(0.5*D.*(1-
((cos(Betta*pi/180)).^2).*((sin(phi0*pi/180)).^2))./((sin(Betta*pi/180)
).^2));
lambda = (((F./F_calc)).^2)*100000;    % Wavelength (cm)
figure(1)
plot(p0, lambda);
xlabel('Distance (km)');
ylabel('cm');
title('Wavelength in cm');

%-----
% Calculate F
%-----
lambda = 3.6e-5;      % Wavelength assumed to be 3.6 cm
F = sqrt(0.5*lambda*D.*(1-
((cos(Betta*pi/180)).^2).*((sin(phi0*pi/180)).^2))./((sin(Betta*pi/180)
).^2));

%-----
% Calculating and plotting spacecraft speed
%-----
p0_pp = spline(SCET',p0');          % Piecewise polynomial
[breaks_p0,coefs_p0,l_p0,k,d] = unmkpp(p0_pp); % Extracts information
                                         % from p0_pp.
                                         % breaks_p0 is the break points of each piece.
                                         % coefs is polynomial coefficients for each
                                         % piece
                                         % l_p0 is the total number of pieces
                                         % k contains the order of the polynomial for
                                         % each piece.
                                         %
for i = 1 : l_p0,                    % Calculate derivative of each piece
    pTemp = polyder(coefs_p0(i,:));
    pTemp = [zeros(1,3 - length(pTemp)) pTemp]; % The piece may
                                         % characterize a polynomial less than the third
                                         % order, we have to compensate for the zero co-
                                         % efficients which result from taking the
                                         % derivative.
    p0_derCoef(i,:) = pTemp;

```

```

end
for i = 1 : l_p0,          % Calculate derivative at starting point of
                           % each piece
    dp0dt_calc(i,1) = polyval(p0_derCoef(i,:),0);
end
% Calculate derivative of last point in last piece
dp0dt_calc(l_p0+1,1) = polyval(p0_derCoef(l_p0,:),breaks_p0(l_p0+1)-
breaks_p0(l_p0));
figure(2)
plot(p0, dp0dt);
xlabel('distance');
ylabel('km/sec');
title('Spacecraft speed');
hold
plot(p0, dp0dt_calc,'g');
legend('Speed in file','Speed calculated');
hold off
figure(3)
plot(p0, dp0dt-dp0dt_calc);
xlabel('distance');
ylabel('km/sec');
title('Difference between spacecraft speed existing in geometry file
and spacecraft speed calculated by data');

%-----
% Calculating dF/dt
% -----
F_pp = spline(SCET',F');          % Piecewise polynomial
[breaks_F,coefs_F,l_F,k,d] = unmkpp(F_pp); % coefs is polynomial
                                         % coefficients
for i = 1 : l_F,                  % Calculate derivative of each
                                   % piece
    pTemp1 = polyder(coefs_F(i,:));
    pTemp1 = [zeros(1,3 - length(pTemp1)) pTemp1];
    F_derCoef(i,:) = pTemp1;
end
for i = 1 : l_F,                  % Calculate derivative of each
                                   % piece
    dFdt_calc(i,1) = polyval(F_derCoef(i,:),0);
end
dFdt_calc(l_F+1,1) = polyval(F_derCoef(l_F,:),breaks_F(l_F+1)-
breaks_F(l_F));
figure(4)
plot(p0, dFdt-dFdt_calc);
xlabel('distance');
ylabel('km/sec');
title('Difference between dF/dt existing in geometry file and dF/dt
calculated by data');
figure(5)
plot(p0, dFdt);
xlabel('distance');
ylabel('km/sec');
title('dF/dt (km/s)');
hold

```

```

plot(p0, dFdt_calc, 'g');
legend('F in file', 'F calculated');
hold off

%-----
% Calculating dF/dp0
% -----

dFdp0_calc = dFdt_calc./dp0dt_calc;
dFdp0 = dFdt./dp0dt; % Calculating from data in file
figure(6)
plot(p0, dFdp0);
xlabel('Distance(km)');
ylabel('\delta F \delta p0');
title('dF/dp0');
hold
plot(p0, dFdp0_calc, 'g');
legend('from file', 'calculated');
hold off

%-----
% Re-calculating variables for uniform steps in p0, as this is what
% will be given to the reconstruction algorithm
% -----

p0_unif = [ceil(p0(length(p0),1)):floor(p0(1))]; % For ingress
B_interp = spline(p0', A, p0_unif);
figure(7)
plot(p0, SCET, 'ro', p0_unif, B_interp(1,:));
figure(8)
plot(p0, ERT, 'ro', p0_unif, B_interp(2,:));
figure(9)
plot(p0, p0, 'ro', p0_unif, B_interp(3,:));
figure(10)
plot(p0, phi0, 'ro', p0_unif, B_interp(6,:));
figure(11)
plot(p0, Betta, 'ro', p0_unif, abs(B_interp(8,:)));
figure(12)
plot(p0, D, 'ro', p0_unif, B_interp(9,:));
figure(13)
plot(p0, B_interp(:,10)', 'ro', p0_unif, B_interp(10,:));

```

5 MATLAB Codes for Chapter 5

5.1 VaryingResolution.m

```

%-----
% VaryingFResolution.m
% This program shows the effect of uncertainty in F on the radial
% resolution
% -----
Q = logspace(-2,1,200);

```

```

C = mfun('FresnelC',Q);
S = mfun('FresnelS',Q);
dR_dFn = Q./(C.^2+S.^2);
loglog(Q, dR_dFn);
grid
title('Effect of variable F on radial resolution');
xlabel('Q = \gamma W/2F');
ylabel('\Delta R_\delta F/\gamma F');

```

5.2 NormalizedResolutionAnalysis4VaryingF.m

```

%-----
% NormalizedResolutionAnalysis4VaryingF
% This program plots
% 1) Maximum normalized* resolution achievable in the presence of a
% Fresnel scale error
% 2) Normalized* filter width versus maximum achievable normalized*
% resolution in the presence of Fresnel scale error
% 3) Number of filter points if sampled at dRn/2 versus normalized*
% resolution. One needs to keep in mind that if a Kaiser window is
% multiplied by the filter then the achievable resolution is almost as
% half as expected.
% * With respect to F
%-----
clear all;
close all;
dRn = logspace(-3,-0.3,100);      % Normalized desired resolution
Gamma = dRn/1.25;                 % Maximum allowable Gamma
dFn = Gamma.^2/2;
Wn = 2 ./ Gamma;                  % Maximum normalized filter width that can be
                                   % used
N = Wn *2 ./ dRn;
% Plotting normalized filter width vs normalized resolution
figure(1)
loglog(dRn, Wn);
grid
xlabel('Normalized Resolution');
ylabel('Normalized filter width');
title('Normalized filter width versus normalized resolution');
% Plotting dF/F vs normalized resolution
figure(2)
loglog(dRn, dFn);
grid
xlabel('Normalized Resolution');
ylabel('\delta F/F');
title('\delta F/F versus normalized resolution');
% Plotting #of filter points if sampled at dRn/2 vs normalized
resolution
figure(3)
loglog(dRn, N);
grid

```

```

xlabel('Normalized Resolution');
ylabel('# of filter points');
title(sprintf('Number of filter points required to achieve the desired
normalized resolution if \nfilter is sampled at dRn/2'));

```

5.3 Fresnel scale update routines.m

5.3.1 FindFBlock.m (Uses binary search technique)

```

function j = FindFBlock(x,iFirst,iLast,dF,p)

% j = FindFBlock(x,iFirst,iLast,dF,p)
% This function finds index j such that for i = iFirst:j-1, where j
% <=iLast and p*dF <= x(1) - x(i) <= dF. x must be a matrix, in which
% its values are arranged in a descending order.
% Created by Katayoun Goudarzi 1/10/06

if (x(1) - x(iLast) <= dF)
    j = iLast;
    return;
end
if (iFirst == iLast - 1 && x(1) - x(iFirst) <= dF)
    j = iLast;
    return;
end
imid = floor((iFirst+iLast)/2);
if ((x(1)-x(imid) <= dF))
    if ((x(1)-x(imid)) >= p*dF)
        j = imid;
        return;
    else
        j = FindFBlock(x,imid,iLast,dF,p);
    end
else
    j = FindFBlock(x,iFirst,imid,dF,p);
end
end

```

5.3.2 UpdateF.m (Uses FindFBlock.m)

```

function F_updated = UpdateF(Fm,dR)
%
% UpdateF takes in the Fresnel scale matrix Fm, calculates how often
% Fresnel scale needs to be updated and returns a matrix which contains
% blocks with fixed Fresnel scales.Fm's elements should be arranged
% either in a descending or ascending order.
% F_updated = [StartingIndexOfThisBlock FixedFresnelScaleForthisBlock]

bFlip = false;          %Flip Fm if elements are in ascending order

```

```

if (Fm(length(Fm)) > Fm(1))
    Fm = fliplr(Fm);
    bFlip = true;
end

i_Now = 1; % What Fm element are we pointing to now
F = Fm(1); % F is the Fresnel scale from data point i_Now
           % to start of next segment.
UpdateCount = 1; % What segment are we in now?
while (i_Now < length(Fm))
    gamma = dR/(F*1.25); % gamma = sqrt(2dF/F);
    dF = 0.5*F*gamma^2; % Allowed dF in this segment
    i_NextOffset = FindFBlock(Fm(i_Now:length(Fm)),1,length(Fm) -
    i_Now+1,dF,0.98) - 1;
    iTemp = i_NextOffset + i_Now;
    F = Fm(iTemp - 1);
    F_updated(UpdateCount,:) = [F i_Now]; % Contains updated Fresnel
                                           % scale F and starting
                                           % index (i_Now) of the
                                           % segment. Fixed F for data
                                           % points i_Now till
                                           % i_Now of next row of
                                           % F_updated.
                                           % start of next segment
    i_Now = iTemp;
    UpdateCount = UpdateCount + 1;
    F = Fm(i_Now); % F for next segment
end

iRow = size(F_updated,1);
%-----
% Skip this section if you don't want to replace the Fresnel scale of
% each block with the mean value and instead you want to keep the
% smallest F in the block.
%-----
for i = 1:iRow % Replace Fresnel scale of each block with mean
               % value of F in that block
    iBlockStart = F_updated(i,2);
    if (i == iRow)
        iBlockStop = length(Fm);
    else
        iBlockStop = F_updated(i+1,2)-1;
    end
    Ftemp = Fm(iBlockStart:iBlockStop);
    F_updated(i,1) = mean(Ftemp);
end
%-----

if (bFlip && (iRow > 1)) % Correct for indices if Fm
was flipped
    F_updated = flipud(F_updated);

    while (iRow > 1)
        F_updated(iRow,2) = length(Fm) + 2 - F_updated(iRow - 1,2);
    end
end

```

```

        iRow = iRow - 1;
    end
    F_updated(1,2) = 1;
End

```

5.3.3 BlockUpdateF.m

```

function F_Updated = BlockUpdateF(Fm,dR, BlockLength)
%
% BlockUpdateF(Fm, dR, BlockLength), takes in the row Fresnel scale
% matrix divides it into blocks of data with length specified
% (BlockLength). The algorithm then searches through each block and
% assigns an appropriate fixed Fresnel scale to the block based on the
% desired resolution (dR) such that the effect of distortion caused by
% Fresnel scale mismatch in the reconstruction algorithm is minimized.

nBlock = ceil(length(Fm)/BlockLength); % Number of Blocks to search
FminPrev = min(Fm(1:BlockLength)); % Minimum F in previous block
FmaxPrev = max(Fm(1:BlockLength)); % Maximum F in previous block
FavgPrev = mean(Fm(1:BlockLength)); % Average F in previous block
UpdateCount = 1; % What segment are we in now?
i_Now = 1; % What Fm element are we pointing to now
for (i = 1: nBlock) % Search through all the blocks except the last
    % one
    if (i == nBlock)
        FBlock = Fm((i-1)*BlockLength + 1:length(Fm));
    else
        FBlock = Fm((i-1)*BlockLength + 1:i*BlockLength);
    end
    FminBlock = min(FBlock);
    FmaxBlock = max(FBlock);
    FavgBlock = mean(FBlock);
    if (FminPrev < FminBlock) % Find minimum of previous and
        Fmin = FminPrev; % current blocks
    else
        Fmin = FminBlock;
    end
    if (FmaxPrev > FmaxBlock) % Find minimum of previous and
        Fmax = FmaxPrev; % current blocks
    else
        Fmax = FmaxBlock;
    end
    if (i == nBlock)
        Favg = mean(Fm(i_Now:length(Fm)));
    else
        Favg = mean(Fm(i_Now:i*BlockLength)); % Find average F of
    end
    % current and previous blocks
    gamma = dR/(Fmax*1.25); % gamma = sqrt(2dF/F)
    dF_allowed = 0.5*Fmax*gamma^2; % Allowed dF in this segment
    dFmax = Fmax - Favg;
    if (Favg - Fmin > dFmax)

```



```

        dFmax = Favg - Fmin;
    end
    if (dFmax > dF_allowed)
        F_Updated(UpdateCount,1) = FavgPrev;
        F_Updated(UpdateCount,2) = i_Now;
        i_Now = (i-1)*BlockLength + 1;
        UpdateCount = UpdateCount + 1;
        gammaBlock = dR/(FmaxBlock*1.25);    % gamma = sqrt(2dF/F)
        dFBlock = 0.5*FmaxBlock*gammaBlock^2;% Allowed dF in this
                                           % segment
        if((FavgBlock - FminBlock > dFBlock) | (FmaxBlock - FavgBlock >
dFBlock))
            error('Please reduce your block length.\n');
        end
        FminPrev = FminBlock;
        FmaxPrev = FmaxBlock;
        FavgPrev = FavgBlock;
    else
        FminPrev = Fmin;
        FmaxPrev = Fmax;
        FavgPrev = Favg;
    end
end
% Enter data for last segment
F_Updated(UpdateCount,1) = FavgPrev;
F_Updated(UpdateCount,2) = i_Now;

```

5.4 TestUpdateAlgorithms.m

```

% -----
% This program reads geometric parameters from a text file and tests
% Fresnel sacle update algorithms.
% Created by Kathy Goudarzi
% -----
clear all; close all;
format long e;
fid = fopen('C:\Documents and
Settings\Owner\Desktop\Rev7Ing_Geometry.txt');
A=fscanf(fid,'%g %g %g %g %g %g %g %g %g %g %g %g %g %g', [14 inf]);
B=A';
SCET = B(:,1);          % Spacecraft event time (past midnight, secs)
ERT = B(:,2);          % Earth receive time (Greenwich mean time,
secs)
p0 = B(:,3);           % Radius of intercept point (km)
IP = B(:,4);           % Impact parameter (km)
phi0_Rel = B(:,5);     % Angle phi0 relative to an inertial reference
% (degrees)
phi0 = B(:,6);         % Angle phi0 with respect to projected Earth
% direction (degrees)
PhiDiff = B(:,7);      % Difference between phi0_Rel and Phi0
% (degrees)
Betta = abs(B(:,8));   % Ring opening angle (degrees)

```

```

D = B(:,9);          % Distance from spacecraft to intercept point
                      % (km)
F = B(:,10);         % Fresnel scale (km), this should be calculated
                      % from geometry file. This has already been
                      % calculated and inserted in the sample
                      % geometry file being used.
dFdt = B(:,11);      % dF/dt (km/Sec), must be calculated
dp0dt = B(:,12);     % dp0/dt (km/sec), must be calculated
dp0dt2 = B(:,13);    % dp0dt/dt (km/sec^2), must be calculated
d = B(:,14);         % d^2(dp0dt)/dt^2 , must be calculated
fclose(fid);
lambda = 3.6e-5;
F_updatedByUpdateF=UpdateF(F', 1);
F_updatedByBlockUpdate=BlockUpdateF(F,1,2900);

```

5.5 ReconstructF.m

```

function [T, X] = ReconstructF(Thatm, Fm, Xi_start, dR, dxi, Beta)
%-----
%
% This function performs a reconstruction algorithm on the
% diffraction-limited data That and returns T.
% Thatm: Diffraction limited image plane transmittance matrix
% Fm: Matrix containing info about constant Fresnel scale blocks
% Xi_start: Starting location of That data
% dR: Resolution desired (km) (dRn = dR/F)
% Beta: If greater than zero it is the Kaiser window parameter, if it
% is zero then a rectangular window is used for reconstruction.
% T: The reconstructed object plane transmittance.
% X: Distances corresponding to T
%
% CHANGE LOG:
%
% 27Dec2005   Created.   K.Goudarzi
%-----
%-----
% F varies over distance. Divide Thatm into segments with Fixed F
% Start moving across That, and update for F
%-----
F_updated = Fm;
%-----
% Create That for each segment
%-----
Nhi_prev = 0;          % Filter length of previous
                        % segment - 1
for segment=1:size(F_updated, 1)
    F = F_updated(segment, 1); % Read F for this segment
    iStart = F_updated(segment, 2); % Segment starts from this data
    % point
    if (segment ~= size(F_updated, 1))
        iStop = F_updated(segment + 1, 2) - 1; % Segment stops at this
        % data point
    end
end

```

```

else
    iStop = length(Thatm);          % If we're in last segment, segment
                                   % stops at last data point
end
dx = dxi;                          % Image plane and object plane sampling
                                   % intervals are the same
Leff = (2*F^2)/dR                  % Filter length in km
Nhi = ceil(Leff/abs(dx));           % Nhi+1 is length of the filter
if (mod(Nhi,2) ~= 0) % Make sure that Nhi is an even number (For
                                   % easier calculations)
    Nhi = Nhi + 1;
end
if (segment ~= 1) % Move into previous segment in order not to
                 % loose data
    iStart = iStart - (Nhi + Nhi_prev)/2;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
*** Define distances
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (segment == 1)
    X_start = Xi_start + Nhi*dx/2;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
*** Define segment of image plane transmittance to be reconstructed
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

That = Thatm(iStart:iStop);
Ni = length(That)-1 % Ni+1 samples in the image plane are
                   % considered for reconstructing the object
                   % plane transmittance
if(Nhi >= Ni)
    error('Length of inverse Fresnel filter must be smaller than
    the length of the image plane transmittance.\n Either increase
    the image plane transmittance data points or reduce the
    resolution');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
*** Defining the reconstruction filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Xh = [-Nhi/2:Nhi/2]*abs(dx)/F;      % Range of function h
hi = ((1+j)/(2))*exp((-j*pi*(Xh).^2)/2);
if(Beta>0)
    Kaiswin = kaiser(Nhi+1,Beta);    % The reconstruction filter

```

```

        hi = hi.*Kaiswin';
    end
    S = abs(sum(hi));
    hi = hi/S;

    %*****
    %** Using add_overlap method to calculate the object plane
    %** transmittance
    %*****

    tic
    Tsegment = fftfilt(hi,That);
                    % Uses the overlap-add method, always uses
                    % power 2 fft points. It always chooses blocks
                    % to minimize overall fft calculations.

    t_FFT = toc

    if (segment == 1)
        T = Tsegment(length(hi):length(That)); % Only considering
                    % points that have had enough datapoints for
                    % reconstruction
    else
        Ttemp = Tsegment(length(hi):length(That));
        T = [T Ttemp];
    end
    end
    Nhi_prev =Nhi;
end
X = X_start + [0:length(T)-1]*dx;

```

5.6 ImpulseResponseVaryingF.m

```

%-----
% To = delta(x-x_middle), the object plane transmittance is a delta
% function
% The diffracted transmittance(That) in the image plane is given to the
% reconstruction algorithm and the object plane transmittance is
% recovered. (T)
%
%-----

clear all;
close all;

F1 = 5;                % Fresnel scale in km
dRn = .1;              % Normalized Resolution of That data in km
dxon = .01;            % Normalized Sampling intervals
if(dRn < dxon)
    error('Sampling intervals must be finer than the desired
    resolution');
end

```

```

Beta = 0;

%-----
%Creating the diffraction-limited pattern
%-----
dxin = dxon;
Ln = (2)/dRn;           % Normalized length of filter used
Nhlf = ceil(Ln/dxin);   % Nhlf+1 is the number of points in the
                        % forward-Fresnel filter

X_middlen = 0;
Xi_startn = -Nhlf*dxin;
Xin = Xi_startn*ones(1,2*Nhlf+1) + [0:2*Nhlf]*dxin; % Normalized distance
That = ((1-j)/(2*F1))*exp(j*0.5*pi*((Xin-X_middlen).^2));
magThat = abs(That);
magThatdB=20*log10(magThat);
angThat = angle(That)*180/pi;

%-----
% Reconstructing the object plane transmittance
%-----
F2 = 5.05;              % Known characteristic Fresnel scale, contains
                        % error
Fm = UpdateF(F2 * ones(1, length(That)),dRn*F1);
[T, X] = ReconstructF(That, Fm, Xi_startn*F1, dRn*F1, dxin*F1, Beta);

magT = abs(T);
magTdB=20*log10(magT);
angT = angle(T)*180/pi;

%-----
% Theoretical response
%-----
Xth = X;
Xthn = Xth/F1;
Tth = ((Nhlf*dxin)/(2*F1))*sinc(((Nhlf*dxin)/2)*(Xthn-
X_middlen)).*exp(j*pi*0.5*((Xthn-X_middlen).^2));

%-----
% Redefining resolution for kaiser window
%-----

if(Beta ~= 0)
    dRn = 2*dRn;
end

%-----
% Plotting the image plane transmittance
%-----

figure(1)
plot(Xin, magThat);

```

```

title(sprintf('Magnitude of the image plane transmittance,Normalized
sampling interval Xs=%g , \n Normalized resolution of %g, Beta =
%1.1f',dxin,dRn,Beta))
xlabel('distance');

figure(2)
plot(Xin, magThatdB);
title(sprintf('Magnitude of the image plane transmittance in
dB,Normalized sampling interval Xs=%g , \n Normalized resolution of %g,
Beta = %1.1f',dxin,dRn,Beta))
xlabel('distance');
ylabel('dB');

figure(3)
plot(Xin, angThat);
title(sprintf('Phase of the image plane transmittance,Normalized
sampling interval Xs=%g , \n Normalized resolution of %g, Beta =
%1.1f',dxin,dRn,Beta))
xlabel('distance');
ylabel('degrees');

%-----
% Plotting the reconstructed object plane transmittance
%-----

figure(4)
plot(X, magT);
title('Magnitude of the reconstructed object plane transmittance')
xlabel('distance');

figure(5)
plot(X, magTdB);
title('Magnitude of the reconstructed object plane transmittance in
dB')
xlabel('distance');
ylabel('dB');

figure(6)
plot(X, angT);
title('Phase of the reconstructed object plane transmittance,')
xlabel('distance');
ylabel('degrees');

%-----
% Plotting the theoretical reconstructed object plane transmittance
%-----

magTth= abs(Tth);
magTthdB=20*log10(magTth);
angTth = angle(Tth)*180/pi;

figure(7)
plot(Xth, magTth);

```

```

title('Magnitude of the theoretical reconstructed object plane
transmittance')
xlabel(' distance');

figure(8)
plot(Xth, magTthdB);
title('Magnitude of the theoreticalreconstructed object plane
transmittance in dB ')
xlabel('distance');
ylabel('dB');

figure(9)
plot(Xth, angTth);
title('Phase of the theoretical reconstructed object plane
transmittance ')
xlabel('distance');
ylabel('degrees');
%-----
%Plotting the reconstructed and theoretical on the same plot
%-----

figure(10)
plot(X,magT, 'k',X,magTth, 'r--');
xlabel('distance');
ylabel('magnitude');
title('comparison of the reconstructed and the theoretical impulse
response magnitude');
legend('Reconstructed impulse response','Theoretical impulse
response');

figure(11)
plot(X,magTdB, 'k',X,magTthdB, 'r--');
xlabel('distance');
ylabel('magnitude (dB)');
title('comparison of the reconstructed and the theoretical impulse
response magnitude (dB)');
legend('Reconstructed impulse response','Theoretical impulse
response');

figure(12)
plot(X,angT, 'k',X,angTth, 'r--');
xlabel('distance');
ylabel('degrees');
title('comparison of the reconstructed and the theoretical impulse
response phase');
legend('Reconstructed impulse response','Theoretical impulse
response');

```

5.7 GrayEdgeSimulationVaryingF.m

```
%-----
% GrayEdgeSimulationVaryingF.m
% The intention in this program is to study effect of Fresnel scale
% mismatch on the reconstruction of an Edge.
% All distances below are in km
%
%-----

clear all;
close all;

dR = .5;           % Desired resolution in km
dxo = .1;          % Sampling intervals in km

%-----
% Defining the object plane transmittance containing an Edge
%-----
tau = 8;
Xo_start = 0;      % Starting distance of object plane
transmittance
Xo_stop = 3000;
Xo_edge = (Xo_start + Xo_stop)/2;
Nxo = round((Xo_stop-Xo_start)/dxo);
Xo = Xo_start*ones(1,Nxo)+[0:Nxo-1]*dxo;
if(Nxo ~= length(Xo))
    error('Error! Check Nxol and Xol');
end
To = ones(1,Nxo);
indxo = find (Xo < Xo_edge);
To(indxo) = ones(size(indxo))*exp(-tau/2);

magTo = abs(To);
magTodB=20*log10(magTo);
angTo = angle(To)*180/pi;

%-----
% Creating the diffraction-limited data with fresnel scale of F1
%-----

F1 = 5;
dRn = dR/F1;      % Normalize desired resolution
dxon = dxo/F1;     % Normalized sampling intervals
Xo_startn = Xo_start/F1;
if(dRn < dxon)
    error('Sampling intervals must be finer than the desired
resolution');
end
Beta = 3.5;
[That, Xi_startn, dxin, Nhfl] = SimulateThat(To, Xo_startn,dRn, dxon,
Beta);
Xin= Xi_startn*ones(size(That)) + [0:length(That)-1]*dxin;
```



```

Xi = Xin * F1;

Xi_Start = Xi(1);
Xi_edge = Xo_edge;
dxi = dxin*F1;
indxi = find (Xi_edge-Nhf*dxi<Xi & Xi<Xi_edge+Nhf*dxi);

magThat = abs(That);
magThatdB=20*log10(magThat);
angThat= angle(That)*180/pi;

%-----
% Reconstructing the object plane transmittance
%-----

F = 5.05* ones(1,length(That));
Fm = UpdateF(F,dRn*F1);
[T, X] = ReconstructF(That, Fm, Xi_Start, dR, dxin*F1, Beta);

magT = abs(T);
magTdB=20*log10(magT);
angT = angle(T)*180/pi;

X_edge = Xo_edge;

%-----
% Creating the theoretical image plane transmittance
%-----
Xith = Xo;
Xm = Xo_edge;
That_th = 0.5*(1-j)*(fcs((Xith-Xm)/F1)+0.5+0.5*j)+0.5*(1-j)*exp(-
0.5*tau)*(0.5+0.5*j+fcs(-(Xith-Xm)/F1));

magThat_th = abs(That_th);
magThat_thdB=20*log10(magThat_th);
angThat_th = angle(That_th)*180/pi;
%-----
% Reconstructing the object plane transmittance of the theoretical
% image plane transmittance
%-----
F = 5.005* ones(1,length(That_th));
Fm = UpdateF(F,dRn*F1);
[Tth, Xth] = ReconstructF(That_th, Fm, Xith(1), dR, dxin*F1, Beta);

magTth = abs(Tth);
magTthdB=20*log10(magTth);
angTth = angle(Tth)*180/pi;

X_edgeth = Xi_edge;
%-----
% Plotting the actual object plane transmittance
%-----

figure(1)

```

```

plot(Xo, magTo);
title('Actual object plane transmittance magnitude');
xlabel('Distance (km)');
grid;

figure(2)
plot(Xo, magTodB);
title('Actual object plane transmittance magnitude (dB)');
xlabel('Distance (km)');
ylabel('dB');
grid;

figure(3)
plot(Xo, angTo);
title('Actual object plane transmittance phase');
xlabel('Distance (km)');
ylabel('Degrees');
grid;

%-----
% Plotting the image plane transmittance
%-----

figure(4)
plot(Xi, magThat);
title('Magnitude of the simulated image plane transmittance,')
xlabel('Distance (km)');
grid;

figure(5)
plot(Xi, magThatdB);
title('Magnitude of the simulated image plane transmittance')
xlabel('Distance (km)');
ylabel('dB');
grid;

figure(6)
plot(Xi, angThat);
title('Phase of the simulated image plane transmittance')
xlabel('Distance (km)');
ylabel('Degrees');
grid;

%-----
% Plotting the reconstructed object plane transmittance
%-----

figure(7)
plot(X, magT);
title('Magnitude of the reconstructed object plane transmittance')
xlabel('Distance (km)');
grid;

figure(8)

```

```

plot(X, magTdB);
title('Magnitude of the reconstructed object plane transmittance in
dB')
xlabel('Distance (km)');
ylabel('dB');
grid;

figure(9)
plot(X, angT);
title('Phase of the reconstructed object plane transmittance')
xlabel('Distance (km)');
ylabel('Degrees');
grid;

%-----
% Plotting all three on the same plot (derived from simulated image
% plane transmittance)
%-----
indx0 = find(Xo_edge-Nhf*dxi<Xo & Xo<Xo_edge+Nhf*dxi);
indx = find(Xo_edge-Nhf*dxi<X & X<Xo_edge+Nhf*dxi);
figure(10)
plot(Xo(indxo),magTo(indxo),'k',Xi(indxi),magThat(indxi),'r',X(indx),ma
gT(indx),'b');
xlabel('Distance (km)');
ylabel('Magnitude');
legend('Actual object plane transmittance','Simulated image plane
transmittance','Reconstructed object plane transmittance');
grid;

figure(11)
plot(Xo(indxo),magTodB(indxo),'k',Xi(indxi),magThatdB(indxi),'r',X(indx
),magTdB(indx),'b');
xlabel('Distance (km)');
ylabel('Magnitude (dB)');
legend('Actual object plane transmittance','Simulated image plane
transmittance','Reconstructed object plane transmittance');
grid;

figure(12)
plot(Xo(indxo),angTo(indxo),'k',Xi(indxi),angThat(indxi),'r',X(indx),an
gT(indx),'b');
xlabel('Distance (km)');
ylabel('Degrees');
legend('Actual object plane transmittance','Simulated image plane
transmittance','Reconstructed object plane transmittance');
grid;
%-----
% Plotting all three on the same plot (derived from theoretical image
% plane transmittance)
%-----

figure(13)
plot(Xo,magTo,'k',Xith,magThat_th,'r',Xth,magTth,'b');
xlabel('Distance (km)');

```

```

ylabel('Magnitude');
legend('Actual object plane transmittance','Theoretical Image plane
transmittance','Reconstructed object plane transmittance');
grid;

figure(14)
plot(Xo,magToB,'k',Xith,magThat_thdB,'r',Xth,magTthdB,'b');
xlabel('Distance');
ylabel('Magnitude (dB)');
legend('Actual object plane transmittance','Theoretical Image plane
transmittance','Reconstructed object plane transmittance');
grid;

figure(15)
plot(Xo,angTo,'k',Xith,angThat_th,'r',Xth,angTth,'b');
xlabel('Distance');
ylabel('Degrees');
legend('Actual object plane transmittance','Theoretical Image plane
transmittance','Reconstructed object plane transmittance');
grid;
t=toc

```

5.8 ThreeGrayGapSimulationVaryingF.m

```

%-----
% ThreeGrayGapSimulationVaryingF.m
% That in this program contains three gaps with different Fresnel
% scales
%-----

clear all;
close all;

dR = .4;           % Desired resolution in km
dxo = .1;          % Sampling intervals in km

%-----
% Defining the object plane transmittance containing the three gaps
%-----
tau = 8;
Xo_start = 0;      % Starting index of object plane transmittance
Xo_stop = 8000;
GapWidth1 = 100;
GapWidth2 = 10;
GapWidth3 = 5;
Nxo = round((Xo_stop-Xo_start)/dxo);
Xo = Xo_start*ones(1,Nxo)+[0:Nxo-1]*dxo;
if(Nxo ~= length(Xo))
    error('Error! Check Nxol and Xol');
end
To = ones(1,Nxo);
Xo_edgeStart1 = Xo_start+2000;

```

```

Xo_edgeStop1 = Xo_edgeStart1 + GapWidth1;
Xo_edgeStart2 = Xo_start+4000;
Xo_edgeStop2 = Xo_edgeStart2 + GapWidth2;
Xo_edgeStart3 = Xo_start+6000;
Xo_edgeStop3 = Xo_edgeStart3 + GapWidth3;
indx01 = find (Xo < Xo_edgeStart1);
To(indx01) = ones(size(indx01))*exp(-tau/2);
indx02 = find (Xo > Xo_edgeStop1 & Xo < Xo_edgeStart2);
To(indx02) = ones(size(indx02))*exp(-tau/2);
indx03 = find (Xo > Xo_edgeStop2 & Xo < Xo_edgeStart3);
To(indx03) = ones(size(indx03))*exp(-tau/2);
indx04 = find (Xo > Xo_edgeStop3);
To(indx04) = ones(size(indx04))*exp(-tau/2);

Xo_edgeStart = Xo_edgeStart1;
Xo_edgeStop = Xo_edgeStop3;

magTo = abs(To);
magTodB=20*log10(magTo);
angTo = angle(To)*180/pi;

%-----
% Creating the diffraction-limited data with Fresnel scale of F1
%-----

F1 = 10;
dRn1 = dR/F1;           % Normalize desired resolution
dxon1 = dxo/F1;         % Normalized sampling intervals
Xo_startn1 = Xo_start/F1;
if(dRn1 < dxon1)
    error('Sampling intervals must be finer than the desired
resolution');
end
Beta = 3.5;
[That1, Xi_startn1, dxin1, Nh1] = SimulateThat(To, Xo_startn1,dRn1,
dxon1, Beta);
Xin1= Xi_startn1*ones(size(That1)) + [0:length(That1)-1]*dxin1;
Xil = Xin1 * F1;

%-----
% Creating the diffraction-limited data with fresnel scale of F2
%-----

F2 = 10.003;
dRn2 = dR/F2;           % Normalize desired resolution
dxon2 = dxo/F2;         % Normalized sampling intervals
Xo_startn2 = Xo_start/F2;
if(dRn2 < dxon2)
    error('Sampling intervals must be finer than the desired
resolution');
end
Beta = 3.5;
[That2, Xi_startn2, dxin2, Nh2] = SimulateThat(To, Xo_startn2,dRn2,
dxon2, Beta);

```

```

Xin2= Xi_startn2*ones(size(That2)) + [0:length(That2)-1]*dxin2;
Xi2 = Xin2 * F2;

%-----
% Creating the diffraction-limited data with Fresnel scale of F3
%-----

F3 = 10.006;
dRn3 = dR/F3;           % Normalize desired resolution
dxon3 = dxo/F3;         % Normalized sampling intervals
Xo_startn3 = Xo_start/F3;
if(dRn3 < dxon3)
    error('Sampling intervals must be finer than the desired
resolution');
end
Beta = 3.5;
[That3, Xi_startn3, dxin3, Nh3] = SimulateThat(To, Xo_startn3,dRn3,
dxon3, Beta);
Xin3= Xi_startn3*ones(size(That3)) + [0:length(That3)-1]*dxin3;
Xi3 = Xin3 * F3;

%-----
% Combining these three patterns to create one diffraction pattern
%-----
Xi_edgeStop1 = Xo_edgeStop1;
Xi_edgeStart2 = Xo_edgeStart2;
Xi_edgeStop2 = Xo_edgeStop2;
Xi_edgeStart3 = Xo_edgeStart3;
indxi1 = find (Xi1 < (Xi_edgeStop1+Xi_edgeStart2)/2 );
indxi2 = find (Xi2 >= (Xi_edgeStop1+Xi_edgeStart2)/2 & Xi2 <=
(Xi_edgeStop2+Xi_edgeStart3)/2);
indxi3 = find (Xi3 > (Xi_edgeStop2+Xi_edgeStart3)/2 );

That = [That1(indxi1) That2(indxi2) That3(indxi3)];
Xi = [Xi1(indxi1) Xi2(indxi2) Xi3(indxi3)];
Xi_Start = Xi(1);
Xi_edgeStart = Xo_edgeStart;
Xi_edgeStop = Xo_edgeStop;
dxi = dxo;
indxi = find (Xi_edgeStart-Nh3*dxi<Xi & Xi<Xi_edgeStop+Nh3*dxi);

magThat = abs(That);
magThatdB=20*log10(magThat);
angThat= angle(That)*180/pi;

%-----
% Reconstructing the object plane transmittance from simulated data
%-----
dRn = dRn1;
dxin = dxin1;
%F = F3 * ones(1,length(That)); % To view oscillation
F = [F1*ones(1,length(indxi1)) F2*ones(1, length(indxi2)) F3*ones(1,
length(indxi3))];
Fm = UpdateF(F,dR);

```

```

[T, X] = ReconstructF(That, Fm, Xi_Start, dR, dxin*F1, Beta);

magT = abs(T);
magTdB=20*log10(magT);
angT = angle(T)*180/pi;

X_edgeStart = Xi_edgeStart;
X_edgeStop = Xi_edgeStop;

if(Beta ~= 0)
    dRn = 2*dRn;
end

%-----
% Theoretical diffraction-limited data
%-----
L1 = GapWidth1/2;
Xm1 = Xo_edgeStart1 + L1;
That_th = 0.5*(1-j)*(fcs((Xo-Xm1+L1)/F1)-fcs((Xo-Xm1-L1)/F1))+0.5*(1-
j)*(1/3)*exp(-0.5*tau)*(0.5+0.5*j-fcs((Xo-Xm1+L1)/F1))+0.5*(1-
j)*(1/3)*exp(-0.5*tau)*(fcs((Xo-Xm1-L1)/F1)+0.5+0.5*j);
L2 = GapWidth2/2;
Xm2 = Xo_edgeStart2 + L2;
That_th = That_th + 0.5*(1-j)*(fcs((Xo-Xm2+L2)/F2)-fcs((Xo-Xm2-
L2)/F2))+0.5*(1-j)*(1/3)*exp(-0.5*tau)*(0.5+0.5*j-fcs((Xo-
Xm2+L2)/F2))+0.5*(1-j)*(1/3)*exp(-0.5*tau)*(fcs((Xo-Xm2-
L2)/F2)+0.5+0.5*j);
L3 = GapWidth3/2;
Xm3 = Xo_edgeStart3 + L3;
That_th = That_th + 0.5*(1-j)*(fcs((Xo-Xm3+L3)/F3)-fcs((Xo-Xm3-
L3)/F3))+0.5*(1-j)*(1/3)*exp(-0.5*tau)*(0.5+0.5*j-fcs((Xo-
Xm3+L3)/F3))+0.5*(1-j)*(1/3)*exp(-0.5*tau)*(fcs((Xo-Xm3-
L3)/F3)+0.5+0.5*j);
% The free space region of That_th above has an extra offset of
% (2/3)*exp(-tau/2) on top of it.
That_th = That_th - 0.5*(1-j)*(2/3)*exp(-0.5*tau)*(fcs((Xo-Xm1+L1)/F1)-
fcs((Xo-Xm1-L1)/F1));
That_th = That_th - 0.5*(1-j)*(2/3)*exp(-0.5*tau)*(fcs((Xo-Xm2+L2)/F2)-
fcs((Xo-Xm2-L2)/F2));
That_th = That_th - 0.5*(1-j)*(2/3)*exp(-0.5*tau)*(fcs((Xo-Xm3+L3)/F3)-
fcs((Xo-Xm3-L3)/F3));

magThat_th = abs(That_th);
magThat_thdB=20*log10(magThat_th);
angThat_th = angle(That_th)*180/pi;
%-----
% Reconstructing the object plane transmittance from the theoretical
% data
%-----
[Tth, Xth] = ReconstructF(That_th, Fm, Xo(1), dR, dxin*F1, Beta);

magTth = abs(Tth);
magTdBth=20*log10(magTth);
angTth = angle(Tth)*180/pi;

```

```

%-----
% Plotting the actual object plane transmittance
%-----

figure(1)
plot(Xo, magTo);
title('Actual object plane transmittance magnitude');
xlabel('Distance(km)');
grid

figure(2)
plot(Xo, magTodB);
title('Actual object plane transmittance magnitude (dB)');
xlabel('Distance(km)');
ylabel('dB');
grid

figure(3)
plot(Xo, angTo);
title('Actual object plane transmittance phase');
xlabel('Distance(km)');
ylabel('Degrees');
grid
%-----
% Plotting the image plane transmittance
%-----

figure(4)
plot(Xi, magThat);
title('Magnitude of the image plane transmittance,');
xlabel('Distance(km)');
grid

figure(5)
plot(Xi, magThatdB);
title('Magnitude of the image plane transmittance')
xlabel('Distance(km)');
ylabel('dB');
grid

figure(6)
plot(Xi, angThat);
title('Phase of the image plane transmittance')
xlabel('Distance(km)');
ylabel('Degrees');
grid
%-----
% Plotting the reconstructed object plane transmittance
%-----

figure(7)
plot(X, magT);
title('Magnitude of the reconstructed object plane transmittance')

```



```

xlabel('Distance(km) ');
grid

figure(8)
plot(X, magTdB);
title('Magnitude of the reconstructed object plane transmittance in
dB')
xlabel('Distance(km) ');
ylabel('dB');
grid

figure(9)
plot(X, angT);
title('Phase of the reconstructed object plane transmittance')
xlabel('Distance(km) ');
ylabel('Degrees');
grid
%-----
% Plotting all three on the same plot(derived from simulated image
plane
% transmittance)
%-----
indx0 = find(Xo_edgeStart-Nhf1*dxi<Xo & Xo<Xo_edgeStop+Nhf3*dxi);
indx = find(Xo_edgeStart-Nhf1*dxi<X & X<Xo_edgeStop+Nhf3*dxi);
figure(10)
plot(Xo(indxo),magTo(indxo),'k',Xi(indxi),magThat(indxi),'r',X,magT,'b'
);
xlabel('Distance(km) ');
ylabel('Magnitude');
legend('Actual object plane transmittance','Simulated image plane
transmittance','Reconstructed object plane transmittance');
grid

figure(11)
plot(Xo(indxo),magTodB(indxo),'k',Xi(indxi),magThatdB(indxi),'r',X,magT
dB,'b');
xlabel('Distance(km) ');
ylabel('Magnitude (dB)');
legend('Actual object plane transmittance','Simulated image plane
transmittance','Reconstructed object plane transmittance');
grid

figure(12)
plot(Xo(indxo),angTo(indxo),'k',Xi(indxi),angThat(indxi),'r',X,angT,'b'
);
xlabel('Distance(km) ');
ylabel('Degrees');
legend('Actual object plane transmittance','Simulated image plane
transmittance','Reconstructed object plane transmittance');
grid
%-----
%Plotting all three on the same plot(derived from theoretical image
plane transmittance)

```

```

%-----
figure(13)
plot(Xo,magTo,'k',Xo,magThat_th,'r',Xth,magTth,'b');
xlabel('Distance(km)');
ylabel('Magnitude');
legend('Actual object plane transmittance','Theoretical image plane
transmittance','Reconstructed object plane transmittance');
grid

figure(14)
plot(Xo,magTodB,'k',Xo,magThat_thdB,'r',Xth,magTdBth,'b');
xlabel('Distance(km)');
ylabel('Magnitude (dB)');
legend('Actual object plane transmittance','Theoretical image plane
transmittance','Reconstructed object plane transmittance');
grid

figure(15)
plot(Xo,angTo,'k',Xo,angThat_th,'r',Xth,angTth,'b');
xlabel('Distance(km)');
ylabel('Degrees');
legend('Actual object plane transmittance','Theoretical image plane
transmittance','Reconstructed object plane transmittance');
grid

```